

Vivado Design Suite Tcl コマンド リファレンス ガイド

UG835 (v2019.2) 2019 年 10 月 30 日

この資料は表記のバージョンの英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

改訂履歴

次の表に、この文書の改訂履歴を示します。

セクション	改訂内容
2019 年 10 月 30 日 v2019.2	
<code>close_hw_manager</code> 、 <code>delete_qor_suggestions</code> 、 <code>get_hw_ddrmcs</code> 、 <code>open_hw_manager</code> 、 <code>open_hw_platform</code> 、 <code>refresh_hw_ddrmc</code> 、 <code>report_hw_ddrmc</code> 、 <code>update_sw_parameters</code> 、 <code>validate_hw_platform</code> 、 <code>write_hw_platform</code> 、 <code>write_hw_platform_metadata</code>	2019.2 で追加されたコマンド
<code>apply_bd_automation</code> 、 <code>assign_bd_address</code> 、 <code>config_timing_analysis</code> 、 <code>connect_hw_server</code> 、 <code>create_bd_intf_port</code> 、 <code>create_bd_port</code> 、 <code>create_ip</code> 、 <code>export_as_example_design</code> 、 <code>get_bels</code> 、 <code>highlight_objects</code> 、 <code>launch_runs</code> 、 <code>make_bd_intf_pins_external</code> 、 <code>make_bd_pins_external</code> 、 <code>place_design</code> 、 <code>pr_recombine</code> 、 <code>pr_subdivide</code> 、 <code>program_hw_devices</code> 、 <code>read_checkpoint</code> 、 <code>read_qor_suggestions</code> 、 <code>report_incremental_reuse</code> 、 <code>report_ip_status</code> 、 <code>report_methodology</code> 、 <code>report_qor_suggestions</code> 、 <code>report_ram_utilization</code> 、 <code>synth_design</code> 、 <code>unhighlight_objects</code> 、 <code>write_checkpoint</code> 、 <code>write_ibis</code> 、 <code>write_project_tcl</code> 、 <code>write_qor_suggestions</code> 、 <code>write_xdc</code>	2019.2 で変更されたコマンド
<code>close_hw</code> 、 <code>launch_sdk</code> 、 <code>open_dsa</code> 、 <code>open_hw</code> 、 <code>report_sdx_utilization</code> 、 <code>validate_dsa</code> 、 <code>write_dsa</code> 、 <code>write_sysdef</code>	2019.2 で削除されたコマンド
2019 年 5 月 22 日 v2019.1	
<code>read_qor_suggestions</code> 、 <code>write_qor_suggestions</code> 、 <code>config_implementation</code> 、 <code>get_qor_suggestions</code> 、 <code>write_abstract_shell</code> 、 <code>get_bd_regs</code> 、 <code>report_config_implementation</code>	2019.1 で追加されたコマンド
<code>get_ports</code> 、 <code>report_ram_utilization</code> 、 <code>report_design_analysis</code> 、 <code>opt_design</code> 、 <code>report_exceptions</code> 、 <code>connect_bd_intf_net</code> 、 <code>write_hw_ila_data</code> 、 <code>get_ips</code> 、 <code>assign_bd_address</code> 、 <code>config_timing_analysis</code> 、 <code>report_disable_timing</code> 、 <code>read_iphys_opt_tcl</code> 、 <code>report_qor_suggestions</code> 、 <code>iphys_opt_design</code> 、 <code>report_control_sets</code> 、 <code>setup_ip_static_library</code>	2019.1 で変更されたコマンド
<code>write_dsa_rom</code>	2019.1 で削除されたコマンド

概要

Vivado の Tcl 機能の概要

Tcl (Tool Command Language) は Vivado[®] ツール環境に統合されているスクリプト言語です。Tcl は、アプリケーション プログラム インターフェイスの半導体業界標準言語で、SDC (Synopsys[®] Design Constraints) に使用されます。

SDC は、Synopsys 社の Synplify やその他のベンダー ツールから FPGA 合成ツールへタイミング制約を渡すメカニズムで、タイミング制約の業界標準なので、スクリプト言語には Tcl が最も適しています。

Tcl を使用することで、自動スクリプトだけでなく、デザイン ツールに対するインタラクティブなクエリを実行できます。Tcl には、デザイン データベースからツール、デザイン設定、ステートなどに関する情報をインタラクティブに取り出す機能があります。たとえば、特定のタイミング解析レポート コマンドを検索したり、制約を適用してその直後にクエリを実行し、ツールの手順を再実行せずに動作が予測どおりかどうかを確認できます。

次のセクションでは、Vivado での基本的な Tcl 機能について説明します。

注記: このガイドは、Tcl 言語の包括的なリファレンスではなく、Vivado Design Suite の Tcl シェル特定の機能を説明するものです。Tcl プログラムの追加情報に関しては、参考資料を示します。

Vivado Design Suite の起動

Vivado Design Suite は、さまざまな方法で起動できます。たとえば、非プロジェクト モードと呼ばれる Tcl スクリプトベースのコンパイル スタイル方法を使用して、ソースおよびデザイン プロセスをユーザーが自分で管理できます。または、プロジェクト モードと呼ばれるプロジェクトベースの方法を使用して、プロジェクトおよびプロジェクトステータスによりデザイン プロセスおよびデザイン データが自動的に管理されるようにすることもできます。どちらの方法も、Tcl スクリプトを使用したバッチ モードで、または Vivado IDE でインタラクティブに実行できます。異なるデザイン フロー モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

Tcl シェル モード

Tcl コマンドを直接使用する場合は、次のいずれかの方法を使用して Tcl コマンドでデザインを処理できます。

- Vivado IDE 環境外で Vivado Design Suite Tcl シェルに Tcl コマンドを入力する。
- Vivado IDE の下部にある [Tcl Console] ウィンドウに個々の Tcl コマンドを入力する。
- Vivado Design Suite Tcl シェルから Tcl スクリプトを実行する。
- Vivado IDE から Tcl スクリプトを実行する。

Vivado Design Suite Tcl シェルを起動するには、Linux コマンド プロンプトまたは Windows コマンド プロンプトに次のように入力します。

```
vivado -mode tcl
```



ヒント: Windows では、[Start]→[All Programs]→[Xilinx Design Tools]→[Vivado yyyy.x]→[Vivado yyyy.x Tcl Shell] をクリックします (yyyy.x はインストールされている Vivado のバージョン)。

Tcl および Tcl スクリプトの使用に関する詳細は、『Vivado Design Suite ユーザー ガイド: Tcl スクリプト機能の使用』(UG894) を参照してください。Vivado ツールで Tcl を使用する手順ごとの詳細な説明は、『Vivado Design Suite チュートリアル: デザイン フローの概要』(UG888) を参照してください。

Tcl バッチ モード

Vivado ツールを起動するときに Tcl スクリプトを指定すると、Vivado ツールをバッチ モードで使用できます。Linux コマンド プロンプトまたは Windows コマンド プロンプトに次のように入力します。

```
vivado -mode batch -source <your_Tcl_script>
```

Vivado Design Suite Tcl シェルが開き、指定の Tcl スクリプトを実行して、スクリプトが完了すると閉じます。バッチ モードでは、複数の Tcl スクリプトをキューに入れ、夜間に複数のデザインに対して合成、シミュレーション、インプリメンテーションを実行し、翌朝結果を確認できます。

バッチ モードで Tcl スクリプトを読み込む際、Vivado コマンドに引数を渡すこともできます。-tclargs オプションを使用すると、実行する Tcl スクリプトの引数を指定できます。次に例を示します。

```
vivado -mode batch -source script.tcl -tclargs "FPGA=115-2"
```



重要: 上記の例に示すように、Tcl 引数と値はクォーテーションで囲む必要があります。そうしないと、引数の処理でエラーが発生する可能性があります。

Vivado IDE モード

Vivado Design Suite は、さまざまな方法で起動できます。たとえば、非プロジェクト モードと呼ばれる Tcl スクリプト ベースのコンパイル スタイル方法を使用して、ソースおよびデザイン プロセスをユーザーが自分で管理できます。または、プロジェクト モードと呼ばれるプロジェクト ベースの方法を使用して、プロジェクトおよびプロジェクト ステータスによりデザイン プロセスおよびデザイン データが自動的に管理されるようにすることもできます。どちらの方法も、Tcl スクリプトを使用したバッチ モードで、または Vivado IDE でインタラクティブに実行できます。異なるデザイン フロー モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

GUI を使用する場合は、Windows または Linux で Vivado IDE を起動します。Vivado IDE の詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

Vivado IDE は作業ディレクトリから起動してください。デフォルトでは、Vivado ジャーナル ファイル、ログ ファイル、生成レポート ファイルは、Vivado ツールが起動されたディレクトリに生成されます。これにより、起動ディレクトリに書き込まれるプロジェクト ファイル、ログ ファイル、ジャーナル ファイルを見つけやすくなります。

Windows OS では、[Start]→[All Programs]→[Xilinx Design Tools]→[Vivado yyyy.x]→[Vivado yyyy.x Tcl Shell] をクリックします (yyyy.x はインストールされている Vivado のバージョン)。



ヒント: または、Windows デスクトップの Vivado IDE のショートカットをダブルクリックします。

Linux OS では、コマンド プロンプトに次のコマンドを入力します。

```
vivado -or- vivado -mode gui
```

Vivado ツール コマンド ライン実行ファイルのヘルプを表示するには、次のように入力します。

```
vivado -help
```

Vivado ツールを Vivado Design Suite Tcl シェルから実行している場合は、Tcl シェルから `start_gui` コマンドを使用して直接 Vivado IDE を開くことができます。

Vivado IDE を閉じて Vivado Tcl シェルに戻るには、Vivado IDE で `stop_gui` コマンドを使用します。

Tcl ジャーナル ファイル

Vivado ツールを起動すると、デザイン セッション中に実行されるコマンドおよび操作の記録が `vivado.log` ファイルに記述されます。`vivado.jou` というファイルも生成され、セッション中に実行された Tcl コマンドのジャーナルのみが記述されます。このジャーナル ファイルを基に、新しい Tcl スクリプトを』作成できます。

注記: Vivado ツールを起動すると、以前の run の詳細を保存するため、ジャーナル ファイルのバックアップ バージョンが `vivado_<id>.backup.jou` という名前で保存されます。`<id>` は、ログ ファイルおよびジャーナル ファイルの複数のバックアップ バージョンを保存するための固有の識別子です。

Tcl ヘルプ

Tcl の `help` コマンドを使用すると、サポートされる Tcl コマンドの概要が表示されます。

- `help`: Tcl コマンド カテゴリのリストを返します。

```
help
```

コマンド カテゴリは、File I/O などの特定のファンクションを実行するコマンド グループです。

- `help -category category`: 指定したカテゴリのコマンドのリストを返します。

```
help -category object
```

この例では、オブジェクトを処理する Tcl コマンドのリストが返されます。

- `help pattern`: 指定の検索パターンに一致したコマンドのリストを返します。この構文を使用すると、コマンド グループから特定のコマンドをすばやく検索できます。

```
help get_*
```

この例では、「get_」で始まる Tcl コマンドのリストが返されます。

- `help command`: 指定のコマンドに関する詳細情報を返します。

```
help get_cells
```

この例では、`get_cells` コマンドの詳細が表示されます。

- `help -args command`: 指定のコマンドの簡単な説明、コマンド構文、および各引数の説明を表示します。

```
help -args get_cells
```

- `help -syntax command`: 指定のコマンドの構文を表示します。

```
help -syntax get_cells
```

Tcl でのスクリプト機能

Tcl 初期化スクリプト



ヒント: 次に、Vivado をスタートアップ時にカスタマイズするために `Vivado_init.tcl` スクリプトを配置する場所を説明します。Vivado リリースでは、`Vivado_init.tcl` ファイルはデフォルトでは提供されません。

Vivado ツールを起動すると、次の 3 箇所で Tcl 初期化スクリプトが検索され、後に見つかったもので前に見つかったものよりも優先されます。

1. エンタープライズ: ツールのインストール ディレクトリ `<installdir>/Vivado/<version>/scripts/Vivado_init.tcl`
2. Vivado バージョン: Vivado Design Suite の特定のバージョン用のローカル ユーザー ディレクトリ
 - Windows 7: `%APPDATA%/Xilinx/Vivado/<version>/Vivado_init.tcl`
 - Linux: `$HOME/.Xilinx/Vivado/<version>/Vivado_init.tcl`
3. Vivado ユーザー: Vivado Design Suite 汎用のローカル ユーザー ディレクトリ
 - Windows 7: `%APPDATA%/Xilinx/Vivado/Vivado_init.tcl`
 - Linux: `$HOME/.Xilinx/Vivado/Vivado_init.tcl`

説明:

- `<installdir>`: Vivado Design Suite のインストール ディレクトリ。

`Vivado_init.tcl` がいずれかまたはすべての場所で見つかった場合、前述の順序でこのファイルが読み込まれます。

- インストール ディレクトリにある `Vivado_init.tcl` ファイルを使用すると、企業またはデザイン グループですべてのユーザーに対して共通の初期化スクリプトをサポートできます。そのインストール ディレクトリから Vivado ツールを起動すると、どのユーザーでも企業用の `Vivado_init.tcl` スクリプトが使用されます。
- ユーザーの `Vivado_init.tcl` ファイルを使用すると、各ユーザーがそれぞれコマンドを追加したり、デザイン要件を満たすためにツールのインストール ディレクトリに含まれるコマンドを変更できます。
- Vivado Design Suite のインストールでは `Vivado_init.tcl` ファイルは提供されません。特定のニーズに合わせて `Vivado_init.tcl` ファイルを作成し、インストール ディレクトリまたはホーム ディレクトリに配置してください。



ヒント: Vivado Design Suite のほかのツールでも、`<tool>_init.tcl` (`<tool>` は Vivado、vivado_lab、xsim、および xelab) という名前の初期化スクリプトがサポートされます。

`Vivado_init.tcl` スクリプトは標準の Tcl コマンド ファイルで、Vivado ツールでサポートされるどの Tcl コマンドも含めることができます。次の文を追加すると、`Vivado_init.tcl` から別の Tcl スクリプト ファイルを読み込むことができます。

```
source <path_to_file>/<file_name>.tcl
```

注記: または、コマンド ラインから Vivado Design Suite を起動する際に `-init` オプションを指定することもできます。詳細は、`vivado -help` と入力してください。

Tcl スクリプトの実行

Tcl スクリプトは、コマンド ライン オプションの 1 つとして指定するか、GUI で指定します。Vivado 統合設計環境 (IDE) で Tcl スクリプトを実行するには、[Tools]→[Run Tcl Script] をクリックします。

コマンド ラインから Tcl スクリプトを実行するには、次のコマンドを使用します。

```
source <file_name>
```

Tcl スクリプトを Vivado IDE から実行すると、進捗状況バーが表示され、スクリプトが終了するまですべての IDE 操作が実行できなくなります。

ランタイム中にスクリプトの実行を一時停止する方法はないので、標準的な OS でのプロセスを停止する方法 (kill) で強制終了するしかありません。この場合、最後に保存した後の作業が失われます。

Tcl コンソールに「help source」と入力すると、source コマンドに関する情報を表示できます。

Tcl.pre および Tcl.post フック スクリプトの使用

Tcl フック スクリプトを使用すると、合成 run またはインプリメンテーション run、あるいはインプリメンテーションの任意の段階の前 (tcl.pre) および後 (tcl.post) にカスタム Tcl スクリプトを実行できます。run を実行すると、定義済みの Tcl スクリプトが使用され、選択したストラテジに基づいて標準デザイン フローが実行されます。Tcl フック スクリプトにより、カスタム レポートを生成するなど、前後にプロセスを追加してこの標準フローをカスタマイズできます。Tcl フック スクリプトには、標準の Tcl スクリプトを使用する必要があります。

デザイン フローの各段階の前後でフック スクリプトを実行できます。一般的に、次のような使用方法があります。

- カスタム レポート: タイミング、消費電力、リソース使用率、またはユーザー定義の Tcl レポート。
- 一時的なパラメーター設定を使用して問題回避。
- フローの一部でタイミング制約を変更。
- ある段階を複数回実行 (phys_opt_design を複数回呼び出すなど)。
- ネットリスト、制約、またはデバイス プログラムを変更。



重要: tcl.pre および tcl.post スクリプト内の相対パスは、プロジェクトの run ディレクトリ <project>/<project.runs>/<run_name> を基準にしています。現在のプロジェクトまたは現在の run の DIRECTORY プロパティを使用して、Tcl フック スクリプト内の相対パスを定義できます。

```
get_property DIRECTORY [current_project] get_property DIRECTORY  
[current_run]
```

Tcl フック スクリプトの定義方法は、『Vivado Design Suite ユーザー ガイド: Tcl スクリプト機能の使用』 ([UG894](#)) を参照してください。

一般的な Tcl 構文のガイドライン

Tcl では、OS に関係なく Linux のファイル区切り文字 (/) が使用されます。

次のセクションでは、Vivado Design Suite で Tcl を使用する際の一般的な構文ガイドラインについて説明します。

eval コマンドの使用

Tcl コマンドを実行する際、Tcl コマンドで使用可能なまたは必須のコマンド ライン引数の代わりに変数置換を使用できます。ただしこの場合、Tcl eval コマンドを使用してコマンドの一部として Tcl 変数を含めたコマンド ラインを評価する必要があります。

たとえば、help コマンドには -category オプションを使用すると、コマンド カテゴリの 1 つを指定できます。

```
help -category ipflow
```

コマンド カテゴリを保持する変数を定義できます。

```
set cat "ipflow"
```

説明:

- set: 変数を定義する Tcl キーワードです。
- cat: 定義される変数の名前です。
- "ipflow": 変数に割り当てる値です。

変数は、Tcl コマンド内で評価できます。

```
eval help -category $cat
```

または

```
set cat "category ipflow" eval help $cat
```

ダブルクォーテーション(“)の代わりに波かっこ ({}) を使用しても同じ結果が得られます。

```
set runblocksOptDesignOpts { -sweep -retarget -propconst -remap }  
eval opt_design $runblocksOptDesignOpts
```

Tcl コンソールに「help eval」と入力すると、eval コマンドに関する情報を表示できます。

特殊文字の使用

コマンドの引数に、Tcl で特別な意味を持つ特殊文字が含まれることがあります。その場合、Tcl で誤った処理が実行されないように引数を波かっこ ({}) で囲む必要があります。次に、よく使用される特殊文字の例を示します。

バス インデックス: 角かっこ [] は Tcl で特別な意味を持つので、角かっこを使用するインデックス付きバス (ビットまたは一部選択) は波かっこで囲む必要があります。たとえば、角かっこを使用してバスのインデックス 4 を Vivado 波形ビューアーに追加するには、次のコマンドを使用します。

```
add_wave {bus[4]}
```

バスのインデックスには丸かっこも使用できますが、丸かっこは Tcl で特別な意味はないので、波かっこは必要ありません。次に例を示します。

```
add_wave bus(4)
```

Verilog のエスケープ文字: Verilog の予約文字またはキーワードを含む Verilog 識別子は、Verilog ソース コードおよびシミュレータ コマンド ラインで冒頭にバックスラッシュ (\) を、末尾にスペースを追加してエスケープ処理する必要があります。さらに Tcl コマンド ラインでは、このエスケープ処理された識別子を波かっこで囲む必要があります。

注記: 識別子に既に波かっこが含まれる場合、波かっこ内の波かっこも Tcl で予約文字として処理されるので、波かっこで囲む方法は使用できません。「VHDL 拡張識別子」に示す方法を使用する必要があります。

たとえば、「my_wire」という名前のワイヤを Vivado 波形ビューアーに追加するには、コマンドを次のように記述する必要があります。

```
add_wave {\my_wire }
```

注記: 最後の文字と閉じかっこの間にスペースを忘れずに追加してください。

Verilog ではどの識別子もエスケープ処理可能ですが、Tcl コマンド ラインでは、エスケープ処理の不要な識別子をエスケープ処理しないでください。たとえば、「w」という名前のワイヤを Vivado 波形ビューアーに追加する場合、Vivado シミュレータで次のコマンドは無効です。

```
add_wave {\w }
```

この識別子 w にはエスケープ処理は不要です。次のコマンドを使用する必要があります。

```
add_wave w
```

VHDL 拡張識別子: VHDL 拡張識別子には、Tcl の予約文字であるバックスラッシュ (\) が含まれます。バックスラッシュの後に閉じ波かっこがあると (\}), Tcl で閉じ波かっこと解釈されるので、VHDL 拡張識別子に波かっこを使用することはできません。各 Tcl 特殊文字の前にバックスラッシュを追加してください。たとえば、「\my_sig\」という信号を波形ビューアーに追加するには、コマンドを次のように記述する必要があります。

```
add_wave \my\_sig\
```

注記: 拡張識別子の一部である 2 つのバックスラッシュと、識別子の間のスペースの前にバックスラッシュが追加されています。

一般的な構文構造

Vivado Design Suite の Tcl コマンドの一般的な構文は、次のとおりです。

```
command [optional_parameters] required_parameters
```

コマンド構文は、アンダースコア (_) で区切られた「動詞 - 名詞」および「動詞 - 形容詞 - 名詞」の形になります。

コマンドは、関連するコマンドに同じ接頭辞が付けられ、分類されています。

- クエリを実行するコマンドには、通常 `get_` が接頭辞として付いています。
- 値やパラメーターを設定するコマンドには、通常 `set_` が接頭辞として付いています。
- レポートを生成するコマンドには、通常 `report_` が接頭辞として付いています。

これらのコマンドは、グローバル名前空間に属しており、コマンドに付属するサブコマンドはありません。

構文例

次に、`get_cells -help` コマンドを実行した例を示します。

```
get_cells

Description:
Get a list of cells in the current design

Syntax:
get_cells [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]
          [-of_objects <args>] [-match_style <arg>] [-quiet] [-verbose]
          [<patterns>]

Returns:
list of cell objects

Usage:
Name          Description
-----
[-hsc]        Hierarchy separator
               Default: /
[-hierarchical] Search level-by-level in current instance
[-regexp]     Patterns are full regular expressions
[-nocase]     Perform case-insensitive matching (valid only when -
regexp
               specified)
[-filter]     Filter list with expression
[-of_objects] Get cells of these pins, timing paths, nets, bels, sites
               or drc violations
[-match_style] Style of pattern matching
               Default: sdc
               Values: ucf, sdc
[-quiet]      Ignore command errors
[-verbose]    Suspend message limits during command execution
[-<patterns>] Match cell names against patterns
               Default: *
```

Categories:
SDC, XDC, Object

不明コマンド

Tcl には、通常サポートされるビルトイン コマンド、Tcl インタープリターに渡される Vivado 特有のコマンド、およびユーザー定義のプロシージャのリストが含まれます。

これらの既知のコマンドに含まれないコマンドは OS に送信され、`exec` コマンドからシェルで実行されます。これにより、OS 特有のシェル コマンドを実行できます。シェル コマンドがない場合、コマンドが見つからなかったことを示すエラー メッセージが表示されます。

戻りコード

Tcl コマンドの中には、オブジェクトのリストやコレクションなどの戻り値が出力されるものがあります。それ以外のコマンドでは、処理は実行されても、ユーザーが直接利用できるような値が返されるとは限りません。Tcl インターフェイスを統合したツールの一部には、コマンドでエラーのない場合は 0、エラーがある場合は 1 を返すものもあります。

Tcl コマンドまたはスクリプトのエラーを正しく処理するには、Tcl ビルトイン コマンドの `catch` を使用する必要があります。一般的には、`catch` コマンドと番号付き情報/警告/エラー メッセージに基づいて、Tcl スクリプトのフローで問題を評価します。

Vivado ツールの Tcl コマンドでは、コマンドの完了時に `TCL_OK` または `TCL_ERROR` が返され、標準の Tcl メカニズムによりグローバル変数 `$ERRORINFO` が設定されます。

`$ERRORINFO` 変数を使用する場合は、Tcl コンソールでエラーがレポートされた後に次を入力します。

```
puts $ERRORINFO
```

これにより、エラーの詳細情報が表示されます。たとえば、次のコード例では Tcl スクリプト (`procs.tcl`) が使用されており、ユーザー定義の手順 (`loads`) が実行されます。数行トランスクリプト メッセージが表示された後、5 行目にエラーが表示されます。

```
Line 1: Vivado % source procs.tcl
Line 2: Vivado% loads
Line 3: Found 180 driving FFs
Line 4: Processing pin a_reg_reg[1]/Q...
Line 5: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of-objects'.
Line 6: Vivado% puts $errorInfo
Line 7: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of-objects'.
       While executing "get_ports -of objects $pin" (procedure "my_report" line
6)
       invoked from within procs.tcl
```

Tcl スクリプト ファイルの `catch` 節に `puts $ERRORINFO` を追加し、エラーが見つかったときに詳細を表示するようにしたり、Tcl コンソールでエラーが発生したときに必要に応じて「`puts $errorInfo`」と入力して特定のエラーの詳細を表示できます。

上記のコード例では、6 行目に「`puts $ERRORINFO`」と入力することで、7 行目にエラーの詳細情報が表示されています。

ファースト クラスの Tcl オブジェクトとその関係

Vivado Design Suite の Tcl コマンドを使用すると、ネットリスト、デバイス、プロジェクトのオブジェクト モデルに直接アクセスできます。これらは Vivado ファースト クラス オブジェクトと呼ばれ、単なる文字列記述ではなく、操作およびクエリが可能であることを意味します。例外もありますが、通常はオブジェクトとしてクエリを実行できます。これらのオブジェクトには、クエリ可能なプロパティが含まれ、ほかのオブジェクトを取得できる関係があります。

オブジェクト タイプと定義

Vivado Design Suite には多くのオブジェクト タイプがありますが、ここでは基本的なタイプの定義と説明を示します。最も基本的で重要なオブジェクト タイプは、デザイン ネットリストのエンティティに関連するもので、次のものがあります。

- [Cell]: セルは、プリミティブまたはネットリスト内の階層のいずれかのインスタンスです。これには、フリップフロップ、LUT、I/O バッファ、RAM、DSP のほか、ほかのセルのグループのラッパーである 階層インスタンスが含まれます。

- **ピン:** ピンはセル上の論理接続ポイントです。ピンにより、セル内部が抽象化されて使用しやすくなります。ピンは、階層またはプリミティブのセル上に存在します。ピンには、クロック ピン、データ ピン、リセット ピン、フリップフロップの出力ピンなどが含まれます。
- **ポート:** ポートは、オブジェクト内の内容がオブジェクト外に接続するために使用する、オブジェクト境界にある接続です。最上位ネットリストまたはデザインのポートは通常タイ上の I/O パッドに接続され、デバイス パッケージのピンに接続されて、システム レベル デザインのデバイスに外部接続されます。階層セル、モジュール、またはエンティティ内のポートは、階層セルのピンとして表されます。
- **[Net]:** ネットは、物理的に直接相互接続される 1 つのワイヤまたは複数のワイヤです。ネットは階層またはフラットにできますが、常に一連のピンがまとめて分類されます。
- **クロック:** クロックは、デザイン内の順序ロジックに伝搬される周期的な信号です。クロックはプライマリ クロック ドメインにできるほか、DCM、PLL、MMCM などのクロック プリミティブで生成できます。クロックは UCF の TIMESPEC PERIOD 制約とほぼ同じで、スタティック タイミング解析アルゴリズムの基盤になっています。

オブジェクトのクエリ

ファーストクラス オブジェクトはすべて、通常次のように Tcl コマンド `get_*` を使用してクエリできます。

- `get_<object_type> <pattern>`

ここで `<pattern>` は検索パターンであり、必要に応じて階層区切り文字を使用して完全な名前を指定します。オブジェクトは通常、階層の各レベルで文字列パターンを一致させることによりクエリされます。検索パターンには次のようにワイルドカードも使用でき、オブジェクトを検索しやすくなっています。

- `get_cells */inst_1`

このコマンドでは、最上位のすぐ下の階層で `inst_1` という名前のセルが検索されます。階層のすべてのレベルで同じパターンを繰り返し検索する場合は、次の構文を使用してください。

- `get_cells -hierarchical inst_1`

このコマンドでは、`inst_1` に一致するインスタンスがすべての階層レベルで検索されます。

コマンド構文の詳細は、次のコマンドでヘルプ情報を参照してください。

- `help get_cells`
- `get_cells -help`

オブジェクト プロパティ

オブジェクトには、クエリを実行できるプロパティが含まれます。プロパティ名はオブジェクト タイプによって異なります。オブジェクトの特定のプロパティをクエリするには、次のコマンドを使用します。

- `get_property <property_name> <object>`

次の例では、セル オブジェクトの `lib_cell` プロパティをクエリしており、指定のインスタンスがどの UniSim コンポーネントにマップされているかがわかります。

- `get_property lib_cell [get_cell inst_1]`

指定したオブジェクトに使用可能なプロパティすべてを表示するには、`report_property` コマンドを使用します。

- `report_property [get_cells inst_1]`

次の表に、特定のオブジェクトに対して返されるプロパティを示します。

キー	値	データ型
bel	OLOGICE1.OUTFF	文字列
class	セル	文字列
job	TRUE	文字列
is_blackbox	0	ブール値
is_fixed	0	ブール値
is_partition	0	ブール値
is_primitive	1	ブール値
is_reconfigurable	0	ブール値
is_sequential	1	ブール値
lib_cell	FD	文字列
LOC	OLOGIC_X1Y27	文字列
name	error	文字列
primitive_group	FD_LD	文字列
primitive_subgroup	flop	文字列
site	OLOGIC_X1Y27	文字列
type	FD & LD	文字列
XSTLIB	1	ブール値

プロパティの中には、読み出し専用のものであれば、ユーザー設定が可能なものもあります。UCF や HDL でアノート可能な属性にマップされるプロパティは、通常 Tcl コマンドの `set_property` でユーザーが設定できます。

- `set_property loc OLOGIC_X1Y27 [get_cell inst_1]`

プロパティに基づいたフィルター処理

オブジェクトをクエリする `get_*` コマンドには、そのオブジェクトのプロパティ値に基づいてクエリをフィルター処理するオプションがあります。このオプションは、非常に優れたオブジェクト クエリ コマンド機能です。たとえば、プリミティブ タイプ FD のセルをすべてクエリするには、次のように入力します。

- `get_cells * -hierarchical -filter "lib_cell == FD"`

また、`==` 演算子を使用すると、文字列パターンでフィルター処理できます。たとえば、デザイン内のすべてのフリップフロップ タイプをクエリするには、次のように入力します。

- `get_cells * -hierarchical -filter "lib_cell =~ FD*"`

OR (`||`) や AND (`&&`) を使用すると、複数のプロパティ フィルターを組み合わせで検索できます。次の例では、デザイン内のすべてのセルから、フリップフロップ タイプで配置済みロケーション制約が設定されているものをクエリしています。

- `get_cells * -hierarchical -filter {lib_cell =~ FD* && loc != ""}`

注記: この例では、フィルター オプションの値が `"` ではなく、`{}` で囲まれています。これはインタープリターによるコマンド変換を回避する標準的な Tcl 構文で、これにより loc プロパティに空の文字列を渡すことができます。

オブジェクトのリストの処理

`get_cells` や `get_sites` などの複数のオブジェクトを返すコマンドは、通常ネイティブ Tcl リストのように機能するコレクションを返します。この機能により、多数の Tcl オブジェクト処理する場合に `foreach_in_collection` コマンドのような特殊なコマンドを必要としないので、パフォーマンスが向上します。Vivado Design Suite 内では、`lsort`、`lsearch`、`foreach` などのビルトイン コマンドを使用して、コレクションを Tcl リストと同様に処理できます。

通常、`get_*` コマンドを実行すると、その結果はコンソールおよびログ ファイルにリストではなく Tcl 文字列として出力されます。内部的には、Tcl では変数または値を文字列、および浮動小数点オブジェクトやリスト オブジェクトのような高速ネイティブ オブジェクトの両方として保存できます。オブジェクトまたは値の表現は、必要に応じてリスト オブジェクトから文字列オブジェクト、または文字列からリストに切り替わります。`get_*` コマンドでは Vivado オブジェクトのリストが返されますが、ログ ファイルおよび Tcl コンソールにはその文字列表現が表示されます。

パフォーマンスを向上してメモリ バッファへの負荷を軽減するため、Vivado Design Suite では表示される文字列は `tcl.collectionResultDisplayLimit` パラメーターで指定されているデフォルトの長さで切り詰められます。`get_cells` や `get_sites` のように多数のオブジェを返す Tcl コマンドを実行した場合は、切り詰められた文字列の最後に省略記号 (...) が付きます。`tcl.collectionResultDisplayLimit` パラメーターの値を変更するには、`set_param` コマンドを使用します。



注意: Vivado Design Suite では、この 2 つの表現の切り替えと `tcl.collectionResultDisplayLimit` パラメーターにより、`in` および `ni` リスト演算子は使用されません。リストから切り替えられた変換された文字列は切り詰められていることがあるので、`in` および `ni` 演算子で指定のオブジェクトが `in` または `not-in` であるのか、オブジェクトのリストであるのかを効果的に判断できません。`in` または `ni` の代わりに `lsearch`、`lsort` などのリスト コマンドを使用してください。

```
if {[lsearch -exact [get_cells *] $cellName] != -1} {...}
```

`get_*` コマンドで返される完全なリストを取得するには、結果を Tcl 変数に割り当てます。

```
set allSites [get_sites]
```

変数に割り当てられるリストには完全な結果が含まれ、`tcl.collectionResultDisplayLimit` パラメーターの値で切り詰められていません。次に、デザインのすべての階層に含まれるすべてのセルをクエリする例を示します。

```
%set allCells [get_cells -hierarchical]
DataIn_pad_0_i_IBUF[0]_inst DataIn_pad_0_i_IBUF[1]_inst \
DataIn_pad_0_i_IBUF[2]_inst DataIn_pad_0_i_IBUF[3]_inst \
DataIn_pad_0_i_IBUF[4]_inst ...
%llength $allCells
42244
%lindex $allCells end
wbArbEngine/s4/next_reg
```

この例では、`get_cells -hierarchical` コマンドの結果を `$allCells` 変数に割り当てています。表示されている結果は省略されていますが、リストの長さを調べると 4 万個以上のセル オブジェがあることがわかり、リストの最後のインデックスは省略記号ではなく実際のオブジェクトであることがわかります。



ヒント: 必要に応じて `join` コマンドを使用し、`get_*` Tcl コマンドのリストを改行 (\n)、タブ (\t)、またはスペース (" ") で結合し、省略されていないオブジェクトのリストを表示することも可能です。

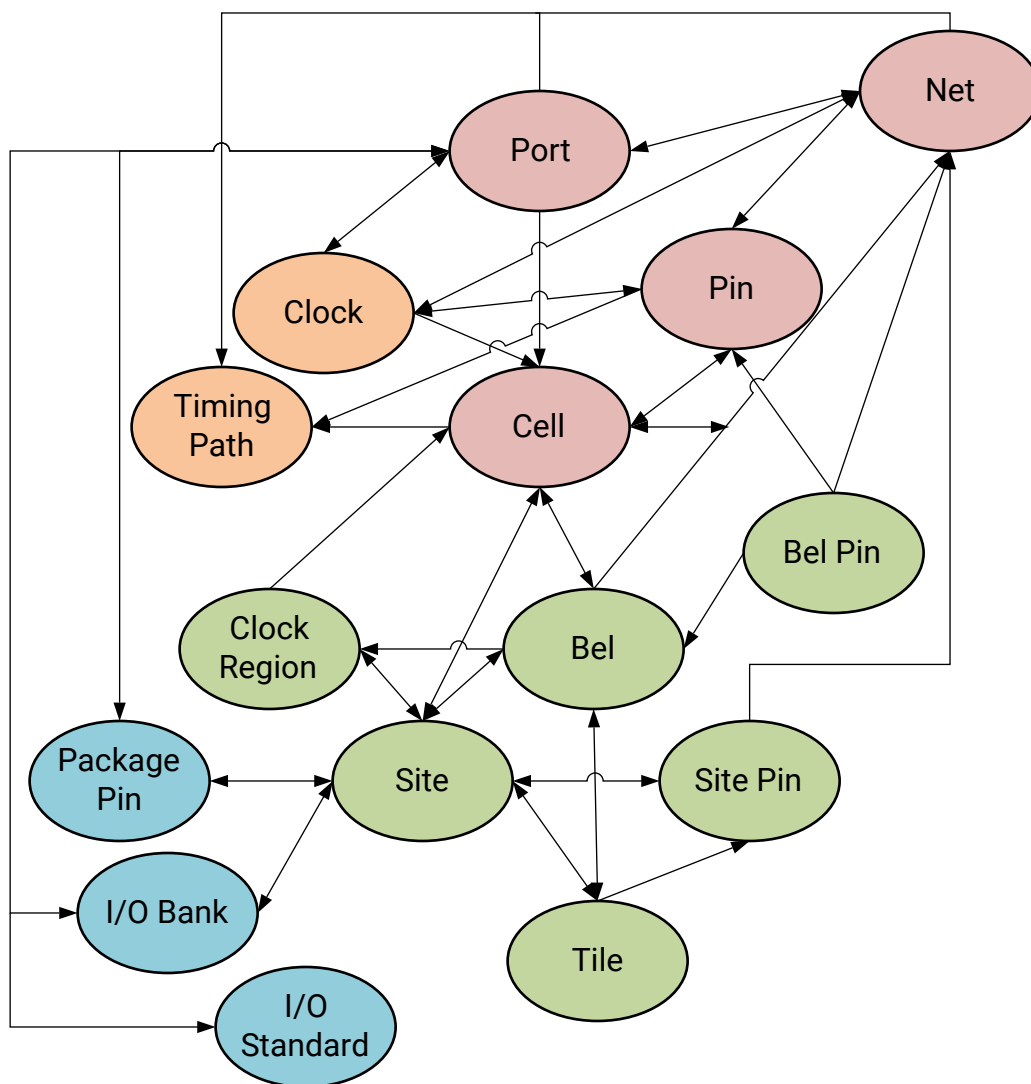
```
join [get_parts] " "
```

オブジェクトの関係

関連するオブジェクトは、`get_*` コマンドに `-of` オプションを使用してクエリできます。たとえば、あるセル オブジェクトに接続されたピンのリストを取得するには、次のように入力します。

- `get_pins -of [get_cells inst_1]`

次の図に、オブジェクト タイプとその関係を示します。オブジェクト間の矢印は、`get_*` コマンドで `-of` オプションを使用して、論理接続をたどって接続されているオブジェクトへの Tcl リファレンスを取得できることを示しています。ファースト クラス オブジェクトおよびその関係の詳細は、『Vivado Design Suite Tcl プロパティ リファレンス ガイド』(UG912)を参照してください。



エラー、警告、クリティカル警告、および情報メッセージ

各コマンドの結果を示すメッセージは、ログ ファイルと、GUI を使用中であればコンソールにも表示されます。これらのメッセージには識別しやすいように番号が付いています。ログ ファイルでは、INFO、WARNING、CRITICAL_WARNING、ERROR の後にサブシステム識別子と独自の番号が付きます。

次は、タイミング ライブラリを読み込んだ後に表示される INFO メッセージの例です。

```
INFO: [HD-LIB 1] Done reading timing library
```

このようなメッセージにより、ログ ファイルで特定の問題を検出しやすくなり、コマンド実行中の動作内容が理解しやすくなります。

通常、Tcl スクリプトからの Tcl コマンドでエラーが発生すると、続くコマンドの実行は停止されます。これは、回復不可能なエラー状況を避けるためです。これらのエラー状況を回避して続行させるための Tcl ビルトイン コマンドがあります。一般的な Tcl メカニズムを使用したエラー処理方法については、Tcl リファレンスで `catch` コマンドを参照してください。

Tcl コマンド

Tcl コマンド カテゴリ

Board (ボード)	Configuration (コンフィギュレーション)	CreatePeripheral (ペリフェラルの作成)
Debug (デバッグ)	DRC	Feasibility (実現可能性)
FileIO (ファイル入力および出力)	Floorplan (フロアプラン)	GUIControl (GUI 制御)
Hardware (ハードウェア)	IPFlow (IP フロー)	IPIntegrator (IP インテグレーター)
Memory (メモリ)	Methodology (設計手法)	Netlist (ネットリスト)
Object (オブジェクト)	Partition (パーティション)	PinPlanning (ピン プランニング)
Platform (プラットフォーム)	Power (電力)	Project (プロジェクト)
projutils (プロジェクト ユーティリティ)	PropertyAndParameter (プロパティおよびパラメーター)	Report (レポート)
SDC	Simulation (シミュレーション)	SysGen (System Generator)
Timing (タイミング)	ToolLaunch (ツール起動)	Tools (ツール)
Vitis	Waiver (除外)	Waveform (波形)
XDC	xilinxctlstore (ザイリンクス Tcl Store)	

Tcl コマンド リスト (カテゴリ別)

Board (ボード):

apply_board_connection	current_board	current_board_part
get_board_bus_nets	get_board_buses	get_board_component_interfaces
get_board_component_modes	get_board_component_pins	get_board_components
get_board_interface_ports	get_board_ip_preferences	get_board_jumpers
get_board_parameters	get_board_part_interfaces	get_board_part_pins
get_board_parts	get_boards	

Configuration (コンフィギュレーション):

[config_implementation](#)

CreatePeripheral (ペリフェラルの作成):

<code>add_peripheral_interface</code>	<code>create_peripheral</code>	<code>generate_peripheral</code>
<code>write_peripheral</code>		

Debug (デバッグ):

<code>apply_hw_ila_trigger</code>	<code>connect_debug_cores</code>	<code>connect_debug_port</code>
<code>create_debug_core</code>	<code>create_debug_port</code>	<code>delete_debug_core</code>
<code>delete_debug_port</code>	<code>disconnect_debug_port</code>	<code>get_debug_cores</code>
<code>get_debug_ports</code>	<code>implement_debug_core</code>	<code>modify_debug_ports</code>
<code>report_debug_core</code>	<code>write_debug_probes</code>	

DRC:

<code>add_drc_checks</code>	<code>create_drc_check</code>	<code>create_drc_ruledeck</code>
<code>create_drc_violation</code>	<code>create_waiver</code>	<code>delete_drc_check</code>
<code>delete_drc_ruledeck</code>	<code>get_drc_checks</code>	<code>get_drc_ruledecks</code>
<code>get_drc_violations</code>	<code>remove_drc_checks</code>	<code>report_drc</code>
<code>reset_drc</code>	<code>reset_drc_check</code>	

Feasibility (実現可能性):

<code>delete_qor_suggestions</code>	<code>get_qor_suggestions</code>	<code>read_qor_suggestions</code>
<code>report_qor_assessment</code>	<code>report_qor_suggestions</code>	<code>write_qor_suggestions</code>

FileIO (ファイル入力および出力):

<code>auto_detect_xpm</code>	<code>config_webtalk</code>	<code>create_port_on_reconfigurable_module</code>
<code>decrypt_bitstream</code>	<code>encrypt</code>	<code>generate_base_platform</code>
<code>generate_mem_files</code>	<code>generate_pblock</code>	<code>generate_rl_platform</code>
<code>generate_shx_platform</code>	<code>infer_diff_pairs</code>	<code>open_hw_platform</code>
<code>pr_recombine</code>	<code>pr_subdivide</code>	<code>pr_verify</code>
<code>read_bd</code>	<code>read_checkpoint</code>	<code>read_csv</code>
<code>read_edif</code>	<code>read_ip</code>	<code>read_mem</code>
<code>read_qor_suggestions</code>	<code>read_saif</code>	<code>read_schematic</code>
<code>read_twx</code>	<code>read_verilog</code>	<code>read_vhdl</code>
<code>read_xdc</code>	<code>refresh_meminit</code>	<code>write_abstract_shell</code>
<code>write_bd_layout</code>	<code>write_bitstream</code>	<code>write_bmm</code>
<code>write_bsd1</code>	<code>write_cfgmem</code>	<code>write_checkpoint</code>
<code>write_csv</code>	<code>write_debug_probes</code>	<code>write_edif</code>
<code>write_hw_platform</code>	<code>write_hw_platform_metadata</code>	<code>write_ibis</code>
<code>write_inferred_xdc</code>	<code>write_mem_info</code>	<code>write_qor_suggestions</code>
<code>write_schematic</code>	<code>write_sdf</code>	<code>write_verilog</code>
<code>write_vhdl</code>	<code>write_xdc</code>	

Floorplan (フロアプラン):

<code>add_cells_to_pblock</code>	<code>create_pblock</code>	<code>delete_pblocks</code>
<code>delete_rpm</code>	<code>get_pblocks</code>	<code>place_cell</code>
<code>place_pblocks</code>	<code>remove_cells_from_pblock</code>	<code>resize_pblock</code>
<code>swap_locs</code>	<code>unplace_cell</code>	

GUIControl (GUI 制御):

<code>create_gui_custom_command</code>	<code>create_gui_custom_command_arg</code>	<code>endgroup</code>
<code>get_gui_custom_command_args</code>	<code>get_gui_custom_commands</code>	<code>get_highlighted_objects</code>
<code>get_marked_objects</code>	<code>get_selected_objects</code>	<code>highlight_objects</code>
<code>mark_objects</code>	<code>redo</code>	<code>remove_gui_custom_command_args</code>
<code>remove_gui_custom_commands</code>	<code>select_objects</code>	<code>show_objects</code>
<code>show_schematic</code>	<code>start_gui</code>	<code>startgroup</code>
<code>stop_gui</code>	<code>undo</code>	<code>unhighlight_objects</code>
<code>unmark_objects</code>	<code>unselect_objects</code>	

Hardware (ハードウェア):

<code>add_hw_hbm_pc</code>	<code>add_hw_probe_enum</code>	<code>boot_hw_device</code>
<code>close_hw_manager</code>	<code>close_hw_target</code>	<code>commit_hw_hbm</code>
<code>commit_hw_mig</code>	<code>commit_hw_sio</code>	<code>commit_hw_sysmon</code>
<code>commit_hw_vio</code>	<code>config_hw_sio_gts</code>	<code>connect_hw_server</code>
<code>create_hw_axi_txn</code>	<code>create_hw_bitstream</code>	<code>create_hw_cfgmem</code>
<code>create_hw_device</code>	<code>create_hw_probe</code>	<code>create_hw_sio_link</code>
<code>create_hw_sio_linkgroup</code>	<code>create_hw_sio_scan</code>	<code>create_hw_sio_sweep</code>
<code>create_hw_target</code>	<code>current_hw_cfgmem</code>	<code>current_hw_device</code>
<code>current_hw_ila</code>	<code>current_hw_ila_data</code>	<code>current_hw_server</code>
<code>current_hw_target</code>	<code>delete_hw_axi_txn</code>	<code>delete_hw_bitstream</code>
<code>delete_hw_cfgmem</code>	<code>delete_hw_probe</code>	<code>delete_hw_target</code>
<code>detect_hw_sio_links</code>	<code>disconnect_hw_server</code>	<code>display_hw_ila_data</code>
<code>display_hw_sio_scan</code>	<code>execute_hw_svf</code>	<code>get_cfgmem_parts</code>
<code>get_hw_axi_txns</code>	<code>get_hw_axis</code>	<code>get_hw_cfgmems</code>
<code>get_hw_ddrmcs</code>	<code>get_hw_devices</code>	<code>get_hw_hbms</code>
<code>get_hw_ila_datas</code>	<code>get_hw_ilas</code>	<code>get_hw_migs</code>
<code>get_hw_probes</code>	<code>get_hw_servers</code>	<code>get_hw_sio_commons</code>
<code>get_hw_sio_gtgroups</code>	<code>get_hw_sio_gts</code>	<code>get_hw_sio_iberts</code>
<code>get_hw_sio_linkgroups</code>	<code>get_hw_sio_links</code>	<code>get_hw_sio_plls</code>
<code>get_hw_sio_rxs</code>	<code>get_hw_sio_scans</code>	<code>get_hw_sio_sweeps</code>
<code>get_hw_sio_txs</code>	<code>get_hw_sysmon_reg</code>	<code>get_hw_sysmons</code>
<code>get_hw_targets</code>	<code>get_hw_vios</code>	<code>list_hw_samples</code>
<code>open_hw_manager</code>	<code>open_hw_target</code>	<code>pause_hw_hbm_amon</code>
<code>program_hw_cfgmem</code>	<code>program_hw_devices</code>	<code>read_hw_ila_data</code>
<code>read_hw_sio_scan</code>	<code>read_hw_sio_sweep</code>	<code>readback_hw_cfgmem</code>
<code>readback_hw_device</code>	<code>refresh_hw_axi</code>	<code>refresh_hw_ddrmc</code>

refresh_hw_device	refresh_hw_hbm	refresh_hw_mig
refresh_hw_server	refresh_hw_sio	refresh_hw_sysmon
refresh_hw_target	refresh_hw_vio	remove_hw_hbm_pc
remove_hw_probe_enum	remove_hw_sio_link	remove_hw_sio_linkgroup
remove_hw_sio_scan	remove_hw_sio_sweep	report_hw_axi_txn
report_hw_ddrmc	report_hw_mig	report_hw_targets
reset_hw_axi	reset_hw_ila	reset_hw_vio_activity
reset_hw_vio_outputs	resume_hw_hbm_amon	run_hw_axi
run_hw_hbm_amon	run_hw_ila	run_hw_sio_scan
run_hw_sio_sweep	run_state_hw_jtag	runtest_hw_jtag
scan_dr_hw_jtag	scan_ir_hw_jtag	set_hw_sysmon_reg
stop_hw_hbm_amon	stop_hw_sio_scan	stop_hw_sio_sweep
update_hw_firmware	update_hw_gpio	upload_hw_ila_data
verify_hw_devices	wait_on_hw_ila	wait_on_hw_sio_scan
wait_on_hw_sio_sweep	write_hw_ila_data	write_hw_sio_scan
write_hw_sio_sweep	write_hw_svf	

IPFlow (IP フロー):

add_peripheral_interface	compile_c	config_ip_cache
convert_ips	copy_ip	create_ip
create_ip_run	create_peripheral	delete_ip_run
extract_files	generate_peripheral	generate_target
get_ip_upgrade_results	get_ipdefs	get_ips
import_ip	open_example_project	read_ip
report_ip_status	reset_target	synth_ip
update_ip_catalog	update_module_reference	upgrade_ip
validate_ip	write_ip_tcl	write_peripheral

IPIntegrator (IP インテグレーター):

apply_bd_automation	apply_board_connection	assign_bd_address
close_bd_design	compile_c	connect_bd_intf_net
connect_bd_net	copy_bd_objs	create_bd_addr_seg
create_bd_cell	create_bd_design	create_bd_intf_net
create_bd_intf_pin	create_bd_intf_port	create_bd_intf_tlm_port
create_bd_net	create_bd_pin	create_bd_port
create_bd_tlm_port	current_bd_design	current_bd_instance
delete_bd_objs	disconnect_bd_intf_net	disconnect_bd_net
exclude_bd_addr_seg	export_as_example_design	find_bd_objs
generate_target	get_bd_addr_segs	get_bd_addr_spaces
get_bd_cells	get_bd_designs	get_bd_intf_nets
get_bd_intf_pins	get_bd_intf_ports	get_bd_nets
get_bd_pins	get_bd_ports	get_bd_regs
get_example_designs	get_template_bd_designs	group_bd_cells

[include_bd_addr_seg](#)
[make_bd_intf_pins_external](#)
[open_bd_design](#)
[replace_bd_cell](#)
[save_bd_design_as](#)
[validate_bd_design](#)

[instantiate_example_design](#)
[make_bd_pins_external](#)
[read_bd](#)
[report_bd_diffs](#)
[ungroup_bd_cells](#)
[write_bd_tcl](#)

[instantiate_template_bd_design](#)
[move_bd_cells](#)
[regenerate_bd_layout](#)
[save_bd_design](#)
[upgrade_bd_cells](#)

Memory (メモリ):

[implement_mig_cores](#)

[implement_xphy_cores](#)

[refresh_meminit](#)

Methodology (設計手法):

[create_waiver](#)
[report_methodology](#)

[get_methodology_checks](#)
[reset_methodology](#)

[get_methodology_violations](#)
[reset_methodology_check](#)

Netlist (ネットリスト):

[connect_net](#)
[create_pin](#)
[remove_cell](#)
[rename_cell](#)
[rename_port](#)
[resize_pin_bus](#)

[create_cell](#)
[disconnect_net](#)
[remove_net](#)
[rename_net](#)
[rename_ref](#)
[tie_unused_pins](#)

[create_net](#)
[get_net_delays](#)
[remove_pin](#)
[rename_pin](#)
[resize_net_bus](#)

Object (オブジェクト):

[add_drc_checks](#)
[config_ip_cache](#)
[create_partition_def](#)
[create_report_config](#)
[current_board_part](#)
[delete_drc_ruledck](#)
[delete_report_configs](#)
[find_routing_path](#)
[get_bels](#)
[get_board_component_interfaces](#)
[get_board_components](#)
[get_board_jumpers](#)
[get_board_part_pins](#)
[get_cdc_violations](#)
[get_clock_regions](#)
[get_debug_cores](#)
[get_drc_checks](#)
[get_files](#)
[get_highlighted_objects](#)

[apply_board_connection](#)
[create_drc_check](#)
[create_pr_configuration](#)
[create_waiver](#)
[current_pr_configuration](#)
[delete_hw_bitstream](#)
[delete_waivers](#)
[generate_reports](#)
[get_board_bus_nets](#)
[get_board_component_modes](#)
[get_board_interface_ports](#)
[get_board_parameters](#)
[get_board_parts](#)
[get_cells](#)
[get_clocks](#)
[get_debug_ports](#)
[get_drc_ruledcks](#)
[get_filesets](#)
[get_hw_axi_txns](#)

[can_resolve_reference](#)
[create_drc_ruledck](#)
[create_reconfig_module](#)
[current_board](#)
[delete_drc_check](#)
[delete_qor_suggestions](#)
[filter](#)
[get_bel_pins](#)
[get_board_buses](#)
[get_board_component_pins](#)
[get_board_ip_preferences](#)
[get_board_part_interfaces](#)
[get_boards](#)
[get_cfgmem_parts](#)
[get_dashboard_gadgets](#)
[get_designs](#)
[get_drc_violations](#)
[get_generated_clocks](#)
[get_hw_axis](#)

<code>get_hw_cfgmems</code>	<code>get_hw_ddrmcs</code>	<code>get_hw_devices</code>
<code>get_hw_hbms</code>	<code>get_hw_ila_datas</code>	<code>get_hw_ilas</code>
<code>get_hw_migs</code>	<code>get_hw_probes</code>	<code>get_hw_servers</code>
<code>get_hw_sio_commons</code>	<code>get_hw_sio_gtgroups</code>	<code>get_hw_sio_gts</code>
<code>get_hw_sio_iberts</code>	<code>get_hw_sio_linkgroups</code>	<code>get_hw_sio_links</code>
<code>get_hw_sio_plls</code>	<code>get_hw_sio_rxs</code>	<code>get_hw_sio_scans</code>
<code>get_hw_sio_sweeps</code>	<code>get_hw_sio_txs</code>	<code>get_hw_sysmons</code>
<code>get_hw_targets</code>	<code>get_hw_vios</code>	<code>get_interfaces</code>
<code>get_io_standards</code>	<code>get_iobanks</code>	<code>get_ip_upgrade_results</code>
<code>get_ipdefs</code>	<code>get_ips</code>	<code>get_lib_cells</code>
<code>get_lib_pins</code>	<code>get_libs</code>	<code>get_macros</code>
<code>get_marked_objects</code>	<code>get_methodology_checks</code>	<code>get_methodology_violations</code>
<code>get_net_delays</code>	<code>get_nets</code>	<code>get_nodes</code>
<code>get_package_pins</code>	<code>get_partition_defs</code>	<code>get_parts</code>
<code>get_path_groups</code>	<code>get_pblocks</code>	<code>get_pins</code>
<code>get_pips</code>	<code>get_pkgpin_bytegroups</code>	<code>get_pkgpin_nibbles</code>
<code>get_ports</code>	<code>get_pr_configurations</code>	<code>get_primitives</code>
<code>get_projects</code>	<code>get_property</code>	<code>get_qor_suggestions</code>
<code>get_reconfig_modules</code>	<code>get_report_configs</code>	<code>get_runs</code>
<code>get_selected_objects</code>	<code>get_site_pins</code>	<code>get_site_pips</code>
<code>get_sites</code>	<code>get_slrs</code>	<code>get_speed_models</code>
<code>get_tiles</code>	<code>get_timing_arcs</code>	<code>get_timing_paths</code>
<code>get_waivers</code>	<code>get_wires</code>	<code>list_hw_samples</code>
<code>list_property</code>	<code>list_property_value</code>	<code>remove_drc_checks</code>
<code>report_property</code>	<code>report_qor_suggestions</code>	<code>report_waivers</code>
<code>reset_drc_check</code>	<code>reset_methodology_check</code>	<code>reset_property</code>
<code>run_state_hw_jtag</code>	<code>runtest_hw_jtag</code>	<code>scan_dr_hw_jtag</code>
<code>scan_ir_hw_jtag</code>	<code>set_property</code>	<code>write_ip_tcl</code>
<code>write_waivers</code>		

Partition (パーティション):

<code>create_partition_def</code>	<code>create_pr_configuration</code>	<code>create_reconfig_module</code>
<code>current_pr_configuration</code>	<code>delete_partition_defs</code>	<code>delete_pr_configurations</code>
<code>delete_reconfig_modules</code>	<code>get_partition_defs</code>	<code>get_pr_configurations</code>
<code>get_reconfig_modules</code>	<code>setup_pr_configurations</code>	

PinPlanning (ピン プランニング):

<code>create_interface</code>	<code>create_port</code>	<code>delete_interface</code>
<code>make_diff_pair_ports</code>	<code>place_ports</code>	<code>remove_port</code>
<code>resize_port_bus</code>	<code>set_package_pin_val</code>	<code>split_diff_pair_ports</code>

Platform (プラットフォーム):

<code>open_hw_platform</code>	<code>validate_hw_platform</code>	<code>write_hw_platform</code>
<code>write_hw_platform_metadata</code>		

Power (電力):

<code>delete_power_results</code>	<code>power_opt_design</code>	<code>read_saif</code>
<code>report_power</code>	<code>report_power_opt</code>	<code>reset_operating_conditions</code>
<code>reset_switching_activity</code>	<code>set_operating_conditions</code>	<code>set_power_opt</code>
<code>set_switching_activity</code>		

Project (プロジェクト):

<code>add_files</code>	<code>add_peripheral_interface</code>	<code>apply_board_connection</code>
<code>archive_project</code>	<code>auto_detect_xpm</code>	<code>can_resolve_reference</code>
<code>check_syntax</code>	<code>close_design</code>	<code>close_project</code>
<code>compile_c</code>	<code>copy_ip</code>	<code>create_dashboard_gadget</code>
<code>create_fileset</code>	<code>create_ip_run</code>	<code>create_peripheral</code>
<code>create_project</code>	<code>create_run</code>	<code>create_xps</code>
<code>current_board_part</code>	<code>current_fileset</code>	<code>current_project</code>
<code>current_run</code>	<code>delete_dashboard_gadgets</code>	<code>delete_fileset</code>
<code>delete_ip_run</code>	<code>delete_runs</code>	<code>find_top</code>
<code>generate_peripheral</code>	<code>generate_target</code>	<code>get_board_parts</code>
<code>get_boards</code>	<code>get_dashboard_gadgets</code>	<code>get_files</code>
<code>get_filesets</code>	<code>get_ip_upgrade_results</code>	<code>get_ips</code>
<code>get_projects</code>	<code>get_runs</code>	<code>help</code>
<code>import_files</code>	<code>import_ip</code>	<code>import_synplify</code>
<code>import_xise</code>	<code>import_xst</code>	<code>launch_runs</code>
<code>list_targets</code>	<code>lock_design</code>	<code>make_wrapper</code>
<code>move_dashboard_gadget</code>	<code>move_files</code>	<code>open_checkpoint</code>
<code>open_example_project</code>	<code>open_io_design</code>	<code>open_project</code>
<code>open_run</code>	<code>refresh_design</code>	<code>refresh_meminit</code>
<code>reimport_files</code>	<code>remove_files</code>	<code>reorder_files</code>
<code>report_compile_order</code>	<code>reset_project</code>	<code>reset_runs</code>
<code>reset_target</code>	<code>save_constraints</code>	<code>save_constraints_as</code>
<code>save_project_as</code>	<code>set_part</code>	<code>set_speed_grade</code>
<code>synth_ip</code>	<code>update_compile_order</code>	<code>update_design</code>
<code>update_files</code>	<code>update_sw_parameters</code>	<code>wait_on_run</code>
<code>write_hwdef</code>	<code>write_ip_tcl</code>	<code>write_peripheral</code>

projutils (プロジェクト ユーティリティ):

<code>convert_ngc</code>	<code>copy_run</code>	<code>create_rqs_run</code>
<code>export_bd_synth</code>	<code>write_project_tcl</code>	

PropertyAndParameter (プロパティおよびパラメーター):

<code>create_property</code>	<code>filter</code>	<code>get_param</code>
<code>get_property</code>	<code>list_param</code>	<code>list_property</code>
<code>list_property_value</code>	<code>report_param</code>	<code>report_property</code>
<code>reset_param</code>	<code>reset_property</code>	<code>set_param</code>
<code>set_part</code>	<code>set_property</code>	

Report (レポート):

<code>calc_config_time</code>	<code>check_timing</code>	<code>create_drc_violation</code>
<code>create_report_config</code>	<code>create_slack_histogram</code>	<code>delete_clock_networks_results</code>
<code>delete_report_configs</code>	<code>delete_timing_results</code>	<code>delete_utilization_results</code>
<code>generate_reports</code>	<code>get_msg_config</code>	<code>get_pplocs</code>
<code>get_report_configs</code>	<code>open_report</code>	<code>report_bus_skew</code>
<code>report_carry_chains</code>	<code>report_cdc</code>	<code>report_clock_interaction</code>
<code>report_clock_networks</code>	<code>report_clock_utilization</code>	<code>report_clocks</code>
<code>report_config_implementation</code>	<code>report_config_timing</code>	<code>report_control_sets</code>
<code>report_datasheet</code>	<code>report_debug_core</code>	<code>report_design_analysis</code>
<code>report_disable_timing</code>	<code>report_drc</code>	<code>report_environment</code>
<code>report_exceptions</code>	<code>report_high_fanout_nets</code>	<code>report_hw_ddrmc</code>
<code>report_hw_mig</code>	<code>report_incremental_reuse</code>	<code>report_io</code>
<code>report_methodology</code>	<code>report_operating_conditions</code>	<code>report_param</code>
<code>report_phys_opt</code>	<code>report_power</code>	<code>report_pr_configuration_analysis</code>
<code>report_property</code>	<code>report_pulse_width</code>	<code>report_qor_assessment</code>
<code>report_qor_suggestions</code>	<code>report_ram_utilization</code>	<code>report_route_status</code>
<code>report_sim_device</code>	<code>report_ssn</code>	<code>report_switching_activity</code>
<code>report_synchronizer_mtbfs</code>	<code>report_timing</code>	<code>report_timing_summary</code>
<code>report_transformed_primitives</code>	<code>report_utilization</code>	<code>report_waivers</code>
<code>reset_drc</code>	<code>reset_methodology</code>	<code>reset_msg_config</code>
<code>reset_msg_count</code>	<code>reset_ssn</code>	<code>reset_timing</code>
<code>set_msg_config</code>	<code>version</code>	

SDC:

<code>all_clocks</code>	<code>all_inputs</code>	<code>all_outputs</code>
<code>all_registers</code>	<code>create_clock</code>	<code>create_generated_clock</code>
<code>current_design</code>	<code>current_instance</code>	<code>get_cells</code>
<code>get_clocks</code>	<code>get_hierarchy_separator</code>	<code>get_nets</code>
<code>get_pins</code>	<code>get_ports</code>	<code>group_path</code>
<code>set_case_analysis</code>	<code>set_clock_groups</code>	<code>set_clock_latency</code>
<code>set_clock_sense</code>	<code>set_clock_uncertainty</code>	<code>set_data_check</code>
<code>set_disable_timing</code>	<code>set_false_path</code>	<code>set_hierarchy_separator</code>
<code>set_input_delay</code>	<code>set_load</code>	<code>set_logic_dc</code>
<code>set_logic_one</code>	<code>set_logic_zero</code>	<code>set_max_delay</code>
<code>set_max_time_borrow</code>	<code>set_min_delay</code>	<code>set_multicycle_path</code>

set_operating_conditions
set_units

set_output_delay

set_propagated_clock

Simulation (シミュレーション):

add_bp
add_force
close_sim
config_compile_simlib
current_scope
current_vcd
export_ip_user_files
generate_mem_files
get_simulators
import_files
log_saif
ltrace
open_vcd
read_saif
remove_conditions
report_bps
report_frames
report_simlib_info
reset_simulation
set_value
step
write_sdf
xsim

add_condition
checkpoint_vcd
close_vcd
create_fileset
current_sim
delete_fileset
export_simulation
get_objects
get_stacks
launch_simulation
log_vcd
move_files
open_wave_database
relaunch_sim
remove_files
report_conditions
report_objects
report_stacks
restart
setup_ip_static_library
stop
write_verilog

add_files
close_saif
compile_simlib
current_frame
current_time
describe
flush_vcd
get_scopes
get_value
limit_vcd
log_wave
open_saif
ptrace
remove_bps
remove_forces
report_drivers
report_scopes
report_values
run
start_vcd
stop_vcd
write_vhdl

SysGen (System Generator):

create_sysgen

make_wrapper

Timing (タイミング):

check_timing
config_timing_corners
delete_timing_results
get_timing_arcs
report_bus_skew
report_clock_networks
report_config_timing
report_disable_timing
report_high_fanout_nets
report_qor_assessment
report_timing

config_design_analysis
create_slack_histogram
get_net_delays
get_timing_paths
report_cdc
report_clock_utilization
report_datasheet
report_drc
report_methodology
report_qor_suggestions
report_timing_summary

config_timing_analysis
delete_qor_suggestions
get_qor_suggestions
read_qor_suggestions
report_clock_interaction
report_clocks
report_design_analysis
report_exceptions
report_pulse_width
report_synchronizer_mtbfb
reset_timing

set_delay_model
update_timing
write_sdf

set_disable_timing
write_inferred_xdc
write_xdc

set_external_delay
write_qor_suggestions

ToolLaunch (ツール起動):

get_simulators
launch_simulation

launch_chipscope_analyzer

launch_impact

Tools (ツール):

iphys_opt_design
load_features
place_design
report_pipeline_analysis
unregister_proc
write_iphys_opt_tcl

link_design
opt_design
read_iphys_opt_tcl
route_design
update_clock_routing

list_features
phys_opt_design
register_proc
synth_design
update_noc_qos

Vitis:

open_hw_platform
write_hw_platform_metadata

validate_hw_platform

write_hw_platform

Waiver (除外):

create_waiver
report_waivers

delete_waivers
write_waivers

get_waivers

Waveform (波形):

add_wave
add_wave_marker
create_wave_config
get_waves
remove_wave

add_wave_divider
add_wave_virtual_bus
current_wave_config
move_wave
save_wave_config

add_wave_group
close_wave_config
get_wave_configs
open_wave_config
select_wave_objects

XDC:

add_cells_to_pblock
all_dsps
all_ffs
all_latches
all_registers
create_clock
create_generated_clock
create_property
current_instance

all_clocks
all_fanin
all_hsios
all_outputs
connect_debug_cores
create_debug_core
create_macro
create_waiver
delete_macros

all_cpus
all_fanout
all_inputs
all_rams
connect_debug_port
create_debug_port
create_pblock
current_design
delete_pblocks

filter	get_bel_pins	get_bels
get_cells	get_clocks	get_debug_cores
get_debug_ports	get_generated_clocks	get_hierarchy_separator
get_iobanks	get_macros	get_nets
get_nodes	get_package_pins	get_path_groups
get_pblocks	get_pins	get_pips
get_pkgpin_bytegroups	get_pkgpin_nibbles	get_ports
get_property	get_site_pins	get_site_pips
get_sites	get_slrs	get_speed_models
get_tiles	get_timing_arcs	get_wires
group_path	make_diff_pair_ports	remove_cells_from_pblock
reset_operating_conditions	reset_switching_activity	resize_pblock
set_bus_skew	set_case_analysis	set_clock_groups
set_clock_latency	set_clock_sense	set_clock_uncertainty
set_data_check	set_disable_timing	set_external_delay
set_false_path	set_hierarchy_separator	set_input_delay
set_input_jitter	set_load	set_logic_dc
set_logic_one	set_logic_unconnected	set_logic_zero
set_max_delay	set_max_time_borrow	set_min_delay
set_multicycle_path	set_operating_conditions	set_output_delay
set_package_pin_val	set_power_opt	set_propagated_clock
set_property	set_switching_activity	set_system_jitter
set_units	update_macro	

xilinx_tclstore (ザイリンクス Tcl Store):

convert_ngc	copy_run	create_rqs_run
export_bd_synth	export_ip_user_files	export_simulation
setup_ip_static_library	write_project_tcl	

Tcl コマンド リスト (アルファベット順)

この章では、すべての SDC および Tcl コマンドをアルファベット順にリストします。

add_bp

HDL ソースの指定した行にブレークポイントを追加します。

構文

```
add_bp [-quiet] [-verbose] <file_name> <line_number>
```

戻り値

指定のファイルの行にブレークポイントが設定されていなかった場合は新しいブレークポイント オブジェクト、ブレークポイントが設定されていた場合は既存のブレークポイント オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file_name></code>	ブレークポイントを追加するファイルの名前を指定します。
<code><line_number></code>	指定のファイルのブレークポイントを設定する行番号を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

`add_bp` コマンドは、現在のシミュレーションを一時停止するブレークポイントを HDL ソース ファイルに追加します。

ブレークポイントは、ユーザーの指定するソース コード内の停止地点で、デザインをデバッグする際に使用できます。ブレークポイントを含むデザインをシミュレーションすると、各ブレークポイントでシミュレーションが停止し、値および動作を確認できます。

現在のシミュレーションのブレークポイントをレポートするには `report_bps` コマンドを使用し、既存のブレークポイントを削除するには `remove_bps` コマンドを使用します。

このコマンドを実行すると、指定のファイルの行にブレークポイントが設定されていなかった場合は新しいブレークポイント オブジェクト、ブレークポイントが既に設定されていた場合は既存のブレークポイント オブジェクトが返されます。



ヒント: 必要に応じて、返されたブレークポイント オブジェクトを Tcl 変数に保存できます。

`add_bp` コマンドが正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file_name>` (必須): ブレークポイントを追加する HDL ソース ファイルの名前を指定します。

`<line_number>` (必須): 指定した HDL ファイル (`<file_name>`) のブレークポイントを追加する行の番号を指定します。

例

次の例では、HDL ソース ファイルの指定の行にブレークポイントを追加します。

```
add_bp C:/Data/ug937/sources/sinegen.vhd 137
```

関連項目

- [remove_bps](#)
- [report_bps](#)

add_cells_to_pblock

Pblock にセルを追加します。

構文

```
add_cells_to_pblock [-top] [-add_primitives] [-clear_locs] [-quiet]
                    [-verbose] <pblock> [<cells>...]
```

使用法

名前	説明
[-top]	最上位インスタンスを追加します。cells 引数または -add_primitives オプションと共に使用することはできません。<cells> 引数または -top オプションのいずれかを指定する必要があります。
[-add_primitives]	指定したセルからプリミティブセルのみを Pblock に割り当てます。
[-clear_locs]	インスタンスのロケーション制約を削除します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pblock>	セルを追加する Pblock を指定します。
[<cells>]	追加するセルを指定します。-top オプションを使用している場合は指定できません。<cells> 引数または -top オプションのいずれかを指定する必要があります。

カテゴリ

[Floorplan \(フロアプラン\)](#)、[XDC](#)

説明

指定したロジック インスタンスを開いているインプリメント済みデザインの Pblock に追加します。Pblock にセルを追加したら、`resize_pblock` コマンドを使用して Pblock を FPGA のファブリック上に配置できます。

`resize_pblock` コマンドを使用すると、Pblock を手動で移動およびサイズ変更できます。

Pblock からインスタンスを削除するには、`remove_cells_from_pblock` コマンドを使用します。

引数

`-top` (オプション): 最上位インスタンスを追加して、デザイン全体の Pblock を作成します。Pblock にオブジェクトを追加するには、<cells> または `-top` オプションのいずれかを使用する必要があります。

`-add_primitives` (オプション): 指定したインスタンスのすべてのプリミティブを Pblock に割り当てます。このオプションを使用すると、階層モジュールのすべてのセルを選択し、選択したセルのうち最下位セルのみを指定の Pblock に追加できます。

注記: `-top` オプションと共に使用することはできません。

`-clear_locs` (オプション): 既に配置されているセルのインスタンス ロケーション制約を消去します。これにより、フロアプランするために新しい Pblock を定義する際に、セルの LOC 制約をリセットできます。このオプションを指定しないと、配置が指定されているインスタンスを Pblock に追加する場合に配置が解除されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<pblock>: Pblock の名前を指定します。

<cells>: 指定の Pblock に追加する 1 つまたは複数のセル オブジェクトを指定します。

注記: `-top` を指定した場合、<cells> を指定することはできません。

例

次の例では、`pb_cpuEngine` という Pblock を作成し、`cpuEngine` モジュールに含まれる最下位セルのみを追加し、配置済みのインスタンスの配置制約を消去します。

```
create_pblock pb_cpuEngine
add_cells_to_pblock pb_cpuEngine [get_cells cpuEngine/*] \
    -add_primitives -clear_locs
```

関連項目

- [get_pblocks](#)
- [place_pblocks](#)
- [remove_cells_from_pblock](#)
- [resize_pblock](#)

add_condition

Tcl コマンドを条件に基づいて実行します。

構文

```
add_condition [-name <arg>] [-radix <arg>] [-notrace] [-quiet] [-verbose]
               <condition_expression> <commands>
```

戻り値

作成された条件オブジェクト

使用法

名前	説明
[-name]	条件の名前 (ラベル) を指定します。複数の条件に同じ名前を付けることはできません。名前を指定しない場合、デフォルト名 (condition<id>) が自動的に作成されます。
[-radix]	使用する基数を指定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-notrace]	条件コマンドのログをオフにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<condition_expression>	条件式を指定します。条件式が満たされると、コマンドが実行されます。
<commands>	条件が満たされたときに実行するコマンドを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

条件式 <condition_expression> を指定し、条件が真の場合に一連のシミュレーション Tcl コマンドを実行します。

条件はシミュレーションの開始前に定義できます。条件を追加すると、シミュレータで信号の変化が検出されるたびに条件式が評価されます。指定の条件が真になると、条件コマンドが実行されます。

add_condition コマンドを実行すると、追加した条件の識別子が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-name <arg> (オプション): 条件の名前を指定します。名前を指定しない場合、デフォルト名が自動的に作成されます。

-radix <arg> (オプション): 条件の値に使用する基数を指定します。有効な値は default、dec、bin、oct、hex、unsigned、ascii、および smag (符号付き絶対値) です。

注記: `dec` は、符号付き 10 進数を示します。符号なしデータの場合は、`unsigned` を指定してください。

`-notrace` (オプション): 条件が真のときに実行される条件 `<commands>` のログをディスエーブルにします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<condition_expression>` (必須): 条件式を指定します。条件式が真の場合、シミュレーションで指定のコマンド (`<commands>`) が実行されます。条件式に使用できる演算子は、等価 (`==`) および不等価 (`!=`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を `AND (&&)` および `OR (||)` で組み合わせることもできます。

`<commands>` (必須): 条件式 (`<condition_expression>`) が真の場合に実行する Tcl コマンドまたは Tcl プロシージャを指定します。コマンドは波かっこ (`{}`) で囲みます。コマンドには、`run`、`restart`、`step` 以外の標準 Tcl コマンドおよびシミュレーション Tcl コマンドを含めることができます。条件式で使用されている Tcl 変数は、変数置換が実行されるように、波かっこ (`{}`) ではなくダブルクォーテーション (`"`) で囲みます。変数置換の詳細は、『Vivado Design Suite ユーザー ガイド: Tcl スクリプト機能の使用』 (UG894) を参照してください。

例

次の例では、`resetLow` という条件 (リセット信号が Low) を定義し、この条件が真になったときに、標準出力にメッセージを表示し、現在のシミュレーションを停止します。

```
add_condition -name resetLow {/testbench/reset == 0} {
  puts "Condition Reset was encountered at [current_time]. Stopping
simulation."
  stop }
```

次の例では、`add_force` コマンドを使用して `clk` および `reset` 信号の値を定義し、完了すると標準メッセージを表示する `myProc` という Tcl プロシージャを定義します。その後、リセットが Low のときにそのプロシージャ `myProc` を呼び出す条件を追加しています。

```
proc myProc {} {
  add_force clk {0 1} { 1 2} -repeat_every 4 -cancel_after 500
  add_force reset 1
  run 10 ns
  remove_force force2
  puts "Reached end of myProc"
}

add_condition -radix unsigned /top/reset==0 myproc
```

関連項目

- [add_force](#)
- [stop](#)

add_drc_checks

ルール デックに DRC ルール チェック オブジェクトを追加します。

構文

```
add_drc_checks [-of_objects <args>] [-regex] [-nocase] [-filter <arg>]
               -ruledock <arg> [-quiet] [-verbose] [<patterns>]
```

戻り値

drc_check オブジェクト

使用法

名前	説明
[-of_objects]	指定した drc_ruledock の rule_check オブジェクトを追加します。
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
-ruledock	変更する DRC ルール デックを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	rule_check オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

指定した DRC ルール デック (drc_ruledock) オブジェクトにデザイン ルール チェックを追加します。

ルール デックとは、デザイン ルール チェックのグループで、I/O プランニングなどのサイリンクス デザイン フローの異なる段階で report_drc コマンドにより実行されます。ツールには定義済みのルール デックが含まれていますが、create_drc_ruledock コマンドを使用して新しいユーザー定義のルール デックを作成できます。

get_drc_ruledocks コマンドを使用すると、report_drc コマンドで使用可能な定義済みのルール デックを取得できます。

ルール デックには、ツールに含まれる定義済みのルール チェックまたは create_drc_check コマンドで作成したユーザー定義のルール チェックを追加できます。get_drc_checks コマンドを使用すると、ルール デックに追加できるルール チェックを取得できます。

ルール デックからチェックを削除するには、remove_drc_checks コマンドを使用します。

注記: 特定の DRC ルールを一時的にディスエーブルにするには、`set_property` コマンドを使用してルールの `IS_ENABLED` プロパティを `false` に設定します。これにより、ルール デックからルールを削除せずに、`report_drc` を実行したときにルールを除外できます。ルールをデフォルト設定に戻すには、`reset_drc_check` コマンドを使用します。

このコマンドを実行すると、ルール デックに追加されているデザイン ルール チェックのリストが返されます。

引数

`-of_objects <arg>` (オプション): 指定した `drc_ruledeck` オブジェクトのルール チェックを追加します。このオプションを使用すると、ルールがあるルール デックから別のルール デックにコピーされます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、検索パターンにより返されるオブジェクトのリストに、指定したプロパティ値に基づくフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-ruledeck <arg>` (必須): 指定のデザイン ルール チェックを追加するルール デックの名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ルール デックに追加するデザイン ルール チェックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのルール チェックが指定のルール デックに追加されます。複数の検索パターンを指定して、異なる検索条件に基づいてルール チェックを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定した検索パターンに一致するルール チェックを `project_rules` ルール デックに追加しています。

```
add_drc_checks -ruledock project_rules {*DCI* *BUF*}
```

次の例では、`placer+` というルール デックを作成し、`placer_checks` ルール デックに含まれるすべてのルール チェックを `placer+` ルール デックにコピーして、さらにルール チェックを追加しています。

```
create_drc_ruledock placer+
add_drc_checks -of_objects [get_drc_ruledocks placer_checks] \
  -ruledock placer+
add_drc_checks -ruledock placer+ *IO*
```

次の例では、重要度が警告のルール チェックのみをルール デックに追加しています。

```
add_drc_checks -filter {SEVERITY == Warning} -ruledock warn_only
```

関連項目

- [create_drc_check](#)
- [create_drc_ruledock](#)
- [get_drc_checks](#)
- [get_drc_ruledocks](#)
- [list_property](#)
- [remove_drc_checks](#)
- [report_drc](#)
- [report_property](#)
- [reset_drc_check](#)
- [set_property](#)

add_files

アクティブなファイルセットにソースを追加します。

構文

```
add_files [-fileset <arg>] [-of_objects <args>] [-norecurse]
          [-copy_to <arg>] [-force] [-scan_for_includes] [-quiet] [-verbose]
          [<files>...]
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
[-fileset]	ファイルセット名を指定します。
[-of_objects]	ファイルを追加するファイルセット、サブデザイン、または RM を指定します。
[-norecurse]	下位ディレクトリでは検索を実行しないよう指定します。
[-copy_to]	ファイルをプロジェクトに追加する前に、指定のディレクトリにコピーします。
[-force]	-copy_to オプションを使用した場合に、既存のファイルを上書きします。
[-scan_for_includes]	ファイルセットの RTL ソースに含まれるファイルすべてをスキャンして追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<files>]	追加するファイルおよびディレクトリ名を指定します。-scan_for_includes を使用しない場合は指定する必要があります。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

1 つ以上のソース ファイルまたは 1 つ以上のディレクトリ内のソース ファイルを、現在のプロジェクトの指定したファイルセットに追加します。有効なソース ファイルには、HDL ソース (VHDL、Verilog、SystemVerilog、およびヘッダー ファイル)、ネットリスト ソース (DCP、EDIF、および NGC)、メモリ インターフェイス ファイル (BMM、MIF、MEM、ELF) などがあります。

IP およびブロック デザイン ソースを追加する際は、add_files コマンドは使用しません。IP およびブロック デザイン ソースは、import_ip、read_bd、read_ip などの個別のコマンドでサポートされます。

Vivado Design Suite では、プロジェクトに追加されたすべてのファイルに対して、そのファイルまたはディレクトリへの相対パスと絶対パスの両方が格納および管理されます。プロジェクトを開くと、これらのパスを使用してファイルおよびディレクトリが検索されます。デフォルトでは、パスを検索するのにまず相対パスが使用され、その後絶対パスが使用されます。PATH_MODE プロパティを使用すると、Vivado ツールでの指定したオブジェクトのファイルパスまたはプロパティの処理方法を変更できます。詳細は、『Vivado Design Suite プロパティ リファレンス ガイド』 (UG912) を参照してください。



重要: 複数のファイルを 1 つずつ追加すると、パフォーマンスが著しく低下します。1 つの `add_files` コマンドで複数のファイルをインポートする方が効率的です。

```
add_files {file1 file2 file3 ... fileN}
```

Vivado ツールでは、`add_files` コマンドでプロジェクトにファイルを追加したときにはファイルの内容は自動的に読み込まれず、必要なときに読み込まれます。たとえば、制約ファイルはデザインに追加されたときには読み込まれず、合成、タイミング、インプリメンテーションが実行されたときに読み込まれます。デザインにファイルを追加したときに読み込むには、`read_xxx` コマンドを使用してください。



ヒント: 非プロジェクト モードを使用しており、プロジェクトのソース ファイルを管理するプロジェクト ファイルがない場合は、`read_xxx` コマンドを使用してソース ファイル メモリ内のデザインに読み込む必要があります。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

`add_files` コマンドは、指定したファイルセットを参照することによってのみファイルを追加します。ファイルをローカルのプロジェクト フォルダにコピーし、指定したファイルセットにも追加する `import_files` コマンドとは異なります。

このコマンドを実行すると、追加されたファイルが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-fileset <name>` (オプション): 指定したソース ファイルを追加するファイルセットを指定します。指定したファイルセットが存在しない場合は、エラーが返されます。ファイルセットを指定しない場合は、デフォルトでソース ファイルセットに追加されます。

`-norecurse` (オプション): 指定したディレクトリの下位ディレクトリでコマンドを実行しないよう指定します。このオプションを指定しない場合、下位ディレクトリでもプロジェクトに追加可能なソース ファイルが検索されます。

`-copy_to <arg>` (オプション): 選択したファイルをプロジェクトに追加する前に、指定のディレクトリにコピーします。このオプションを使用すると、ファイルを現在の場所から新しいフォルダに移動し、プロジェクトのソース構造の一部として参照できます。

`-force` (オプション): `-copy_to` オプションを使用する際、指定のディレクトリにコピーするファイルと同じ名前のファイルがある場合に、既存のファイルを上書きします。

`-scan_for_includes` (オプション): Verilog ソース ファイルで `'include` 文を検索し、それらの参照ファイルも指定したファイルセットに追加します。デフォルトでは、`'include` ファイルは追加されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (オプション): ファイルセットに追加する 1 つまたは複数のファイルまたはディレクトリ名を指定します。ディレクトリ名を指定した場合は、そのディレクトリとそれに含まれる下位ディレクトリの有効なソース ファイルすべてがファイルセットに追加されます。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、`rtl.v` というファイルを現在のプロジェクトに追加しています。

```
add_files rtl.v
```

この例ではパスが指定されていないので、現在の作業ディレクトリで `rtl.v` ファイルが検索されます。ファイルセットは指定されていないので、ファイルはデフォルトでソース ファイルセットに追加されます。

次の例では、`top.xdc` というファイルを `constrs_1` 制約ファイルセットに追加し、`project_1` ディレクトリとその下位ディレクトリの有効なソース ファイルを追加しています。

```
add_files -fileset constrs_1 -quiet c:/Design/top.xdc c:/Design/project_1
```

また、`-quiet` オプションが使用されているので、コマンド ライン エラーは表示されません。

`-norecurse` オプションが指定されていれば、`project_1` ディレクトリの制約ファイルのみが追加され、下位ディレクトリは検索されません。

次の例では、既存の IP コア ファイルを現在のプロジェクトに追加しています。

```
add_files -norecurse C:/Data/ip/c_addsub_v11_0_0.xci
```

注記: IP ファイルをローカル プロジェクト フォルダにインポートするには、`import_ip` コマンドを使用します。

次の例では、非プロジェクト モードの最上位デザイン ネットリストを読み込み、`char_fifo` IP ファイルを追加しています。

```
# Read top-level EDIF and IP DCP
read_edif ./sources/wave_gen.edif
add_files ./my_IP/char_fifo/char_fifo.xci
```

注記: IP コアを読み込むには、`add_files` または `read_ip` コマンドを使用できます。デザイン チェックポイント (DCP) を含む IP のすべての出力ファイルが必要に応じて読み込まれます。

次の例では、System Generator で作成した既存の DSP モジュールを現在のプロジェクトに追加しています。

```
add_files C:/Data/model11.mdl
```

注記: System Generator を使用して DSP モジュールを生成するには、`create_sysgen` コマンドを使用します。

関連項目

- [create_sysgen](#)
- [import_files](#)
- [import_ip](#)
- [read_ip](#)
- [read_verilog](#)
- [read_vhdl](#)
- [read_xdc](#)

add_force

信号、ワイヤ、またはレジスタを特定の値に強制的に設定します。

構文

```
add_force [-radix <arg>] [-repeat_every <arg>] [-cancel_after <arg>]
          [-quiet] [-verbose] <hdl-object> <values>...
```

戻り値

追加された force オブジェクト

使用法

名前	説明
[-radix]	使用する基数を指定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-repeat_every]	指定の時間間隔で繰り返します。
[-cancel_after]	指定の時間後にキャンセルします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hdl-object>	force を追加するオブジェクトを指定します。
<values>	force に値と時間オフセットを追加します ({<value> [<time>]})。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレーション中に信号、ワイヤ、またはレジスタを特定の値に強制的に設定します。

add_force コマンドは、テストベンチまたはモジュール定義の Verilog force/release コマンドと同じように機能します。指定の期間、-cancel_after オプションで指定されたイベントまで、または remove_forces コマンドが使用されるまで、HDL オブジェクトを指定の値に維持します。



重要: テストベンチまたはモジュールの HDL オブジェクトに Verilog force/release 文がある場合、それらの文より Tcl add_force コマンドが優先されます。Tcl force が期限切れになるかまたは解除されると、シミュレーションで、Verilog force の実行も含め、HDL オブジェクトが通常の動作に戻ります。

このコマンドを実行すると、作成された force オブジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。返された force オブジェクトの名前は、remove_forces コマンドを使用する場合に重要であり、後で使えるよう、下の例に示すように Tcl 変数に代入しておく必要があります。

引数

`-radix <arg>` (オプション): `<values>` を指定する際の基数を指定します。有効な値は `default`、`dec`、`bin`、`oct`、`hex`、`unsigned`、`ascii`、および `smag` です。指定した HDL オブジェクト タイプに基数が定義されていない場合、デフォルトの基数は 2 進数 (bin) です。

`-repeat_every <arg>` (オプション): `add_force` を指定の時間ごとに繰り返します。指定の `<hdl_object>` に対して反復 `force` を作成する場合に使用できます。

注記: このオプションで指定する時間は、`<values>` を `{<value> <time>}` で指定した場合の時間よりも長くする必要があります。そうでないと、エラーが発生します。

`-cancel_after <arg>` (オプション): `current_time` からの指定された時間後に `force` を解除します。これは、指定の時間後に `remove_forces` コマンドを実行した場合と同じです。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hdl_object>` (必須): 値を強制する 1 つの HDL オブジェクトの名前を指定します。オブジェクトは、名前または `get_objects` コマンドで指定します。有効なオブジェクトは信号、ワイヤ、およびレジスタです。

`<values>` (必須): HDL オブジェクトを強制する値を指定します。1 つの値を指定でき、シミュレーション中、`-cancel_after` オプションまたは `remove_forces` コマンドを使用して解除されるまで、その値に維持されます。

指定の値 (`<value>`) は、`<hdl_object>` のタイプによって異なります。HDL オブジェクト タイプには、`logic`、浮動小数点、VHDL 列挙型、VHDL 整数があります。`logic` 以外では、`-radix` オプションは無視されます。

- `logic` は、実際の HDL オブジェクト タイプを示すのではなく、次のような VHDL `std_logic` と同様の値を持つオブジェクトを示します。
 - Verilog 暗示 4 値ビット型。
 - VHDL ビットおよび `std_logic` 既定義型。
 - `std_logic` のサブセットである VHDL 列挙型 (文字リテラル 0 および 1 を含む)。
- `logic` 型では、値は基数によって異なります。
 - 指定の値のビット数が `logic` 型に適切なビット数より少ない場合、値は適切な長さになるよう 0 で拡張されません (符号拡張されません)。
 - 指定の値のビット数が `logic` 型に適切なビット数より多い場合、MSB 側の余分なビットはすべて 0 である必要があります、そうでない場合は Vivado シミュレータでサイズ不一致エラーが返されます。
- 浮動小数点オブジェクトの値は浮動小数点値です。
- `logic` でない VHDL 列挙型の値は、値の列挙セットからのスカラー値 (文字の場合はシングル クォーテーションなし) です。
- VHDL 整数型の値は、その型で許容される範囲の符号付き 10 進整数です。

<value> は {<value> <time>} のペアで指定することもでき、この場合 HDL オブジェクトが現在の時間から <time> で指定された期間 <value> で指定された値に維持され、その後次のペアの <time> で指定された期間 <value> で指定された値に維持され、最後の {<value> <time>} ペアまでそのように値が強制されます。

注記: 最初の {<value> <time>} ペアでは <time> はオプションであり、指定されていない場合は 0 であると想定されます。その後の {<value> <time>} ペアでは、<time> は必須です。

{<value> <time>} ペアで指定される <time> は、現在のシミュレーション時間に対して定義され、`current_time` からの期間を示します。たとえば、現在のシミュレーション時間が 1000 ns の場合、<time> を 20 ns にすると、1000 ns ~ 1020 ns までの期間が定義されます。



重要: <time> はシミュレーションのタイム ラインで増加していくように指定する必要があり、繰り返すとエラーが発生します。

<time> は、現在のシミュレーションのデフォルト `TIME_UNIT` で定義するか、時間単位をスペースなしで含めて指定できます。有効な時間単位は fs、ps、ns、us、ms、s です。50 を指定すると、デフォルトで 50 ns に設定されます。50 ps を指定すると、50 ピコ秒に設定されます。

例

次の例では、リセット信号をデフォルトの基数を使用して 300 ns 間 High に設定し、返された force オブジェクトの名前を後で force を解除するために使用できるように Tcl 変数に保存しています。

```
set for10 [ add_force reset 1 300 ]
```

次の例では、{<value> <time>} ペアの使用法を示しており、定期的に繰り返して、指定の時間後キャンセルします。

```
add_force mySig {0} {1 50} {0 100} {1 150} -repeat_every 200
-cancel_after 10000
```

注記: この例では、最初の {<value> <time>} ペアでは時間は設定されていません。これは、指定の値 0 が時間 0 (`current_time`) に適用されることを示します。

関連項目

- [current_time](#)
- [get_objects](#)
- [remove_forces](#)

add_hw_hbm_pc

指定したハードウェア HBM のアクティビティ モニターに追加する擬似チャネルを選択します。まずメモリ コントローラーの番号を指定し、その後に擬似チャネル番号を指定します。

構文

```
add_hw_hbm_pc [-quiet] [-verbose] <mc_num> <pc_num> <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><mc_num></code>	選択するメモリ コントローラーの番号 (0 ~ 7) を指定します。
<code><pc_num></code>	選択する擬似チャネルの番号 (0 または 1) を指定します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

HBM アクティビティ モニターは、統合高帯域幅メモリ (HBM) コントローラーを含む特定のザイリンクス UltraScale + デバイスのパフォーマンス モニターおよび温度センサーにリアルタイムにアクセスするために使用できます。このコアの詳細は、『LogiCORE IP 製品ガイド: AXI High Bandwidth Memory Controller』(PG276) を参照してください。HBM コントローラーおよびメモリ スタックにはパフォーマンス カウンターと温度センサーの両方が含まれ、ザイリンクス Vivado ハードウェア マネージャー内から HBM アクティビティ モニターを使用してアクセスできます。各 HBM スタックは 8 個の独立したメモリ チャネルに分割され、それらのチャネルがさらに 2 つの 64 ビット擬似チャネル (pc) に分割されます。

HBM をイネーブルにしたデバイスを AXI High Bandwidth Memory Controller のインスタンスを含むデザインでコンフィギュレーションすると、Vivado ハードウェア マネージャーに HBM コアが表示されます。add_hw_hbm_pc コマンドを使用すると、run_hw_hbm_amon コマンドを使用する前に、HBM アクティビティ モニターでの監視用にメモリ チャネル (mc)/擬似チャネル (pc) を指定できます。



ヒント: 擬似チャネルを追加または削除するときには、HBM アクティビティ モニターが実行されていないことを確認してください。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<mc_num> (必須): HBM コントローラーの 8 つのメモリ チャンネル (0 ~ 7) のうち、HBM アクティビティ モニターに追加するチャンネルを 1 つ指定します。

<pc_num> (必須): 2 つの擬似チャンネル (0 または 1) のいずれかを指定します。

<hw_objects> (必須): モニターする HBM コントローラー (hw_hbm) を指定します。

例

次の例では、指定の HBM コントローラー (hw_hbm) オブジェに指定のメモリ チャンネル/擬似チャンネルを追加し、Vivado ハードウェア マネージャーで HBM アクティビティ モニターを実行しています。

```
set hbm_mon [get_hw_hbms *HBM_2]
add_hw_hbm_pc 0 0 $hbm_mon
add_hw_hbm_pc 0 1 $hbm_mon
add_hw_hbm_pc 1 0 $hbm_mon
add_hw_hbm_pc 1 1 $hbm_mon
add_hw_hbm_pc 2 0 $hbm_mon
add_hw_hbm_pc 2 1 $hbm_mon
add_hw_hbm_pc 3 0 $hbm_mon
add_hw_hbm_pc 3 1 $hbm_mon
run_hw_hbm_amon $hbm_mon
```

関連項目

- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

add_hw_probe_enum

列挙名と値のペアを hw_probe に追加します。

構文

```
add_hw_probe_enum [-no_gui_update] [-dict <args>] [-quiet] [-verbose]
                  <name> <value> <hw_probe>
```

使用法

名前	説明
<code>[-no_gui_update]</code>	GUI をアップデートしません。
<code>[-dict]</code>	パラメーター名と値のペアを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	列挙名を指定します。
<code><value></code>	値を指定します。
<code><hw_probe></code>	hw_probe オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

列挙名と値のペアを指定したハードウェア プロブ オブジェクトに割り当てます。

このコマンドは、Vivado ロジック解析で信号のステートを監視しやすくするためのものです。このコマンドを使用すると、ハードウェア プロブ (hw_probe) オブジェクトに含まれる指定した値に関連付けるステートのセット (列挙名) を定義できます。これにより、シンボル名をトリガー値および波形データ値と比較して、ステート マシン プロブおよびその他のタイプのプロブを監視できます。

列挙名は指定した hw_probe オブジェクトの ENUM.NAME プロパティとして追加され、プロブの指定したビット値に関連付けられます。列挙名は、hw_probe のトリガー/キャプチャ比較値を指定するために使用できます。



ヒント: 列挙名は、Vivado ロジック解析の波形ビューアーに表示されます。列挙名の表示は、プロブごとにディスプレイブルにできます。波形ビューアーの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

このコマンドを実行すると、列挙名プロパティが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-no_gui_update` (オプション): プロブの列挙値の変更で Vivado ロジック解析の GUI をアップデートしません。

`-dict` (オプション): `hw_probe` のプロパティ名 (`<name>`) と値 (`<value>`) のペアを列挙したディクショナリを指定します。 `<name>` と `<value>` のペアは、ダブルクォーテーション (") または波かっこ ({}) で囲んで指定します。

```
-dict "name1 value1 name2 value2 ... nameN valueN"
```



ヒント: 複数の列挙値を指定する場合は、 `<name>` と `<value>` の代わりに `-dict` オプションを使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、 `set_msg_config` コマンドで定義できます。

`<name>` (必須): `hw_probe` オブジェクトの指定した `<value>` に関連付けられている ENUM プロパティの名前を指定します。列挙名プロパティでは、大文字/小文字が区別されます。指定した名前は、`hw_probe` 上のビット値が指定の `<value>` と一致すると使用されます。

`<value>` (必須): 定義された列挙名 (`<name>`) に関連付ける `hw_probe` オブジェクト上のビット値を指定します。この値は、2 進数、8 進数、16 進数の符号付または符号なしの値で指定できます。



重要: 2 進ビット値 x およびエッジ ビット値 (F、B、RT) は指定できません。

`<hw_probe>` (必須): 列挙名プロパティを割り当てる `hw_probe` オブジェクトを指定します。

例

次の例では、`-dict` オプションを使用して指定の `hw_probe` オブジェクトの名前と値のペアを定義しています。

```
add_hw_probe_enum -dict {ZERO eq5'h00 RED eq5'h12 GREEN eq5'h13 \
    BLUE eq5'h14 WHITE eq5'h15 YELLOW eq5'h16 GREY eq5'h17} \
    [get_hw_probes op1 -of_objects [current_hw_ila]]
```

次の例では、指定の `hw_probe` オブジェクトの列挙名と値のペアを定義しています。

```
add_hw_probe_enum ZERO eq5'h00 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum RED eq5'h12 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum GREEN eq5'h13 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum BLUE eq5'h14 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum WHITE eq5'h15 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum YELLOW eq5'h16 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
add_hw_probe_enum GREY eq5'h17 [get_hw_probes op1 \
    -of_objects [current_hw_ila]]
```

次の例では、指定した hw_probe オブジェクトに割り当てられている ENUM プロパティを返しています。

```
report_property [get_hw_probes op1 -of_objects [current_hw_ila]] ENUM*
Property      Type      Read-only  Visible  Value
ENUM.ZERO     string  true      true     eq5'h00
ENUM.RED      string  true      true     eq5'h12
ENUM.GREEN    string  true      true     eq5'h13
ENUM.BLUE     string  true      true     eq5'h14
ENUM.WHITE    string  true      true     eq5'h15
ENUM.YELLOW   string  true      true     eq5'h16
ENUM.GREY     string  true      true     eq5'h17
```

関連項目

- [current_hw_device](#)
- [current_hw_ila](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [remove_hw_probe_enum](#)
- [report_property](#)

add_peripheral_interface

ペリフェラルに新しいバス インターフェイスを追加します。

構文

```
add_peripheral_interface -interface_mode <arg> -axi_type <arg> [-quiet]
                        [-verbose] <name> <peripheral>
```

使用法

名前	説明
-interface_mode	インターフェイスのモードを指定します。指定可能な値は master または slave です。
-axi_type	AXI インターフェイスのタイプを指定します。指定可能な値は lite、full、stream です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	新しく追加するインターフェイスの名前 (S1_AXI、M1_AXI など) を指定します。
<peripheral>	ペリフェラル オブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[CreatePeripheral \(ペリフェラルの作成\)](#)

説明

create_peripheral コマンドで作成したペリフェラルに AXI バス インターフェイスを追加します。

引数

-interface_mode [master | slave] (オプション): インターフェイスをマスターまたはスレーブに指定します。マスター インターフェイスは出力される AXI トランザクションを生成するので、AXI 転送のソースとなります。スレーブ インターフェイスは入力される AXI トランザクションを受信するので、AXI 転送のターゲットとなります。

-axi_type <arg> (オプション): 追加する AXI インターフェイスのタイプを指定します。指定可能な値は full、lite、および stream です。

- フル AXI4 インターフェイスはメモリ マップされたインターフェイス用で、1 つのアドレス フェーズのみで 256 データ転送サイクルまでのバーストが可能です。
- AXI4-Lite プロトコルは AXI4 プロトコルのサブセットで、より単純で小型の制御レジスタ型インターフェイスとの通信用です。
- AXI4-Stream は、マスターからスレーブの一方方向のデータ転送用で、信号配線が大幅に削減されます。

注記: AXI インターフェイスの詳細は、『AXI リファレンス ガイド』 (UG761) を参照してください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 追加するインターフェイスの名前を指定します。

`<peripheral>` (必須): インターフェイスを追加するペリフェラルを指定します。ペリフェラルは `create_peripheral` コマンドで作成し、このコマンドおよびその他の関連コマンドで処理しやすいように Tcl 変数に格納してください。例を参照してください。

例

次の例では、VLNV 属性を指定して新しい AXI ペリフェラルを作成し、そのペリフェラル オブジェクトを後の処理用に Tcl 変数に格納して、そのペリフェラルに AXI スレーブ インターフェイスを追加しています。

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} {1.3} \
    -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S2_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
```

関連項目

- [create_peripheral](#)
- [generate_peripheral](#)
- [write_peripheral](#)

add_wave

新しい波形を追加します。

構文

```
add_wave [-into <args>] [-at_wave <args>] [-after_wave <args>]
         [-before_wave <args>] [-reverse] [-radix <arg>] [-color <arg>]
         [-name <arg>] [-recursive] [-r] [-regexp] [-nocase] [-quiet] [-verbose]
         <items>...
```

戻り値

新しい波形

使用法

名前	説明
[-into]	新しい波形オブジェクトを挿入する波形設定、グループ、または仮想バスを指定します。
[-at_wave]	指定の波形オブジェクト、あるいはグループまたは仮想表示しませんでない場合は指定の波形オブジェクトの後に新しい波形オブジェクトを挿入します。
[-after_wave]	指定の波形オブジェクトの後に新しい波形オブジェクトを挿入します。
[-before_wave]	指定の波形オブジェクトの前に新しい波形オブジェクトを挿入します。
[-reverse]	新しい波形オブジェクトの表示ビット順を逆にします。
[-radix]	新しい波形オブジェクトの表示基数を設定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-color]	新しい波形オブジェクトの色を設定します。標準の色の名前を指定するか、RRGGBB 形式で指定します。
[-name]	1 つの新しい波形オブジェクトの表示名を設定します。
[-recursive]	デザイン オブジェクトがスコープの場合に、そのスコープ内のデザイン オブジェクトすべての波形オブジェクトを作成します。
[-r]	デザイン オブジェクトがスコープの場合に、そのスコープ内のデザイン オブジェクトすべての波形オブジェクトを作成します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<items>	波形オブジェクトを作成する基になるデザイン オブジェクトを指定します。

カテゴリ

Waveform (波形)

説明

`add_wave` コマンドは、1 つまたは複数のデザインに基づく波形オブジェクトを作成します。

このコマンドを実行すると、新しく作成された波形オブジェクトの名前が返されます。

注記: このコマンドは、シミュレーションでのみ使用できます。<item> には、シミュレーション プロジェクトの HDL オブジェクト (信号) を少なくとも 1 つ指定する必要があります。Vivado インターフェイスでは、オブジェクトは [Objects] ウィンドウに表示されます。

引数

`-into <wcfgGroupVbusObj>` (オプション): 新しい波形オブジェクトを挿入する波形設定、グループ、または仮想バスを指定します。<wcfgGroupVbusObj> がオブジェクトではなく文字列の場合は、現在の WCFG でその名前のグループが検索されます。その名前のグループが見つからない場合は、現在の WCFG の仮想バス名が検索されます。その名前の仮想バスも見つからない場合は、すべての WCFG オブジェクトの名前が検索されます。<into> オプションでオブジェクトを指定しない場合は、現在の波形設定に追加されます。

`-at_wave <waveObj>` (オプション): 指定の波形オブジェクトの位置に波形オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-after_wave <waveObj>` (オプション): 指定の波形オブジェクトの後に波形オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-before_wave <waveObj>` (オプション): 指定の波形オブジェクトの前に波形オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-reverse` (オプション): 新しい波形オブジェクトの `IS_REVERSED` プロパティを `true` に設定します。

`-radix <arg>` (オプション): 新しい波形オブジェクトの基数プロパティを設定します。有効な値は `default`、`dec`、`bin`、`oct`、`hex`、`unsigned`、`ascii`、および `smag` です。

`-color <arg>` (オプション): 新しい波形オブジェクトの色プロパティを設定します。定義済みの色の名前を指定するか、6 桁の RGB 形式 (`#RRGGBB`) で指定します。

`-name <arg>` (オプション): 新しい波形オブジェクトの `DISPLAY_NAME` プロパティを設定します。複数の波形オブジェクトが作成されている場合は、エラーになります。

`-recursive` | `-r` (オプション): <items> でスコープが指定されている場合、このスコープのすべてのサブスコープを追加します。

`-regexp` (オプション): <items> が正規表現を使用して記述されていることを指定します。ザイリンクス Tcl コマンドでは、正規表現は常に検索文字列の先頭にアンカーされています。検索文字列の先頭に「`.*`」を追加して、検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<items>` (必須): 指定した HDL オブジェクトの波形を現在のシミュレーションに追加します。

例

次の例は、波形グループを定義してそのグループの色を定義し、名前が `s` で始まり 2 文字以上のすべての信号を波形ウィンドウのその波形グループに追加します。

```
set groupColor YELLOW
set AXIS_ID [add_wave_group "Streaming Data"]
add_wave -into $AXIS_ID -color $groupColor -regexp s.*
```

次の例は、`rsb_design_testbench` の `dout_tvalid` 信号を既存のシミュレーション波形設定に追加します。

```
add_wave dout_tvalid /rsb_design_testbench/dout_tvalid
```

関連項目

- [add_wave_divider](#)
- [add_wave_group](#)
- [add_wave_marker](#)
- [add_wave_virtual_bus](#)

add_wave_divider

仕切りを追加します。

構文

```
add_wave_divider [-into <args>] [-at_wave <args>] [-after_wave <args>]
                 [-before_wave <args>] [-color <arg>] [-quiet] [-verbose] [<name>]
```

戻り値

新しい仕切り

使用法

名前	説明
[-into]	新しい仕切りを挿入する波形設定またはグループを指定します。
[-at_wave]	指定の波形オブジェクト、またはグループでない場合は指定の波形オブジェクトの後に新しい仕切りを挿入します。
[-after_wave]	指定の波形オブジェクトの後に新しい仕切りを挿入します。
[-before_wave]	指定の波形オブジェクトの前に新しい仕切りを挿入します。
[-color]	新しい仕切りの色を設定します。標準の色の名前を指定するか、RRGGBB 形式で指定します。デフォルトは default です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	新しい仕切りの表示名を指定します。デフォルトは new_divider です。

カテゴリ

Waveform (波形)

説明

波形ビューアーに仕切りを追加します。仕切りは、見やすくするために関連のオブジェクトのグループを仕切るために使用します。

仕切りは、指定の波形設定 (WCFG) または現在の波形設定 (WCFG) の指定の位置に追加できます。位置が指定されていない場合は、波形設定の最後に追加されます。

このコマンドを実行すると、新しく作成された仕切りの名前が返されます。

注記: このコマンドは、シミュレーションでのみ使用できます。

引数

`-into <wcfgGroupVbusObj>` (オプション): 新しい波形仕切りオブジェクトを挿入する波形設定、グループ、または仮想バスを指定します。<wcfgGroupVbusObj> がオブジェクトではなく文字列の場合は、現在の WCFG でその名前のグループが検索されます。その名前のグループが見つからない場合は、現在の WCFG の仮想バス名が検索されます。その名前の仮想バスも見つからない場合は、すべての WCFG オブジェクトの名前が検索されます。`-into` オプションでオブジェクトを指定しない場合は、現在の波形設定に追加されます。

`-at_wave <waveObj>` (オプション): 指定の波形オブジェクトの位置に仕切りを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-after_wave <waveObj>` (オプション): 指定の波形オブジェクトの後に仕切りを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-before_wave <waveObj>` (オプション): 指定の波形オブジェクトの前に仕切りを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-color <arg>` (オプション): 仕切りの色プロパティを設定します。定義済みの色の名前を指定するか、6 桁の RGB 形式 (RRGGBB) で指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (オプション): 指定した表示名の仕切りを作成します。デフォルト名は `new_divider` です。

例

次の例では、CLK 波形オブジェクトの後に Div1 という名前の仕切りを挿入します。

```
add_wave_divider -after_wave CLK Div1
```

関連項目

- [add_wave](#)
- [add_wave_group](#)
- [add_wave_marker](#)
- [add_wave_virtual_bus](#)

add_wave_group

グループを追加します。

構文

```
add_wave_group [-into <args>] [-at_wave <args>] [-after_wave <args>]  
               [-before_wave <args>] [-quiet] [-verbose] [<name>]
```

戻り値

新しいグループ

使用法

名前	説明
[-into]	新しいグループを挿入する波形設定またはグループを指定します。
[-at_wave]	指定の波形オブジェクト、またはグループでない場合は指定の波形オブジェクトの後に新しいグループを挿入します。
[-after_wave]	指定の波形オブジェクトの後に新しいグループを挿入します。
[-before_wave]	指定の波形オブジェクトの前に新しいグループを挿入します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	新しいグループの表示名を指定します。デフォルトは new_group です。

カテゴリ

Waveform (波形)

説明

指定の波形設定または現在の波形設定に波形グループを作成します。波形オブジェクトと仕切りを波形グループに追加し、波形表示を見やすくできます。

波形グループは、指定の位置に挿入できます。位置が指定されていない場合は、指定の波形設定の最後に追加されます。

このコマンドを実行すると、新しく作成された波形グループ オブジェクトの名前が返されます。

注記: このコマンドは、シミュレーションでのみ使用できます。

引数

`-into <wcfgGroupVbusObj>` (オプション): 新しい波形グループ オブジェクトを挿入する波形設定、グループ、または仮想バスを指定します。<wcfgGroupVbusObj> がオブジェクトではなく文字列の場合は、現在の WCFG でその名前のグループが検索されます。その名前のグループが見つからない場合は、現在の WCFG の仮想バス名が検索されます。その名前の仮想バスも見つからない場合は、すべての WCFG オブジェクトの名前が検索されます。`-into` オプションでオブジェクトを指定しない場合は、現在の波形設定に追加されます。

`-at_wave <waveObj>` (オプション): 指定の波形オブジェクトの位置に波形グループ オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-after_wave <waveObj>` (オプション): 指定の波形オブジェクトの後に波形グループ オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-before_wave <waveObj>` (オプション): 指定の波形オブジェクトの前に波形グループ オブジェクトを追加します。<waveObj> が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (オプション): 指定した表示名の波形グループを作成します。デフォルト名は `new_group` です。

例

次の例では、clk を既存の波形設定に追加しています。

```
add_wave_group clk
group10
```

関連項目

- [add_wave](#)
- [add_wave_divider](#)
- [add_wave_marker](#)
- [add_wave_virtual_bus](#)

add_wave_marker

新しい波形マーカーを作成します。

構文

```
add_wave_marker [-into <arg>] [-name <arg>] [-quiet] [-verbose] [<time>]  
                [<unit>]
```

戻り値

作成されたマーカー

使用法

名前	説明
[-into]	マーカーを作成する波形設定を指定します。
[-name]	マーカーの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<time>]	新しいマーカーを指定した時間に作成します。デフォルトは 0 です。
[<unit>]	マーカーの時間の単位を指定します。有効な値は fs、ps、ns、us、ms、および s です。

カテゴリ

[Waveform \(波形\)](#)

説明

現在の波形設定の指定の時間に指定した名前の波形マーカーを作成します。

このコマンドを実行しても、何も返されません。

注記: このコマンドは、シミュレーションでのみ使用できます。

引数

-into <wcfg> (オプション): 波形マーカーを挿入する波形設定を指定します。-into オプションを指定しない場合、波形マーカーは現在の波形設定に追加されます。

-name <arg> (オプション): マーカーの名前を指定します。デフォルト名は new_marker です。

<time> (オプション): マーカーを追加する波形上のシミュレーション時間を指定します。デフォルト値は時間 0 です。

<unit> (オプション): 時間の単位を指定します。有効な値は s、ms、us、ns、および ps です。デフォルトは、指定の波形設定で使用されている時間単位です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、既存の波形設定の時間 500 ns にマーカーを追加しています。

```
add_wave_marker 500 ns
```

関連項目

- [add_wave](#)
- [add_wave_divider](#)
- [add_wave_group](#)
- [add_wave_virtual_bus](#)

add_wave_virtual_bus

新しい仮想バスを追加します。

構文

```
add_wave_virtual_bus [-into <args>] [-at_wave <args>] [-after_wave <args>]
    [-before_wave <args>] [-reverse] [-radix <arg>] [-color <arg>] [-quiet]
    [-verbose] [<name>]
```

戻り値

新しい仮想バス

使用法

名前	説明
[-into]	新しい仮想バスを挿入する波形設定、グループ、または仮想バスを指定します。
[-at_wave]	指定の波形オブジェクト、あるいはグループまたは仮想バスでない場合は指定の波形オブジェクトの後に新しい仮想バスを挿入します。
[-after_wave]	指定の波形オブジェクトの後に新しい仮想バスを挿入します。
[-before_wave]	指定の波形オブジェクトの前に新しい仮想バスを挿入します。
[-reverse]	新しい仮想バスの表示ビット順を逆にします。
[-radix]	新しい仮想バスの表示基数を設定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-color]	新しい仮想バスの色を設定します。標準の色の名前を指定するか、RRGGBB 形式で指定します。デフォルトは default です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	新しい仮想バスの表示名を指定します。デフォルトは new_virtual_bus です。

カテゴリ

Waveform (波形)

説明

add_wave_virtual_bus コマンドは、指定した名前 (<name>) の新しい仮想バスを作成します。指定した波形設定 (WCFG) またはデフォルトでは既存の WCFG の一番下に仮想バスが挿入されます。作成された仮想バス vb### が返されます。

注記: このコマンドは、シミュレーションでのみ使用できます。少なくとも、新しい仮想バスの名前を指定する必要があります。

引数

`-into <wcfgGroupVbusObj>` (オプション): 仮想バスを挿入する波形設定、グループ、または仮想バスを指定します。`<wcfgGroupVbusObj>` がオブジェクトではなく文字列の場合は、現在の WCFG でその名前のグループが検索されます。その名前のグループが見つからない場合は、現在の WCFG の仮想バス名が検索されます。その名前の仮想バスも見つからない場合は、すべての WCFG オブジェクトの名前が検索されます。`-into` オプションでオブジェクトを指定しない場合は、現在の波形設定に追加されます。

`-at_wave <waveObj>` (オプション): 指定の波形オブジェクトの位置に仮想バス オブジェクトを追加します。`<waveObj>` が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-after_wave <waveObj>` (オプション): 指定の波形オブジェクトの後に仮想バス オブジェクトを追加します。`<waveObj>` が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-before_wave <waveObj>` (オプション): 指定の波形オブジェクトの前に仮想バスを追加します。`<waveObj>` が文字列の場合は、波形オブジェクトの表示名として処理されます。

`-reverse` (オプション): 新しい仮想バス オブジェクトの `IS_REVERSED` プロパティを `true` に設定します。

`-radix <arg>` (オプション): 新しい仮想バス オブジェクトの基数プロパティを設定します。有効な値は `default`、`dec`、`bin`、`oct`、`hex`、`unsigned`、`ascii`、および `smag` です。

`-color <arg>` (オプション): 新しい仮想バス オブジェクトの色プロパティを設定します。定義済みの色の名前を指定するか、6 桁の RGB 形式 (RRGGBB) で指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (オプション): 新しい仮想バス オブジェクトの `DISPLAY_NAME` プロパティを `<name>` に設定します。

例

次の例では、`dout_tvalid` という名前の仮想バスを現在の波形設定の最後に追加しています。

```
add_wave_virtual_bus dout_tvalid
```

関連項目

- [add_wave_divider](#)
- [add_wave_group](#)
- [add_wave_marker](#)
- [add_wave](#)

all_clocks

現在のデザインに含まれるすべてのクロックのリストを取得します。

構文

```
all_clocks [-quiet] [-verbose]
```

戻り値

クロック オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC

説明

現在のデザインで宣言されたすべてのクロックのリストを返します。

デザインの特定のクロックのリストを取得するには、`get_clocks` コマンドを使用するか、`filter` コマンドを使用して `all_clocks` で返された結果にフィルターを適用します。

クロックを定義するには、`create_clock` または `create_generated_clock` コマンドを使用します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、サンプル CPU ネットリスト プロジェクトのすべてのクロックが表示されます。

```
% all_clocks
cpuClk wbClk usbClk phy_clk_pad_0_i phy_clk_pad_1_i fftClk
```

次の例では、`set_propagated_clock` コマンドをすべてのクロックに適用し、返されたリスト (`all_clocks`) を別のコマンドに渡しています。

```
% set_propagated_clock [all_clocks]
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [filter](#)
- [get_clocks](#)
- [set_propagated_clock](#)

all_cpus

現在のデザインの CPU セルのリストを取得します。

構文

```
all_cpus [-quiet] [-verbose]
```

戻り値

CPU セル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC

説明

現在のデザインの CPU セル オブジェクトのリストを返します。現在のデザインで宣言されたすべての CPU セル オブジェクトのリストを作成します。

`all_cpus` で返された CPU のリストに対して `filter` コマンドを使用すると、CPU セル オブジェクトのプロパティに基づいてリストをフィルターできます。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

`all_cpus` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

注記: このコマンドを実行すると、CPU セル オブジェクトのリストが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの CPU オブジェクトすべてが返されます。

```
all_cpus
```

次の例では、返されたリストを別のコマンドに渡しています。

```
set_false_path -from [all_cpus] -to [all_registers]
```

関連項目

- [all_dsps](#)
- [all_hsios](#)
- [all_registers](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)
- [list_property](#)
- [report_property](#)
- [set_false_path](#)

all_dsps

現在のデザインの DSP セルのリストを取得します。

構文

```
all_dsps [-quiet] [-verbose]
```

戻り値

DSP セル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC

説明

現在のデザインで宣言されたすべての DSP セル オブジェクトのリストを返します。

`all_dsps` で返された DSP のリストに対して `filter` コマンドを使用すると、DSP オブジェクトのプロパティに基づいてリストをフィルターできます。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

`all_dsps` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインで定義されているすべての DSP のリストを取得し、そのリストをフィルターして指定のサイトに割り当てられている 1 つの DSP を取得しています。

```
filter [all_dsps] {SITE == DSP48_X1Y6}
```

次の例では、返されたリストを別のコマンドに渡しています。

```
set_false_path -from [all_dsps] -to [all_registers]
```

関連項目

- [all_cpus](#)
- [all_hsios](#)
- [all_registers](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)
- [list_property](#)
- [report_property](#)
- [set_false_path](#)

all_fanin

指定したシンクのファンインのピンまたはセルのリストを取得します。

構文

```
all_fanin [-startpoints_only] [-flat] [-only_cells] [-levels <arg>]
          [-pin_levels <arg>] [-trace_arcs <arg>] [-quiet] [-verbose] <to>
```

戻り値

セルまたはピン オブジェクトのリスト

使用法

名前	説明
<code>[-startpoints_only]</code>	タイミング始点のみを検出します。
<code>[-flat]</code>	階層を無視します。
<code>[-only_cells]</code>	セルのみを検出します。
<code>[-levels]</code>	処理するセル レベルの最大数を指定します。0 以上の値を指定します。デフォルト値は0です。
<code>[-pin_levels]</code>	処理するピン レベルの最大数を指定します。0 以上の値を指定します。デフォルト値は0です。
<code>[-trace_arcs]</code>	トレースするネットワーク アークのタイプを指定します。有効な値は timing、enabled、all です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><to></code>	シンク ピン、ポート、またはネットを指定します。

カテゴリ

XDC

説明

指定したシンクのファンインのポート、ピン、またはセルのリストを返します。

`all_fanin` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。すべての階層レベルのファンインを取得するには、`-flat` オプションを使用します。

引数

`-startpoints_only` (オプション): タイミング始点のみを検出します。このオプションを使用すると、ファンイン ネットワークの中間点は返されません。このオプションは、シンクのプライマリ ドライバーを特定する場合に使用できます。

-flat (オプション): デザインの階層を無視します。デフォルトでは、シンクと同じ階層レベルのオブジェクトのみが返されます。このオプションを使用すると、階層にかかわらず、シンクのファンイン ネットワークのすべてのオブジェクトが返されます。

-only_cells (オプション): 指定したシンクのファンイン パスにあるセル オブジェクトのみを返します。ピンまたはポートは返しません。

-levels <value> (オプション): 処理するセル レベルの最大数を指定します。1 を指定すると、現在のインスタンス最上位が処理されます。デフォルト値は 0 で、階層のすべてのレベルが処理されます。

-pin_levels <value> (オプション): 処理するピン レベルの最大数を指定します。デフォルト値は 0 です。

-trace_arcs <value> (オプション): トレースするネットワーク アークのタイプを指定します。有効な値は timing、enabled、および all です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<to> (必須): ファンイン オブジェクトをレポートするピン、ポート、またはネットを指定します。

例

次の例では、led_pins 出力ポートのタイミング ファンインがリストされます。

```
all_fanin [get_ports led_pins[*] ]
```

次の例では、指定のフリップフロップのクロック ピンからクロック ソース (この例の場合は MMCM 出力ピン) までをトレースします。

```
all_fanin -flat -startpoints_only [get_pins cmd_parse_i0/prescale_reg[7]/C]
```

次の例では、デザイン階層にかかわらず、IDELAY の入力ピンに接続されているポートを返しています。

```
all_fanin -flat -startpoints_only [get_pins IDELAY*/IDATAIN]
```

関連項目

- [all_fanout](#)
- [current_instance](#)
- [get_cells](#)
- [get_pins](#)
- [get_ports](#)

all_fanout

指定したソースのファンアウトのピンまたはセルのリストを取得します。

構文

```
all_fanout [-endpoints_only] [-flat] [-only_cells] [-levels <arg>]
           [-pin_levels <arg>] [-trace_arcs <arg>] [-quiet] [-verbose] <from>
```

戻り値

セルまたはピン オブジェクトのリスト

使用法

名前	説明
<code>[-endpoints_only]</code>	タイミング終点のみを検出します。
<code>[-flat]</code>	階層を無視します。
<code>[-only_cells]</code>	セルのみを検出します。
<code>[-levels]</code>	処理するセル レベルの最大数を指定します。0 以上の値を指定します。デフォルト値は 0 です。
<code>[-pin_levels]</code>	処理するピン レベルの最大数を指定します。0 以上の値を指定します。デフォルト値は 0 です。
<code>[-trace_arcs]</code>	トレースするネットワーク アークのタイプを指定します。有効な値は timing、enabled、all です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><from></code>	ソース ピン、ポート、またはネットを指定します。

カテゴリ

XDC

説明

指定したソースのファンアウトのポート、ピン、またはセルのリストを返します。

`all_fanout` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。すべての階層レベルのファンアウトを取得するには、`-flat` オプションを使用します。

引数

`-endpoints_only` (オプション): タイミング終点のみを検出します。このオプションを使用すると、ファンアウト ネットワークの中間点は返されません。このオプションは、ドライバーのプライマリ ロードを特定する場合に使用できます。

-flat (オプション): デザインの階層を無視します。デフォルトでは、シンクと同じ階層レベルのオブジェクトのみが返されます。このオプションを使用すると、階層にかかわらず、ドライバーのファンアウト ネットワークのすべてのオブジェクトが返されます。

-only_cells (オプション): 指定したソースのファンアウト パスにあるセル オブジェクトのみを返します。

-levels <value> (オプション): 処理するセル レベルの最大数を指定します。1 を指定すると、現在のインスタンス最上位が処理されます。デフォルト値は 0 で、階層のすべてのレベルが処理されます。

-pin_levels <value> (オプション): 処理するピン レベルの最大数を指定します。デフォルト値は 0 です。

-trace_arcs <value> (オプション): トレースするネットワーク アークのタイプを指定します。有効な値は timing、enabled、および all です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<from> (必須): リストするファンアウト パスのオブジェクトのソース ポート、ピン、またはネットを指定します。

例

次の例では、デザインのすべての入力ポートのファンアウトを取得しています。

```
all_fanout [all_inputs]
```

次の例では、現在のデザインの I/O バンク 15 に割り当てられているすべての入力のファンアウトを取得しています。

```
all_fanout [filter [all_inputs] {IOBANK == 15}]
```

関連項目

- [all_fanin](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)
- [get_pins](#)
- [get_ports](#)

all_ffs

現在のデザインのフリップフロップセルのリストを取得します。

構文

```
all_ffs [-quiet] [-verbose]
```

戻り値

フリップフロップセル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC

説明

現在のデザインのフリップフロップ インスタンスすべてのリストを返します。

特定のプロパティを持つフリップフロップ フロップセルのリストを取得するには、`get_cells` コマンドを使用するか、`all_ffs` の結果に対して `filter` コマンドを使用します。

`all_ffs` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインのすべてのフリップフロップを返し、fftEngine モジュールに含まれるフリップフロップをカウントしています。

```
current_instance
INFO: [Vivado 12-618] Current instance is the top level of design
'netlist_1'.
top
llength [all_ffs]
15741
current_instance fftEngine
fftEngine
llength [all_ffs]
1519
```

次の例では、all_ffs の結果にフィルターを適用して FDRE フリップフロップのみを取得しています。

```
filter [all_ffs] {REF_NAME == FDRE}
```

関連項目

- [all_latches](#)
- [all_registers](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)

all_hsios

現在のデザインの HSIO セルのリストを取得します。

構文

```
all_hsios [-quiet] [-verbose]
```

戻り値

HSIO セル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[XDC](#)

説明

現在のデザインで宣言されたすべての高速 I/O (HSIO) セル オブジェクトのリストを返します。これらの HSIO セル オブジェクトは、変数に代入するか、別のコマンドに渡すことができます。

`all_hsios` で返された高速 I/O のリストに対して `filter` コマンドを使用すると、HSIO オブジェクトのプロパティに基づいてリストをフィルタできます。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

`all_hsios` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの HSIO オブジェクトすべてが返されます。

```
all_hsios
```

次の例では、返されたリストを直接別のコマンドに渡しています。

```
set_false_path -from [all_hsios] -to [all_registers]
```

関連項目

- [all_cpus](#)
- [all_dsps](#)
- [all_registers](#)
- [filter](#)
- [get_cells](#)
- [list_property](#)
- [report_property](#)
- [set_false_path](#)

all_inputs

現在のデザインの入力ポートすべてのリストを取得します。

構文

```
all_inputs [-quiet] [-verbose]
```

戻り値

ポート オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[SDC](#)、[XDC](#)

説明

現在のデザインの入力ポート オブジェクトすべてのリストを返します。

デザインの特定の入力のリストを取得するには、`get_ports` コマンドを使用するか、`filter` コマンドを使用して `all_inputs` で返された結果にフィルターを適用します。

`all_inputs` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの入力ポート オブジェクトすべてが返されます。

```
all_inputs
```

次の例では、すべての入力ポート オブジェクトを取得し、GT ポートを除外して、GT ではないポートに I/O 規格を設定しています。

```
set non_gt_ports [filter [all_inputs] {!is_gt_term}]
set_property IOSTANDARD LVCMOS18 $non_gt_ports
```

次の例では、返されたリストを別のコマンドに渡しています。

```
set_input_delay 5 -clock REFCLK [all_inputs]
```

関連項目

- [all_clocks](#)
- [all_outputs](#)
- [current_instance](#)
- [filter](#)
- [get_clocks](#)
- [get_ports](#)
- [set_input_delay](#)
- [set_property](#)

all_latches

現在のデザインに含まれるラッチ セルすべてのリストを取得します。

構文

```
all_latches [-quiet] [-verbose]
```

戻り値

ラッチ セル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC

説明

現在のデザインで宣言されているすべてのラッチのリストを返します。

`all_latches` で返されたラッチのリストに対して `filter` コマンドを使用すると、ラッチのプロパティに基づいてリストをフィルターできます。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

`all_latches` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインに含まれるすべてのラッチのリストが返されます。

```
all_latches
```

次の例では、返されたリストを別のコマンドに渡しています。

```
set_false_path -from [all_mults] -to [all_latches]
```

関連項目

- [all_ffs](#)
- [all_registers](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)
- [list_property](#)
- [report_property](#)
- [set_false_path](#)

all_outputs

現在のデザインの出力ポートすべてのリストを取得します。

構文

```
all_outputs [-quiet] [-verbose]
```

戻り値

ポート オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[SDC](#)、[XDC](#)

説明

現在のデザインで宣言されたすべての出力ポート オブジェクトのリストを返します。

デザインの特定の出力のリストを取得するには、`get_ports` コマンドを使用するか、`filter` コマンドを使用して `all_outputs` で返された結果にフィルターを適用します。

`all_outputs` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの出力ポート オブジェクトすべてが返されます。

```
all_outputs
```

次の例では、デザインのすべての出力の出力遅延が設定されます。

```
set_output_delay 5 -clock REFCLK [all_outputs]
```

関連項目

- [all_inputs](#)
- [current_instance](#)
- [filter](#)
- [get_ports](#)
- [set_output_delay](#)

all_rams

現在のデザインの RAM セルのリストを取得します。

構文

```
all_rams [-quiet] [-verbose]
```

戻り値

RAM セル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC

説明

現在のインスタンスに含まれるすべての RAM セル オブジェクト (ブロック RAM、ブロック RAM FIFO、および分散 RAM を含む) のリストを返します。これらの RAM セル オブジェクトは、変数に代入するか、別のコマンドに渡すことができます。

デザインの特定の RAM セルのリストを取得するには、`all_rams` で返された結果に `filter` コマンドを使用して RAM セルに設定されたプロパティに基づいてフィルターを適用します。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

`all_rams` コマンドでは、デザインの最上位または現在のインスタンスのレベルにあるオブジェクトが返されます。デフォルトでは、デザインの最上位が現在のインスタンスとして定義されますが、これは `current_instance` コマンドを使用して変更できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインの RAM セルすべてが返されます。

```
all_rams
```

次の例では、デザインのすべての RAM セルを取得し、そのリストをフィルターして FIFO ブロック メモリのみを取得しています。

```
filter [all_rams] {PRIMITIVE_SUBGROUP == fifo}
```

次の例では、現在のインスタンスを設定し、現在のインスタンスの階層レベルにあるすべての RAM オブジェクトを取得しています。

```
current_instance usbEngine0  
all_rams
```

関連項目

- [all_cpus](#)
- [current_instance](#)
- [filter](#)
- [get_cells](#)
- [list_property](#)
- [report_property](#)

all_registers

現在のデザインのレジスタ セルまたはピンのリストを取得します。

構文

```
all_registers [-clock <args>] [-rise_clock <args>] [-fall_clock <args>]
             [-cells] [-data_pins] [-clock_pins] [-async_pins] [-output_pins]
             [-level_sensitive] [-edge_triggered] [-no_hierarchy] [-quiet]
             [-verbose]
```

戻り値

セルまたはピン オブジェクトのリスト

使用法

名前	説明
[-clock]	指定したクロックが供給されるレジスタを取得します。
[-rise_clock]	クロックの立ち上がりエッジでトリガーされるレジスタを取得します。
[-fall_clock]	クロックの立ち下がりエッジでトリガーされるレジスタを取得します。
[-cells]	セルのリストを返します (デフォルト)。
[-data_pins]	レジスタのデータ ピンのリストを返します。
[-clock_pins]	レジスタのクロック ピンのリストを返します。
[-async_pins]	非同期プリセット/クリア ピンのリストを返します。
[-output_pins]	レジスタの出力ピンのリストを返します。
[-level_sensitive]	レベルで認識されるラッチのみを返します。
[-edge_triggered]	エッジでトリガーされるフリップフロップのみを返します。
[-no_hierarchy]	現在のインスタンスのみを検索します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[SDC](#)、[XDC](#)

説明

現在のデザインに含まれるシーケンシャル レジスタ セルまたはレジスタ ピンのリストを返します。



ヒント: DSP および BRAM には内部レジスタが含まれるので、返されるオブジェクトには DSP および BRAM が含まれます。

返されるオブジェクトは、下記のオプションを使用することにより制限できます。指定のクロックに制限したり、指定のクロックの立ち上がりエッジまたは立ち下がりエッジでトリガーされるレジスタに制限したりすることが可能です。

`all_registers` で返されたレジスタのリストに対して `filter` コマンドを使用すると、レジスタのプロパティに基づいてリストをフィルターできます。オブジェクトのプロパティは、`list_property` または `report_property` コマンドを使用して返すことができます。

また、ピン オプションのいずれかを使用することにより、レジスタ オブジェクトではなくレジスタ ピンを取得できます。

引数

`-clock <args>` (オプション): 指定したクロックのファンアウトにクロック ピンが含まれるレジスタすべてのリストを返します。

`-rise_clock <args>` (オプション): 指定したクロックの立ち上がりエッジでトリガーされるレジスタのリストを返します。

`-fall_clock <args>` (オプション): 指定したクロックの立ち下がりエッジでトリガーされるレジスタのリストを返します。

注記: `-clock`、`-rise_clock`、および `-fall_clock` を同じコマンドで同時に使用しないでください。

`-cells` (オプション): ピン オブジェクトではなくレジスタ セル オブジェクトを返します。これがデフォルトの動作です。

`-data_pins` (オプション): デザインに含まれるすべてのレジスタまたは検索条件を満たすレジスタのデータ ピンのリストを返します。

`-clock_pins` (オプション): 検索条件を満たすレジスタのクロック ピンのリストを返します。

`-async_pins` (オプション): 検索条件を満たすレジスタの非同期ピンを返します。

`-output_pins` (オプション): 検索条件を満たすレジスタの出力ピンのリストを返します。

注記: `--*_pins` オプションは個別に使用してください。複数のオプションを同時に使用すると、1 つのオプションのみが `-data_pins`、`-clock_pins`、`-async_pins`、`-output_pins` の優先順位で使用されます。

`-level_sensitive` (オプション): レベルで認識されるレジスタまたはラッチのリストを返します。

`-edge_triggered` (オプション): エッジでトリガーされるレジスタまたはフリップフロップのリストを返します。

`-no_hierarchy` (オプション): デザインの階層を検索しません。 `current_instance` のレベルのみが検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインの任意のクロックの立ち下がりエッジでトリガーされるレジスタのリストが返されます。

```
all_registers -fall_clock [all_clocks]
```

次の例では、最小遅延を設定しています。

```
set_min_delay 2.0 -to [all_registers -clock CCLK -data_pins]
```

次の例では、clk_A 上のレジスタで名前に「meta」が含まれるものが返されます。

```
filter [all_registers -clock clk_A] {name =~ *meta*}
```

関連項目

- [all_clocks](#)
- [current_instance](#)
- [filter](#)
- [list_property](#)
- [report_property](#)
- [set_min_delay](#)

apply_bd_automation

IP インテグレーター オブジェクトに対してオートメーション ルールを実行します。

構文

```
apply_bd_automation -rule <arg> [-config <args>] -dict <arg> -opts <arg>
[-quiet] [-verbose] <objects>...
```

戻り値

実行結果 (正常に完了したか、エラーが発生したか)

使用法

名前	説明
-rule	ルール ID 文字列を指定します。
[-config]	パラメーターと値のペアを指定します。
-dict	オブジェクトと、対応するパラメーター名および値のペアのリストを指定します。
-opts	指定したルールのすべてのオブジェクトに適用する設定のリストを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	自動化ルールを実行するオブジェクトを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーターには、`apply_bd_automation` コマンドを使用した設計アシスタンス機能が含まれており、選択した IP インテグレーター オブジェクトの周辺にあるほかの関連の IP インテグレーター セルを自動的に設定したり追加したりできます。設計アシスタンス機能の詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896) または『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』(UG994) を参照してください。

現在のところ、セル、インターフェイス、ピン、ポートに対してブロック オートメーションおよびコネクション オートメーションがあります。ブロック オートメーション機能は、Zynq デバイス、MicroBlaze プロセッサ、AXI イーサネット、メモリ IP などの複雑なブロックに対して使用できます。

コネクション オートメーション機能は、異なるタイプの接続を自動化します。たとえば、スレーブ AXI-MM インターフェイスを接続する際、オートメーションにより関連のクロックとリセット ピンが接続され、必要な場合はインターコネクトが作成されます。また、ボード レベルの接続では、関連のセルからのピンおよびインターフェイスを外部ポートおよびインターフェイスに接続し、これらの外部 I/O に適切なボード制約を適用します。

注記: この IP インテグレーター コマンドは、Vivado IDE の設計アシスタンス GUI から実行します。この機能は、Tcl スクリプトから直接使用するのではなく、Vivado IDE の IP インテグレーターから使用することをお勧めします。ユーザー スクリプトで使用するには、write_bd_tcl コマンドを使用して Tcl を出力します。

引数

-rule <arg> (必須): 選択したオブジェクトに使用する定義されたオートメーション ルールを指定します。

-config <args> (オプション): IP インテグレーター オブジェクトのコンフィギュレーション パラメーターとその値を指定します。パラメーター名 (param) はクォーテーションを使用せずに指定し、値にはクォーテーションを付けて param と区別します。パラメーターと値はペアで指定します (param "value")。指定の <objects> に対して複数の param "value" ペアを指定するには、波かっこ ({}) で囲みます。

```
-config {local_mem "16KB" ecc "Basic" debug_module "Debug Only" }
```

-dict (オプション): ブロック デザイン オブジェクトと各オブジェクトのコンフィギュレーション パラメーター (<object> と <params> のペアで指定) のディクショナリを指定します。

```
apply_bd_automation -rule <ruleID> \
  -dict "[get_bd_intf_net /intf_net0] { DATA_SEL "1" TRIG_SEL "2" } \
    [get_bd_intf_net /intf_net1] { DATA_SEL "2" } \
    [get_bd_net /net1] {WIDTH "32" TYPE "1" } "
```



ヒント: 複数の値を指定する場合は、<objects> と -config オプションの代わりに -dict オプションを使用します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<objects> (必須): オートメーションを適用する IP インテグレーター オブジェクトを指定します。オブジェクトは、get_bd_cells、get_bd_pins、get_bd_interface などのコマンドで取得します。名前で参照することはできません。1 つの IP インテグレーター オブジェクトのみを指定し、-rule および -config オプションと互換している必要があります。

例

次の例では、ブロック オートメーションを MicroBlaze セルに指定したルールで適用し、指定のパラメーターを指定の値に設定します。

```
apply_bd_automation -rule xilinx.com:bd_rule:microblaze \
  -config {local_mem "16KB" ecc "Basic" debug_module "Debug Only" \
    axi_periph "1" axi_intc "1" clk "New Clocking Wizard (100 MHz)" } \
  [get_bd_cells /microblaze_1]
```

次の例では、コネクション オートメーション機能により、ボードに既知の互換インターフェイスがある場合にボードルールを IP サブシステムのピンまたはインターフェイス (クロック インターフェイス オブジェクト) に適用します。最初の `get_board_interfaces` コマンドは、IP オブジェクトと互換性のあるターゲット ボード上のインターフェイスを返します。2 番目の `apply_bd_automation` コマンドは、クロック インターフェイスを選択したボード インターフェイスに接続します。

```
get_board_interfaces -filter "VLNV==[get_property VLNV \
    [get_bd_intf_pins clk_wiz_1/CLK_IN1_D]]" \
sys_diff_clock

apply_bd_automation -rule xilinx.com:bd_rule:board \
    -config {Board_Interface "sys_diff_clock" } \
    [get_bd_intf_pins /clk_wiz_1/CLK_IN1_D]
```

次の例では、ターゲット ボードにクロック インターフェイスがない場合に、カスタム ボード ルールを IP サブシステムのクロック インターフェイス オブジェクト `CLK_IN1_D` に適用します。

```
apply_bd_automation -rule xilinx.com:bd_rule:board \
    [get_bd_pins /clk_wiz_2/CLK_IN1_D]
```

次の例では、ボードにリセット インターフェイスがない場合に、カスタム ボード ルールを IP サブシステムのリセット ピン `ext_reset_in` に適用します。

```
apply_bd_automation -rule xilinx.com:bd_rule:board \
    -config {rst_polarity "ACTIVE_HIGH" } \
    [get_bd_pins /proc_sys_reset_1/ext_reset_in]
```

関連項目

- [create_bd_cell](#)
- [create_bd_design](#)
- [get_bd_cells](#)
- [get_bd_intf_pins](#)
- [get_bd_pins](#)
- [write_bd_tcl](#)

apply_board_connection

デザインにボード接続を適用します。

構文

```
apply_board_connection [-board_interface <arg>] -ip_intf <arg>  
                        -diagram <arg> [-quiet] [-verbose]
```

戻り値

コマンドが正常に実行されたか、エラーが発生したか。

使用法

名前	説明
<code>[-board_interface]</code>	接続を適用する必要があるボード インターフェイスの名前を指定します。
<code>-ip_intf</code>	ボード オートメーションを適用する必要がある IP インターフェイスの完全パスを指定します。
<code>-diagram</code>	IP インテグレーター デザイン名を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[Board \(ボード\)](#)、[IPIntegrator \(IP インテグレーター\)](#)

説明

指定のブロック デザインの IP コアのインターフェイス ピンを、現在のプロジェクトまたはデザインの現在のボード パーツのインターフェイスに接続します。

ボード パーツは、ボード レベル システム内でのサイリンクス デバイスを表しており、クロック制約、I/O ポートの割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。ボード パーツ ファイルにボード 属性に関する情報が格納されています。このファイルは `board_part.xml` という名前で、Vivado Design Suite インストール エリアの `data/boards/board_parts` フォルダにあります。

このコマンドを使用すると、IP インテグレーター ブロック デザインの互換性のあるインターフェイス ピンを、現在のボード パーツ上の適切なインターフェイス 定義にすばやく接続できます。IP コアをボード パーツに接続するため、IP インテグレーターによりブロック デザインに外部インターフェイス ポートとインターフェイス接続が追加されます。追加された外部インターフェイス ポートには、指定のボード パーツ インターフェイスに対応する名前が付けられます。

`apply_board_connection` コマンドは、プロジェクトで定義されている現在のボード パーツで使用可能なインターフェイスを使用します。プロジェクトでターゲット ボードではなくターゲット パーツが使用されていると、エラーが返されます。`current_board_part` コマンドを使用するとプロジェクトで使用されているターゲット ボードを取得でき、`get_board_parts` コマンドを使用するとプロジェクトで使用可能なボードのリストを取得できます。`get_board_part_interfaces` コマンドを使用すると、現在のボードで使用可能なインターフェイスのリストを取得できます。

既存の IP インターフェイス接続を削除するには、`-board_interface` オプションを指定せずに `-ip_intf` オプションを指定します。ボード パーツ インターフェイスを指定しない場合、IP インターフェイス ピンの接続は削除されます。

このコマンドを実行すると、実行されたアクションが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-board_interface <arg>` (オプション): 指定の IP コア インターフェイスに接続するインターフェイス定義を指定します。`-board_interface` オプションを指定しない場合、または空の文字列値 ("") を指定した場合、IP インターフェイス ピン上の既存の接続は削除されます。

`-ip_intf <arg>` (必須): `-diagram` オプションで定義された IP インテグレーター ブロック デザインに含まれる IP コアのインターフェイス ピンを指定します。IP インターフェイス ピンは、「<IP_core_name>/<interface_pin_name>」の形式で指定します。下の最初の例では、IP コアのインスタンス名は `/proc_sys_reset_0`、インターフェイス ピン名は `/ext_reset` で、例に示すように指定します。

`-diagram <arg>` (必須): IP コア インスタンスが存在する IP インテグレーター ブロック デザインの名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のボード パーツで定義されている SGMII インターフェイスをプロセッサ リセット IP コアに接続しています。

```
apply_board_connection -board_interface "reset" \
    -ip_intf "/proc_sys_reset_0/ext_reset" -diagram "design_2"
```

次の例では、`-board_interface` オプションで空の文字列が指定されているので、指定のインターフェイス ピンの接続が削除されます。

```
apply_board_connection -board_interface "" \
    -ip_intf "/proc_sys_reset_0/ext_reset" -diagram "design_2"
```

関連項目

- [create_port](#)
- [current_board_part](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [write_checkpoint](#)
- [write_edif](#)

apply_hw_ila_trigger

スタートアップ時にデザインの ILA コアにトリガー初期値を適用します。

構文

```
apply_hw_ila_trigger [-ila_cell <arg>] [-quiet] [-verbose] [<file>]
```

使用法

名前	説明
<code>[-ila_cell]</code>	指定した ILA セルにトリガー設定を適用します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<file>]</code>	ILA スタートアップトリガー設定ファイルを指定します。

カテゴリ

Debug (デバッグ)

説明

スタートアップでの ILA トリガーをサポートするため、トリガー構成ファイルをデザインのビットストリームに適用します。

このコマンドは、デザインのビットストリーム ファイル (.bit) に ILA コアのトリガー設定を指定し、デバイスのコンフィギュレーションおよびスタートアップ直後に ILA デバッグ コアでトリガー イベントを検出できるようにします。これにより、デバイス アクティビティの最初の段階からデータをキャプチャできるようになります。これは、Vivado Design Suite のハードウェア マネージャーまたは `run_hw_ila` コマンドを使用する場合には不可能です。

`apply_hw_ila_trigger` コマンドは、`run_hw_ila -file` コマンドを使用して記述したトリガー構成ファイルを読み込み、インプリメント済みデザインの ILA コアにさまざまなトリガー設定を適用します。ILA コアのトリガー構成は `write_bitstream` コマンドで記述されるビットストリームの一部となり、このビットストリームがザイリンクス FPGA デバイスをプログラムするのに使用されます。

スタートアップ時にトリガー機能を使用するには、次の手順を実行します。

1. ハードウェア マネージャーから、`run_hw_ila -file` コマンドを使用して ILA コアのトリガー レジスタ マップ ファイルをエクスポートします。
2. インプリメント済みデザインまたはインプリメント済みデザイン チェックポイントを開きます。
3. `apply_hw_ila_trigger` コマンドを使用して、メモリ内のデザインにトリガー設定を適用します。
4. `write_bitstream` コマンドを使用して、適用されたトリガー構成ファイルを含むビットストリームを記述します。

注記: Vivado IDE の Flow Navigator コマンドではなく、必ず `write_bitstream` コマンドを使用してください。

5. ハードウェア マネージャーに戻り、`program_hw_device` コマンドで新しいビットストリーム ファイルを使用してハードウェア デバイスをプログラムします。

プログラムすると、新しい ILA コアはスタートアップ時にすぐにトリガー待機状態になります。Vivado ロジック解析機能では、トリガー イベントまたはキャプチャ条件が発生すると、ILA コアの [Trigger Capture Status] にキャプチャされたデータ サンプルが表示されます。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

引数

`-ila_cell <arg>` (オプション): 指定した ILA コアにトリガー設定を適用します。セルは、`get_cells` コマンドを使用して指定する必要があります。

注記: トリガー構成ファイルには、適用される ILA セルのインスタンス情報が含まれます。`-ila_cell` オプションは必須ではなく、デザインの異なる ILA セルにトリガー ファイルを適用する場合にのみ使用してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (オプション): `run_hw_ila -file` コマンドで作成したトリガー構成ファイルを指定します。

例

次の例では、指定のファイルで定義された ILA セル インスタンスにトリガー構成を適用しています。

```
apply_hw_ila_trigger C:/Data/ila1_triggers.tas
```

関連項目

- [current_hw_device](#)
- [current_hw_ila](#)
- [get_hw_ilas](#)
- [run_hw_ila](#)
- [write_bitstream](#)

archive_project

現在のプロジェクトのアーカイブを作成します。

構文

```
archive_project [-temp_dir <arg>] [-force] [-exclude_run_results]
               [-include_config_settings] [-include_runs_in_progress]
               [-include_local_ip_cache] [-quiet] [-verbose] [<file>]
```

戻り値

true

使用法

名前	説明
[-temp_dir]	アーカイブするプロジェクトのコピーを一時的に保存する場所を指定します。デフォルトは. です。
[-force]	既存のアーカイブ ファイルを上書きします。
[-exclude_run_results]	アーカイブに run の結果を含めません。
[-include_config_settings]	アーカイブに現在のプロジェクトの環境設定/ファイルを含めます。
[-include_runs_in_progress]	run が実行中でも run の結果を含めます。-exclude_run_results オプションを指定した場合はこのオプションは無視されます。
[-include_local_ip_cache]	アーカイブに IP キャッシュの結果を含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	アーカイブ ファイル名を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

プロジェクトのアーカイブを作成して、プロジェクトのバックアップとして保存したり、リモート サイトに送信したりするために使用します。デザイン階層を解析し、必要なソース ファイル、インクルード ファイル、リモート ファイルがライブラリ ディレクトリからコピーし、制約ファイルをコピーし、さまざまな合成、シミュレーション、インプリメンテーション run の結果をコピーしてから、プロジェクトの ZIP ファイルを作成します。



ヒント: [Design Run Settings] ダイアログ ボックスで合成およびインプリメンテーション段階に関連付けられている tcl.pre および tcl.post スクリプトをアーカイブするには、これらのスクリプト ファイルをデザイン ソースとしてプロジェクトに追加する必要があります。

プロジェクトをアーカイブする別の方法として、write_project_tcl コマンドを使用して、プロジェクトを現在の形で再作成する Tcl スクリプトを作成することもできます。

引数

`-temp_dir <arg>` (オプション): プロジェクトのアーカイブを作成するときにファイルをコピーする一時ディレクトリを指定します。一時ディレクトリが存在しない場合は作成され、アーカイブ プロセスが完了するとディレクトリ内のファイルは削除されます。デフォルトでは、現在の作業ディレクトリ内に一時ディレクトリが作成されます。

`-force` (オプション): 指定した名前 ZIP ファイルが存在する場合に上書きします。ZIP ファイルが存在する場合に `-force` が指定されていないと、エラー メッセージが表示されます。

`-exclude_run_results` (オプション): 合成またはインプリメンテーション `run` の結果を除外します。このコマンドにより、プロジェクト アーカイブのサイズを大幅に縮小できます。

`-include_config_settings` (オプション): 初期化 Tcl コマンド (`init.tcl`) をアーカイブ ファイルの `<project_name>/config_settings` フォルダに追加します。

`-include_runs_in_progress` (オプション): 完了していない `run` のディレクトリからのデータをアーカイブに含めます。 `-exclude_run_results` を指定した場合は無視されます。

`-include_local_ip_cache` (オプション): プロジェクト アーカイブにキャッシュされた IP 合成結果を含めます。キャッシュされた合成結果は、IP キャッシュがプロジェクトのローカルにある場合にのみ含まれます。詳細は、`config_ip_cache` コマンドを参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (オプション): `archive_project` コマンドで作成する ZIP ファイルの名前を指定します。 `<file>` を指定しない場合、プロジェクトと同じ名前の ZIP ファイルが作成されます。

例

次の例では、現在のプロジェクトのアーカイブが作成されます。

```
archive_project
```

注記: ファイル名が指定されていないので、プロジェクト アーカイブの名前は `<project_name>.zip` となります。

次の例では、現在のプロジェクトとして `project_3` を指定し、`proj3.zip` というプロジェクトのアーカイブを作成しています。

```
current_project project_3
archive_project -force -exclude_run_results proj3.zip
```

注記: `-force` を使用しているので、`proj3.zip` ファイルが既に存在している場合は上書きされます。 `-exclude_run_results` を使用しているので、合成 `run` またはインプリメンテーション `run` の結果はアーカイブに含まれません。プロジェクトで定義されたさまざまな `run` は含まれますが、その結果は含まれません。

次の例では、現在のプロジェクトを指定のファイルにアーカイブしています。必要に応じて既存のファイルが上書きされ、run の結果が含まれず、Vivado ツールを起動したときに使用されていた環境設定が含まれます。

```
archive_project -force mbl_archive.zip -temp_dir C:/Data/Temp \  
-exclude_run_results -include_config_settings
```

関連項目

- [current_project](#)
- [write_project_tcl](#)

assign_bd_address

マップされていない IP に自動的にアドレスを割り当てます。

構文

```
assign_bd_address [-target_address_space <arg>] [-boundary]
                  [-master_boundary] [-external] -dict <arg> [-import_from_file <arg>]
                  [-export_to_file <arg>] [-export_gui_to_file <arg>] [-offset <arg>]
                  [-range <arg>] [-base_high <arg>] [-force] [-quiet] [-verbose]
                  [<objects>...]
```

戻り値

新しくマップされたセグメント、エラーが発生した場合は ""。

使用法

名前	説明
[-target_address_space]	セグメントを配置するターゲット アドレス空間を指定します。
[-boundary]	ペリフェラルをデザインのエクスポートされたスレーブの階層境界に割り当てます。
[-master_boundary]	エクスポートされたマスター インターフェイス セグメントを内部マスターに割り当てることにより、階層マスターの境界を設定します。
[-external]	外部マスター インターフェイスを複数のアドレスにマップできるようにします。
-dict	外部インターフェイスを複数のアドレスにマップする際に、オフセット範囲アドレス ペア (例: {offset 0x00000000 range 32K offset 0x20000000 range 32K}) のディクショナリを指定します。
[-import_from_file]	ファイルからアドレス指定をインポートします。拡張子が csv の場合、フォーマットは <address space name>,<slave segment>,<offset>,<range> です。
[-export_to_file]	アドレス マップを CSV フォーマットにエクスポートします。フォーマットは <address space name>,<slave segment>,<offset>,<range> です。
[-export_gui_to_file]	アドレス マップを CSV フォーマットのファイルにエクスポートします。構造は [Address Editor] ウィンドウと同じです。
[-offset]	割り当てのオフセット (0x00000000 など) を指定します。
[-range]	割り当ての範囲 (4096、4K、16M、1G など) を指定します。
[-base_high]	範囲割り当てのベース オフセットとハイ オフセットをコロンで区切って指定します (例: 0x0000:0xFFFF)。
[-force]	割り当てを強制的に適用します。有効チェックは実行されません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	割り当てるオブジェクトを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

マップされていない IP アドレス セグメントを IP インテグレーター サブシステム デザインのアドレス空間に割り当てます。

ターゲット アドレス空間が指定されていない場合、IP インテグレーターによりアドレス セグメントが接続されている AXI マスターの使用可能なアドレス空間に自動的に割り当てられます。

ボード アドレス セグメント (bd_addr_seg) オブジェクトが指定されていない場合、assign_bd_address コマンドによりマップされていないすべてのアドレス セグメントが接続されている任意の AXI マスターのアドレス空間に割り当てられます。

このコマンドを実行すると、新しくマップされたアドレス セグメントが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-target_address_space <arg> (オプション): セグメントを配置するターゲット アドレス空間を指定します。



ヒント: ターゲット アドレス空間が指定されていない場合、Vivado IP インテグレーターによりアドレス セグメントが使用可能なアドレス空間に自動的に割り当てられます。

-boundary (オプション): アドレス空間をデザインの階層境界に割り当てます。このオプションを使用すると、ブロック デザイン外に存在するメモリのアドレス空間を割り当てることができます。指定の外部スレーブ インターフェイスを 0x000_0000 で開始する連続オフセットのアドレス空間にマップします。

-master_boundary (オプション): 指定の外部マスター インターフェイスに対して、スレーブ セグメントを -offset オプションで指定されたオフセットと -range オプションで指定された範囲の図の内部マスターにマップします。

-external (オプション): 以前に定義された外部アドレス空間のメモリ セグメントを定義します。BD 外のスレーブ IP に接続された外部 AXI ポートに複数のメモリ セグメントを割り当てることができます。

-import_from_file <arg> (オプション): メモリ マップを含む CSV フォーマット ファイルからアドレス空間割り当てを読み出します。CSV ファイルのフォーマットは、<address space name>,<slave segment>,<offset>,<range> である必要があります。このコマンドは、CSV ファイルに有効なアドレス マップが含まれていれば、既存のアドレス空間定義を上書きします。そうでない場合はエラーが返されます。

-export_to_file <arg> (オプション): アドレス空間割り当てを指定した CSV フォーマット ファイルに書き込みます。CSV ファイルは、<address space name>,<slave segment>,<offset>,<range> という形式である必要があります。同じ名前のファイルが存在する場合、-force を使用していなければ上書きされません。

-force (オプション): -export_to_file オプションを使用しているときに、指定した CSV ファイルが存在する場合に上書きします。

-base_high <arg> (オプション): アドレス空間の範囲を <base address>:<high address> という形式で指定します (例: -base_high 0x4000_0000:0x4000_FFFF)。

-offset <arg> (オプション): アドレス セグメントを割り当てるアドレス空間のメモリ オフセットを指定します (例: 0x00000000)。

`-range <arg>` (オプション): アドレス セグメントを割り当てるアドレス空間の範囲 (サイズ) を指定します (例: 4096、4K、16M、1G)。

`-dict <arg>` (オプション): オフセット/範囲値のペアを一度に定義するディクショナリを指定します。オフセット/範囲値のペアのフォーマットは次のとおりです。

```
-dict {offset 0x00000000 range 64M offset 0x20000000 range 64M}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>`: ターゲット アドレス空間に割り当てるブロック デザインのアドレス セグメント オブジェクトを指定します。

例

次の例では、指定のアドレス セグメント オブジェクトをアドレス空間に自動的に割り当てています。この例ではターゲット空間が指定されていないので、IP インテグレーターによりアドレス セグメントが使用可能なアドレス空間に自動的に割り当てられます。

```
assign_bd_address [get_bd_addr_segs \
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB/Mem ]]
```

次の例では、外部 AXI ポートにメモリ空間を作成し、その外部アドレス空間内に、指定のブロック デザイン アドレス セグメントを指定のオフセットで 2 つのアドレス範囲に割り当てています。

```
assign_bd_address -offset 0xA0000000 -range 64K [get_bd_addr_segs {/
M02_AXI_0/Reg}]
assign_bd_address -external -offset 0xA0000000 -range 4K \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
assign_bd_address -external -offset 0xA0001000 -range 4K \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
```

次の例は上記の例と同様ですが、`-dict` オプションを使用して 1 つの `assign_bd_address` コマンドで複数のオフセット/範囲値のペアを指定しています。

```
assign_bd_address -offset 0xA0000000 -range 64K [get_bd_addr_segs {/
M02_AXI_0/Reg}]
assign_bd_address -external -dict {offset 0xA0000000 range 4K offset
0xA0001000 range 4K} \
[get_bd_addr_segs /M02_AXI_0/Seg] -target_address_space [get_bd_addr_space /
M02_AXI_0/Reg]
```

関連項目

- [create_bd_addr_seg](#)

- [exclude_bd_addr_seg](#)
- [get_bd_addr_segs](#)
- [get_bd_addr_spaces](#)
- [include_bd_addr_seg](#)

auto_detect_xpm

デザインで使用されている XPM ライブラリを自動検出し、XPM_LIBRARIES プロジェクト プロパティを設定します。

構文

```
auto_detect_xpm [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Project \(プロジェクト\)](#)

説明

RTL デザイン ファイルで使用されているサイリンクス パラメーター指定マクロ (XPM) のリストを取得し、XPM_LIBRARIES プロパティを設定します。RTL ソース ファイルで使用されている XPM を特定するので、例に示すように、RTL ファイルを読み込んだ後、またはデザインに読み込んだ後に使用する必要があります。



ヒント: このコマンドは非プロジェクト モードで使用するためのもので、プロジェクト ベース フローでは不要です。

このコマンドを実行すると、RTL で使用されている XPM がすべて検索され、現在のプロジェクトの XPM_LIBRARIES プロパティがアップデートされます。合成およびシミュレーションで XPM_LIBRARIES プロパティが使用され、これらのマクロがサポートされます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定した RTL ファイルを読み込み、`auto_detect_xpm` コマンドを使用してファイルで使用されている XPM_LIBRARIES を定義しています。

```
read_verilog ../top_bgRAM_173_0.v
read_xdc ../top_bgRAM_173_0.xdc
auto_detect_xpm
```

関連項目

- [read_verilog](#)
- [read_vhdl](#)

boot_hw_device

hw_device に対して JTAG プログラム コマンドを発行します。

構文

```
boot_hw_device [-disable_done_check] [-timeout <arg>] [-quiet] [-verbose]
               <hw_device>
```

使用法

名前	説明
<code>[-disable_done_check]</code>	ブート デバイスの DONE チェックをディスエーブルにします。
<code>[-timeout]</code>	ブートのタイムアウト時間を秒数で指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_device></code>	ターゲット hw_device 接続を指定します。

カテゴリ

Hardware (ハードウェア)

説明

ハードウェア デバイス (hw_device) (FPGA) に対して JTAG プログラム コマンドを発行します。

このコマンドは、FPGA ブートおよびボード スタートアップ シーケンスをトリガーします。ブート シーケンスは、FPGA コンフィギュレーション プロセスを開始して、モード ピンの設定に応じてデバイスの以前のプログラムをクリアし、新しいプログラムを読み込みます。

モード ピン設定に応じて hw_device が起動します。デバイス上の FPGA のモード ピンが JTAG モードに設定されている場合、またはインターフェイスが非アクティブの場合 (PROM がコンフィギュレーションされていないなど) は、boot_hw_device コマンドにより以前のプログラムがクリアされます。

このコマンドを実行すると、DONE ピンが HIGH になった場合は 1 が返され、それ以外の場合は 0 が返されます。

引数

`-disable_done_check` (オプション): ブート デバイスの DONE チェックをディスエーブルにします。このオプションは、たとえば JTAG モードでデバイスを再プログラムせずにクリアする場合などに使用します。

`-timeout <arg>` (オプション): boot_hw_device コマンドによるデバイスのブートの試行を停止するタイムアウト時間を秒数で指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_device> (オプション): JTAG ブート コマンドを送信する hw_device オブジェクトを指定します。hw_device は、get_hw_devices または current_hw_device コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、メモリ デバイスを作成し、現在のハードウェア デバイス オブジェクト (current_hw_device) と関連付け、ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトのさまざまなプロパティを割り当てています。その後、write_cfgmem コマンドを使用してデザイン ビットストリームをフォーマットし、コンフィギュレーション メモリ ファイル (.mcs) を hw_cfgmem オブジェクトの PROGRAM.FILE プロパティに関連付けます。そして、hw_cfgmem オブジェクトを指定のオフセット アドレスから MCS ファイルのデータでプログラムします。プログラム後、current_hw_device に対して boot_hw_device コマンドを発行し、DONE ピンをチェックして、適切なメッセージを返します。

```
set memObj [create_hw_cfgmem -hw_device [current_hw_device] \
    [lindex [get_cfgmem_parts {28f00ap30t-bpi-x16}] 0]]
set_property PROGRAM.BLANK_CHECK 1 $memObj
set_property PROGRAM.ERASE 1 $memObj
set_property PROGRAM.CFG_PROGRAM 1 $memObj
set_property PROGRAM.VERIFY 1 $memObj
write_cfgmem -force -format MCS -size 64 -interface BPIx16 \
    -loadbit "up 0x0 ./project_netlist.runs/impl_1/sinegen_demo.bit" \
    ./config_28f00ap30t
set_property PROGRAM.FILE ./config_28f00ap30t.mcs $memObj
program_hw_cfgmem -hw_cfgmem $memObj
if {[boot_hw_device [current_hw_device]] == 1} {
    puts stderr "DONE signal is HIGH"
} else {
    puts stderr "DONE signal is LOW"
}
```

関連項目

- [connect_hw_server](#)
- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [program_hw_cfgmem](#)

calc_config_time

デバイスのコンフィギュレーション時間 (ms) を算出します。

構文

```
calc_config_time [-verbose] [-max] [-min] [-typical] [-por_used]
                 [-por_ramp <arg>] [-clk_freq <arg>] [-bitstream_size <arg>] [-quiet]
```

戻り値

レポート

使用法

名前	説明
[-verbose]	算出パラメーターを出力します。
[-max]	最長コンフィギュレーション時間を算出します。
[-min]	最短コンフィギュレーション時間を算出します。
[-typical]	標準コンフィギュレーション時間を算出します。
[-por_used]	(廃止予定) -por_ramp を 0 以外の値に設定してパワーオン リセット (POR) を使用するかを指定します。
[-por_ramp]	パワーオン リセット (POR) のランプ レートを 1 ms ~ 50 ms の間で指定します。デフォルトは 0 ms です。
[-clk_freq]	スレーブ モードの場合およびマスター モードで外部マスター クロックを使用する場合のクロック周波数を MHz で指定します。デフォルトは 0 MHz です。
[-bitstream_size]	ビットストリーム サイズを指定します。デフォルトは 0 です。
[-quiet]	コマンド エラーを表示しません。

カテゴリ

[Report \(レポート\)](#)

説明

現在のデザインでザイリンクス デバイスをコンフィギュレーションするのにかかる時間 (ms) を見積もります。



ヒント: このコマンドを実行するには、デバイス コンフィギュレーション モードを指定する必要があります。

アプリケーションによっては、ザイリンクス デバイスが短時間でコンフィギュレーションされ、動作を開始することが必要です。このコマンドを使用すると、特定のデバイスとデザインのコンフィギュレーション時間を見積もることができます。コンフィギュレーション時間には、デバイスの初期化時間とコンフィギュレーション時間が含まれます。コンフィギュレーション時間は、デバイスのサイズとコンフィギュレーション ロジックの速度によって異なります。コンフィギュレーション時間の詳細は、『UltraFast 設計手法ガイド (Vivado Design Suite 用)』 (UG949)、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』 (UG570)、または『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』 (UG470) を参照してください。

コンフィギュレーション時間の算出に必要な設定は、プロパティ (BITSTREAM.CONFIG.CONFIGRATE、BITSTREAM.CONFIG.EXTMASTERCLK_EN など) として現在のデザインに保存されています。一部のマスター モードでは、コンフィギュレーションを制御するコンフィギュレーション クロックは FPGA 上にあり、公称コンフィギュレーション クロック周波数が BITSTREAM.CONFIG.CONFIGRATE プロパティで指定されます。プロパティに値を設定するには、Vivado Design Suite IDE の [Edit Device Properties] ダイアログ ボックスまたは `set_property` を使用します。

スレーブ コンフィギュレーション モードまたは外部マスター クロックを使用するコンフィギュレーション モードでは、必要なクロック周波数を `-clk_freq` オプションで指定します。

このコマンドが正常に実行された場合は見積もり値が ms で返され、正常に実行されなかった場合はエラーが返されます。

引数

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`-max` (オプション): コマンドで見積もられる最大コンフィギュレーション時間をレポートします。

`-min` (オプション): コマンドで見積もられる最小コンフィギュレーション時間をレポートします。

`-typical` (オプション): コマンドで見積もられる典型的なコンフィギュレーション時間をレポートします。

`-por_ramp <arg>` (オプション): パワーオン リセット (POR) のランプ レートを 1 ~ 50 ms の間で指定します。デフォルトは 0 です。システムのランプ レートを制御することにより、パワーオン リセット時間 (Tpor) の削減できます。Tpor ランプ オプションの仕様は、使用しているデバイスのデータシートの FPGA コンフィギュレーション スイッチ 特性を参照してください。

`-clk_freq <arg>` (オプション): スレーブ モードまたは外部マスター クロックを使用する場合のクロック周波数を MHz で指定します。デフォルトは 0 です。

注記: マスター コンフィギュレーション モードで実行する場合は、クロック周波数は BITSTREAM.CONFIG.CONFIGRATE プロパティで指定します。

`-bitstream_size <arg>` (オプション): ビットストリームのサイズをビット数で指定します。デフォルトは 0 です。



ヒント: ビットストリームのサイズは、現在のデザインに関連付けられているビットストリームから自動的に算出されます。このオプションを指定すると、自動的に算出されたサイズの代わりにこのオプションで指定したサイズが使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

例

次の例では、現在のターゲット パーツの指定された外部クロック周波数での最大コンフィギュレーション時間を算出します。

```
calc_config_time -max -clk_freq 50
```

関連項目

- [program_hw_cfgmem](#)
- [set_property](#)
- [write_bitstream](#)

can_resolve_reference

モジュールが参照可能かをチェックします。

構文

```
can_resolve_reference [-quiet] [-verbose] <module>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><module></code>	モジュール名を指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)

説明

このコマンドは、モジュールをデザインにインポートする前に、モジュールへの参照を検証します。これは主に `write_bd_tcl` コマンドで生成されるスクリプトなどのスクリプトで使用されますが、独自のスクリプトでも使用できます。

参照を解決できなくなった場合は 0 が返され、解決できた場合は 1 が返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<module>` (必須): 検証するモジュールの名前を指定します。モジュール名は、読み込まれた RTL デザイン ソース ファイルから参照されます。

例

次の例では、指定したモジュールへの参照を解決できるかどうかを検証しています。

```
can_resolve_reference clk_div
```

関連項目

- [write_bd_tcl](#)
- [write_project_tcl](#)

check_syntax

指定したファイルセットまたはアクティブ ファイルセットの HDL 構文をチェックします。

構文

```
check_syntax [-fileset <arg>] [-return_string] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-fileset]</code>	構文をチェックするファイルセットを指定します。
<code>[-return_string]</code>	構文チェック メッセージを文字列として返します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

Verilog、SystemVerilog、および VHDL ソース ファイルを解析し、デザインの構文に関する警告およびエラー メッセージを生成します。



ヒント: 構文は、Vivado テキスト エディターでファイルを編集しているとき、およびファイルを保存するときにも自動的にチェックされます。

このコマンドを実行すると、チェックされたファイルに関する警告またはエラーが返されるか、問題が検出されなかった場合は何も返されません。

引数

`-fileset <arg>` (オプション): ファイルの構文をチェックするファイルセットを指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、シミュレーション ファイルセットに含まれるファイルの構文をチェックしています。

```
check_syntax -fileset sim_1
```

check_timing

デザインで発生する可能性のあるタイミング問題をチェックします。

構文

```
check_timing [-file <arg>] [-no_header] [-loop_limit <arg>] [-append]
             [-name <arg>] [-override_defaults <args>] [-include <args>]
             [-exclude <args>] [-return_string] [-rpx <arg>] [-cells <args>]
             [-verbose] [-quiet]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-no_header]	ヘッダーなしのレポートを生成します。
[-loop_limit]	loops チェックでレポートするループの最大数を指定します。デフォルトは 100 です。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-override_defaults]	デフォルトのタイミング チェックではなく、指定したチェックを実行します。
[-include]	デフォルトのタイミング チェックに加えて実行するチェックを指定します。
[-exclude]	デフォルトのタイミング チェックから指定のチェックを除外します。
[-return_string]	レポートを文字列として返します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-cells]	check_timing コマンドを指定のセルに対して実行します。
[-verbose]	検出されたタイミング問題すべての詳細を示します。
[-quiet]	コマンド エラーを表示しません。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

ポート、ピン、およびパスのデザイン エレメントを現在のタイミング制約に対してチェックします。このコマンドは report_timing コマンドを実行する前にデザイン データおよびタイミング制約に問題がないかどうかを確認するために使用します。check_timing コマンドはデフォルト タイミング チェックを実行し、検出された違反のサマリをレポートします。違反に関する詳細を取得するには、-verbose オプションを使用します。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、-name オプションを使用して GUI に表示したり、-file オプションを使用してファイルに書き込むこともできます。

デフォルトのタイミング チェックは、次のとおりです。

- `constant_clock`: 定数信号 (gnd、vss、data) に接続されているクロック信号を確認します。
- `generated_clocks`: 生成クロック ネットワーク内でループや循環定義がないかを確認します。生成クロックが別の生成クロックをソースとして生成されており、その2番目の生成クロックが最初の生成クロックをソースとして生成されている場合にエラーを返します。
- `latch_loops`: デザインに組み合わせラッチ ループがないことをチェックします。
- `loops`: デザインに組み合わせフィードバック ループがないことをチェックします。
- `multiple_clock`: レジスタ クロック ピンに複数のクロックが供給されていないことをチェックします。レジスタ クロック ピンに複数のクロックが供給されている場合、解析にどのクロックを使用するかが不明になります。その場合、`set_case_analysis` コマンドを使用してレジスタ クロック ピンに1つのクロックのみが伝搬されるようにしてください。
- `no_clock`: クロックが供給されていないレジスタをレポートします。レジスタにクロックが供給されていない場合、データ ピンでレジスタ クロック ピンに対するセットアップまたはホールド チェックが実行されません。
- `no_input_delay`: 入力遅延制約が設定されていない入力ポートをレポートします。入力遅延は、`set_input_delay` コマンドを使用して指定できます。クロックに同期しない入力ポートでは、入力遅延はチェックされません。
- `no_output_delay`: 出力遅延制約が設定されていない出力ポートをレポートします。出力遅延は、`set_output_delay` コマンドを使用して指定できます。クロックに同期しない出力ポートでは、出力遅延はチェックされません。
- `partial_input_delay`: 入力遅延制約が部分的に定義されている入力ポートをレポートします。入力ポートに `set_input_delay -max` または `set_input_delay -min` のいずれかのみを設定すると、部分的に定義された入力遅延が作成されます。この場合、入力ポートからのパスが制約されず、ポートのタイミングがチェックされないことがあります。`set_input_delay` を `-min` または `-max` を指定せずに設定した場合、ツールで最小遅延と最大遅延の両方が想定され、部分的に定義された入力遅延は作成されません。

注記: クロックが供給されていない入力ポートは、部分的に定義された入力遅延があるかどうかはチェックされません。

- `partial_output_delay`: 出力遅延制約が部分的に定義されている出力ポートをレポートします。出力ポートに `set_output_delay -max` または `set_output_delay -min` のいずれかのみを設定すると、部分的に定義された出力遅延が作成されます。この場合、ポートに到達するパスが制約されず、ポートのタイミングがチェックされないことがあります。`set_output_delay` を `-min` または `-max` を指定せずに設定した場合、ツールで最小遅延と最大遅延の両方が想定され、部分的に定義された出力遅延は作成されません。

注記: クロックが供給されていない出力ポートは、部分的に定義された出力遅延があるかどうかはチェックされません。

- `pulse_width_clock`: セットアップまたはホールド チェック、リカバリ チェック、リムーバル チェック、または `clk->Q` チェックが関連付けられておらず、パルス幅チェックのみが関連付けられているクロック ピンをレポートします。
- `unconstrained_internal_endpoints`: レジスタ データ ピンでの制約が設定されていないタイミング パスの終点をレポートします。終点がレジスタ データ ピンの場合、`create_clock` コマンドを使用して割り当てたクロックで制約されます。出力ポートでの終点は、`no_output_delay` チェックによりチェックおよびレポートされます。
- `unexpandable_clocks`: クロック間にパスが1つ以上ある場合に、共通周期なしのクロックのセットをレポートします。1000 サイクル以内にソース クロックとデスティネーション クロックの周期に公倍数が見つからない場合、共通周期なしとなります。

引数

`-file <arg>` (オプション): 結果をディスク上の指定のファイルに書き込みます。デフォルトでは、このコマンドの結果は Tcl コンソールに表示されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append`: 結果を指定したファイルの最後に追加します。デフォルトでは、`-file` オプションを指定すると、`check_timing` コマンドで既存のファイルが上書きされます。

`-no_header` (オプション): レポートに標準ヘッダーを含めません。これはブール値のオプションであり、使用するとイネーブルになります。

`-loop_limit <arg>` (オプション): `loop` および `latch_loop` チェックでレポートするループの数を指定します。`-loop_limit` に到達した後、`check_timing` コマンドはほかのチェックを実行します。

`-name <arg>` (オプション): GUI の [Timing] ウィンドウに表示する結果の名前を指定します。

`-override_defaults <args>` (オプション): デフォルトのタイミング チェックではなく、指定したチェックのみを実行します。



ヒント: 複数のチェックは、ダブル クォーテーション (") または波かっこ ({}) で囲む必要があります。

`-include <args>` (オプション): デフォルト チェックに加えて実行するチェックを実行します。

`-exclude <args>` (オプション): `check_timing` コマンドで実行されるデフォルト チェックから指定のチェックを除外します。デフォルト チェックから除外するチェックを指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-cells <arg>` (オプション): 指定した階層セルのタイミング チェックを実行します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): 実行されたチェックからより詳細な結果を返します。検出された問題の詳細を示します。

例

次の例では、デフォルトのタイミング チェックから指定のチェックが除外されて `check_timing` が実行されます。

```
check_timing -exclude {loops generated_clocks}
```

次の例では、multiple_clocks チェックのみを実行し、-verbose オプションで詳細な結果を表示しています。その後、get_clocks を使用して問題を調べています。

```
check_timing -verbose -override_defaults {multiple_clock}
Checking multiple_clock.
There are 2 register/latch pins with multiple clocks.
procEngine/mode_du/set_reg[0]/C
provEngine/mode_du/set_reg[1]/C
get_clocks -of_objects [get_pin procEngine/mode_du/set_reg[0]/C]
sysClk coreClk
```

関連項目

- [create_clock](#)
- [get_clocks](#)
- [open_report](#)
- [report_timing](#)
- [set_case_analysis](#)
- [set_input_delay](#)
- [set_max_delay](#)
- [set_output_delay](#)

checkpoint_vcd

VCD チェックポイントを作成します (Verilog \$dumpall システム タスクと同等)。

構文

```
checkpoint_vcd [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

`checkpoint_vcd` コマンドは、現在の HDL オブジェクト信号値を VCD (Value Change Dump) ファイルに挿入します。戻り値はありません。この Tcl コマンドは Verilog \$dumpall システム タスクと同等で、指定の信号の初期値を示します。

VCD は、ヘッダー情報、変数定義、HDL 信号値の変更の詳細を含む ASCII 形式のファイルです。VCD ファイルを使用すると、VCD ビューアーでシミュレーション結果を表示したり、デザインの消費電力見積もりを実行したりできます。VCD ファイル フォーマットの詳細は、Verilog ハードウェア記述言語の IEEE 規格 (IEEE Std 1364-2005) を参照してください。

`checkpoint_vcd` コマンドを実行する前に、`open_vcd` および `log_vcd` コマンドを実行する必要があります。`checkpoint_vcd` コマンドを実行したら、シミュレーションを実行して信号値をキャプチャします。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

`checkpoint_vcd` の次の例では、指定の HDL オブジェクトの信号値を開いている VCD ファイルに記述しています。

```
checkpoint_vcd
```

関連項目

- [flush_vcd](#)
- [log_vcd](#)
- [open_vcd](#)

close_bd_design

デザインを閉じます。

構文

```
close_bd_design [-quiet] [-verbose] <name>
```

戻り値

デザイン オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	閉じるデザインの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado Design Suite の IP インテグレーターの指定した IP サブシステムを閉じます。

デザインが変更されている場合でも、閉じる前にデザインを保存するかどうか尋ねるメッセージは表示されません。
`close_bd_design` コマンドを使用する前に、`save_bd_design` コマンドを実行して変更を保存しておく必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: 閉じる IP サブシステム デザイン オブジェクトの名前を指定します。

例

次の例では、現在のプロジェクトの現在の IP サブシステム デザインを閉じています。

```
close_bd_design [current_bd_design]
```

関連項目

- [create_bd_design](#)
- [current_bd_design](#)
- [get_bd_designs](#)
- [open_bd_design](#)
- [save_bd_design](#)

close_design

現在のデザインを閉じます。

構文

```
close_design [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在アクティブなデザインを閉じます。デザインが変更されている場合でも、閉じる前にデザインを保存するかどうか尋ねるメッセージは表示されません。close_design コマンドを使用する前に、save_design または save_design_as コマンドを実行して変更を保存しておく必要があります。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、現在のプロジェクトが閉じられます。

```
close_design
```

注記: 複数のデザインが開いている場合は、close_design コマンドを実行する前に current_design コマンドで現在のデザインを指定できます。

次の例では、現在のデザインを指定してから閉じています。

```
current_design rtl_1
close_design
```

`current_design` コマンドで `rtl_1` デザインが現在のデザインとして指定され、`close_design` コマンドで閉じられます。

関連項目

- [current_design](#)

close_hw_manager

ハードウェア ツールを閉じます。

構文

```
close_hw_manager [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite でハードウェア マネージャーを閉じます。

デザインをザイリンクス FPGA ハードウェアにプログラムしたりデバッグしたりするには、まず `open_hw_manager` コマンドを使用してハードウェア マネージャーを開きます。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、Vivado Design Suite でハードウェア マネージャーを閉じます。

```
close_hw_manager
```

関連項目

- [connect_hw_server](#)
- [open_hw_manager](#)

close_hw_target

ハードウェア ターゲットを閉じます。

構文

```
close_hw_target [-quiet] [-verbose] [<hw_target>]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_target>]</code>	ハードウェア ターゲットを指定します。デフォルトは、現在のハードウェア ターゲットです。

カテゴリ

Hardware (ハードウェア)

説明

`open_hw_target` コマンドを使用して開いた現在のハードウェア ターゲットまたは指定のハードウェア ターゲットへの接続を閉じます。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のザイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、`hw_server` アプリケーションで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

このコマンドを実行すると、ハードウェア サーバーからの接続メッセージが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_target> (オプション): 接続を閉じるハードウェア ターゲット (hw_target) オブジェクトを指定します。hw_target は、get_hw_targets または current_hw_target コマンドを使用してオブジェクトとして指定する必要があります。ターゲットを指定しない場合は、現在のハードウェア ターゲット (current_hw_target) への開いている接続が閉じられます。

例

次の例では、現在のハードウェア ターゲットを閉じています。

```
close_hw_target
```

関連項目

- [get_hw_targets](#)
- [open_hw_target](#)
- [refresh_hw_target](#)

close_project

現在開いているプロジェクトを閉じます。

構文

```
close_project [-delete] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-delete]</code>	プロジェクトをディスクからも削除します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Project (プロジェクト)

説明

現在開いているプロジェクトを閉じます。



ヒント: プロジェクト名前空間ではなくグローバル名前空間のユーザー定義 Tcl 変数は、このコマンドではリセットまたは消去されません。グローバル変数は Vivado の起動中は保持され、Vivado Design Suite を終了したときにのみ消去されます。特定の Tcl 変数を消去するには、`unset` コマンドを使用できます。

引数

`-delete` (オプション): プロジェクトを閉じた後、ハード ディスクからプロジェクト データを削除します。

注記: このオプションを使用する際には、注意が必要です。プロジェクト データの削除を確認するメッセージは表示されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、アクティブ プロジェクトが閉じられます。

```
close_project
```

次の例では、現在のプロジェクトが閉じられます。複数のプロジェクトが開いている場合は、現在のプロジェクトに対してのみ `close_project` コマンドが実行されます。現在のプロジェクトは、`current_project` コマンドで指定できます。

次の例では、`project_1` を現在のプロジェクトとして指定してから閉じ、コンピューターのハード ディスクから削除します。

```
current_project project_1  
close_project -delete
```

注記: `-delete` オプションを使用する際には、注意が必要です。プロジェクト データの削除を確認するメッセージは表示されません。

関連項目

- [current_project](#)

close_saif

SAIF トグル情報を SAIF 出力ファイルに保存し、ファイルを閉じます。

構文

```
close_saif [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

開いている SAIF ファイルを閉じます。

Vivado シミュレータでは、`open_saif` を使用して一度に開くことのできる SAIF ファイルは 1 つのみです。別の SAIF ファイルを開くには、現在開いている SAIF ファイルを閉じる必要があります。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次に例を示します。

```
close_saif
```

関連項目

- [log_saif](#)
- [open_saif](#)

close_sim

Vivado を終了せずに現在のシミュレーションをアップロードします。

構文

```
close_sim [-force] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-force]</code>	変更が失われる場合でも、強制的にシミュレーションを閉じます。デフォルトでは、変更がある場合は波形設定は閉じず、エラー メッセージが表示されます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の Vivado シミュレーションを閉じます。

注記: このコマンドでは、サードパーティ シミュレータを閉じることはできません。

引数

`-force` (オプション): 変更が失われる場合でも、強制的にシミュレーションを閉じます。閉じる前に保存するかどうかを確認するメッセージは表示されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、変更が失われる場合でも、強制的にシミュレーションを閉じています。

```
close_sim -force
```

関連項目

- [current_sim](#)

close_vcd

VCD 情報を VCD 出力ファイルに保存し、ファイルを閉じます。

構文

```
close_vcd [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

開いている VCD (Value Change Dump) ファイルを閉じます。

Vivado シミュレータでは、一度に開くことのできる VCD ファイルは 1 つのみです。別の VCD ファイルを開くには、現在開いている VCD ファイルを閉じる必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在の VCD オブジェクトを閉じています。

```
close_vcd
```

関連項目

- [open_vcd](#)

close_wave_config

波形設定を閉じます。

構文

```
close_wave_config [-force] [-quiet] [-verbose] [<wcfgobj>]
```

使用法

名前	説明
[-force]	変更が失われる場合でも、強制的に波形設定を閉じます。デフォルトでは、変更がある場合は波形設定は閉じず、エラーメッセージが表示されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<wcfgobj>]	指定の波形設定オブジェクトを閉じ、破棄します。指定しない場合は、現在の波形設定を閉じます。デフォルトは NULL です。

カテゴリ

[Waveform \(波形\)](#)

説明

現在の、または指定した波形設定を閉じます。

Vivado® シミュレータ GUI では、波形ウィンドウを使用してデザインを解析し、コードをデバッグできます。波形設定オブジェクトには最上位 HDL オブジェクトが表示されますが、`add_wave` および `add_wave_divider` などのコマンドを使用してオブジェクトを追加できます。`create_wave_config` コマンドを使用すると、現在のシミュレーションで新規波形設定オブジェクトを作成できます。

波形設定オブジェクトに加えた変更は、`save_wave_config` コマンドを使用して保存できます。保存した波形設定ファイルを開くには、`open_wave_config` コマンドを使用します。

引数

`-force` (オプション): 変更が失われる場合でも、強制的に波形設定ファイルを閉じます。デフォルトでは、変更がある場合は波形設定は閉じず、エラーメッセージが表示されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<wcfgobj> (オプション): 閉じる波形設定オブジェクトを指定します。デフォルトでは、`current_wave_config` コマンドで返される現在の波形設定ファイルが閉じます。

注記: 波形設定オブジェクトは、`get_wave_configs` コマンドを使用して指定する必要があります。

例

次の例では、現在のシミュレーションに関連付けられているすべての波形設定ファイルを閉じています。

```
close_wave_config [get_wave_configs]
```

関連項目

- [create_wave_config](#)
- [current_wave_config](#)
- [get_wave_configs](#)
- [open_wave_config](#)
- [open_wave_database](#)
- [save_wave_config](#)

commit_hw_hbm

現在のハードウェア オブジェクトのプロパティの変更を確定します。HBM またはデバイス ハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。

構文

```
commit_hw_hbm [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

Vivado ハードウェア マネージャーの指定の HBM コントローラーで定義されているプロパティの現在の値を、現在のハードウェア デバイスに確定します。

`commit_hw_hbm` コマンドは、ハードウェア HBM (`hw_hbm`) オブジェクトで定義されている現在のプロパティ値を取り込み、ハードウェア サーバーに接続されている現在のハードウェア デバイスに確定します。

`hw_hbm` オブジェクトのプロパティ 値を変更した場合、`commit_hw_hbm` コマンドを使用して確定するまで、プロパティ 値はハードウェア デバイスには書き込まれません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (必須): 実行する HBM アクティビティ モニター オブジェクト (`hw_hbm`) を指定します。

例

次の例では、指定した HBM コントローラーオブジェクトのプロパティを変更し、その変更をデバイスに確定しています。

```
set_property MC2.INIT.AM.REPEAT_EN 1 [get_hw_hbms *HBM_2]  
commit_hw_hbm [get_hw_hbm *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

commit_hw_mig

現在のハードウェア オブジェクトのプロパティの変更を確定します。任意の MIG、デバイス、ターゲット、またはサーバー ハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。

構文

```
commit_hw_mig [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクト

カテゴリ

Hardware (ハードウェア)

説明

Vivado Design Suite のハードウェア マネージャー機能で指定のメモリ IP デバッグ コア ハードウェア オブジェクトで定義されているプロパティの現在の値を、現在のハードウェア デバイスに確定します。

`commit_hw_mig` コマンドは、Vivado ロジック解析のハードウェア MIG (`hw_mig`) オブジェクトで定義されている現在のプロパティ値を取り込み、ハードウェア サーバーに接続されている現在のハードウェア デバイスに確定します。

`hw_mig` オブジェクトのプロパティ (`CONFIG.*` など) の値を変更した場合、`commit_hw_mig` コマンドを使用して確定するまで、プロパティ値はハードウェア デバイスには書き込まれません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (オプション): `hw_mig`、ハードウェア デバイス (`hw_device`)、ハードウェア ターゲット (`hw_target`)、またはハードウェア サーバー (`hw_server`) オブジェクトを指定します。オブジェクトを指定しない場合は、現在の `hw_device` がターゲットとなります。

注記: オブジェクトは、名前で指定するのではなく、`get_hw_sysmon` などの `get_hw_XXX` コマンドを使用して指定する必要があります。

例

次の例では、現在のハードウェア デバイス上にある `hw_mig` オブジェクトの現在のプロパティを現在のハードウェア デバイスに確定しています。

```
commit_hw_mig [lindex [get_hw_migs] 0]
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_migs](#)
- [implement_mig_cores](#)
- [refresh_hw_mig](#)
- [report_hw_mig](#)
- [set_property](#)

commit_hw_sio

現在のハードウェア オブジェクトのプロパティの変更を確定します。任意のシリアル I/O (スキャンまたはスイープ以外)、デバイス、ターゲット、またはサーバー ハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。

構文

```
commit_hw_sio [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

Vivado Design Suite のハードウェア マネージャー機能で指定のシリアル I/O ハードウェア オブジェクトで定義されているプロパティの現在の値を、現在のハードウェア デバイスに確定します。

オブジェクトには、GT、RX、TX、PLL、または Common など、ハードウェア SIO スキャン (hw_sio_scan) およびハードウェア SIO スイープ (hw_sio_sweep) オブジェクトを除く任意のシリアル I/O オブジェクトを指定できます。SIO オブジェクトには、デバイス、ターゲット、またはサーバー ハードウェア オブジェクトも含まれます。

SIO IBERT コアでは、オブジェクト プロパティを使用して値を設定および確定します。ハードウェア オブジェクトのプロパティ値は、`set_property` コマンドを使用して設定します。その後、`commit_hw_sio` コマンドを使用して、現在のハードウェア デバイスにこれらの値を駆動します。



ヒント: ハードウェア オブジェクトのプロパティをデバイスの実際の値でアップデートするには、`refresh_hw_sio` コマンドを使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_objects> (必須): 更新するハードウェア SIO (hw_sio) オブジェクトを 1 つ以上指定します。hw_sio は、`get_hw_sio_*` コマンドのいずれかを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定のシリアル I/O GT オブジェクトの `CPLL_REFCLK_DIV` プロパティの値を設定し、そのオブジェクトのすべてのプロパティの値を現在のハードウェア デバイスに確定しています。

```
set_property CPLL_REFCLK_DIV 3 [get_hw_sio_gts *MGT_X0Y8]
commit_hw_sio [list [get_hw_sio_gts *MGT_X0Y8]] ]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)
- [set_property](#)

commit_hw_sysmon

現在のハードウェア オブジェクトのプロパティの変更を確定します。ハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、ハードウェア デバイス (hw_device)、またはハードウェア システム モニター (hw_sysmon) オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。

構文

```
commit_hw_sysmon [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_objects>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

commit_hw_sysmon コマンドは、ハードウェア システム モニター (hw_sysmon) オブジェクトの現在のプロパティ値を、ハードウェア デバイス上のシステム モニター レジスタに確定します。

hw_sysmon オブジェクトのプロパティ (CONFIG.* など) の値を変更した場合、commit_hw_sysmon コマンドを使用して確定するまで、プロパティ値はハードウェア デバイスには書き込まれません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_objects> (オプション): 指定した hw_sysmon オブジェクトのプロパティを確定します。システム モニター オブジェクトは hw_sysmon オブジェクトとして指定するか、関連付けられているハードウェア デバイス (hw_device)、ハードウェア ターゲット (hw_target)、またはハードウェア サーバー (hw_server) オブジェクトによりシステム モニターとして指定できます。

注記: オブジェクトは、名前で指定するのではなく、`get_hw_sysmon` などの `get_hw_XXX` コマンドを使用して指定する必要があります。

例

次の例では、`hw_sysmon` オブジェクトの単極/双極コンフィギュレーション レジスタ プロパティの値を設定し、その値を現在のハードウェア デバイスのシステム モニターに確定しています。

```
set_property CONFIG_REG.BU 1 [get_hw_sysmon]
commit_hw_sysmon [lindex [get_hw_sysmons] 0]
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_sysmons](#)
- [get_hw_sysmon_reg](#)
- [refresh_hw_sysmon](#)
- [set_hw_sysmon_reg](#)
- [set_property](#)

commit_hw_vio

ハードウェア VIO プロープの OUTPUT_VALUE プロパティ 値を VIO コアに書き込みます。

構文

```
commit_hw_vio [-quiet] [-verbose] <hw_objects>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア VIO およびハードウェア プロープ オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

VIO デバッグ コアのプロープに定義されている現在の値を、現在のハードウェア デバイスに確定します。

VIO (Virtual Input/Output) デバッグ コアは、ザイリンクス FPGA にプログラムされている内部信号をリアルタイムで監視および駆動できます。VIO コアは、ハードウェア プロープ (hw_probe) オブジェクトを使用してデバイス上の信号を監視および駆動します。入力プロープは、VIO コアへの入力として信号を監視します。出力プロープは、VIO コアから信号を指定の値に駆動します。

VIO コアでは、オブジェクト プロパティを使用して値を設定および確定します。set_property コマンドを使用して VIO コアの出力プロープの OUTPUT_VALUE プロパティを設定し、その後、commit_hw_vio コマンドを使用して、ハードウェア デバイス上のプロープされる信号にこれらの値を駆動します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_objects> (オプション): ハードウェア デバイスに OUTPUT_VALUE プロパティの値を確定するハードウェア VIO (hw_vio) デバッグ コアまたは hw_probe オブジェクトを指定します。1 つまたは複数のハードウェア プローブの値を確定するか、hw_vio オブジェクトを指定して VIO コアのすべてのプローブの値を確定できます。

注記: オブジェクトは、名前で指定するのではなく、get_hw_vios または get_hw_probes コマンドを使用して指定する必要があります。

例

次の例では、リセット hw_probe の OUTPUT_VALUE プロパティを High に設定し、ハードウェア デバイス (hw_device) に確定してリセット プロセスをトリガーし、その後値を Low に設定してリセットを開放しています。

```
set_property OUTPUT_VALUE 1 [get_hw_probes fast_cnt_reset \
    -of_objects [get_hw_vios hw_vio_1]]
commit_hw_vio [get_hw_probes {fast_cnt_reset} \
    -of_objects [get_hw_vios hw_vio_1]]
set_property OUTPUT_VALUE 0 [get_hw_probes fast_cnt_reset \
    -of_objects [get_hw_vios hw_vio_1]]
commit_hw_vio [get_hw_vios hw_vio_1]
```

注記: 最初の commit_hw_vio コマンドでは 1 つのハードウェア プローブをオブジェクトとして指定しており、2 番目の commit_hw_vio コマンドでは hw_vio デバッグコア全体を指定しています。

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [program_hw_devices](#)
- [refresh_hw_vio](#)
- [reset_hw_vio_activity](#)
- [reset_hw_vio_outputs](#)
- [set_property](#)

compile_c

C コードを RTL にコンパイルします。

構文

```
compile_c [-force] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-force]</code>	コンパイルを強制的に実行します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	C から RTL に変換するオブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[IPIntegrator \(IP インテグレーター\)](#)

説明

`compile_c` コマンドは、Vivado HLS からインポートされた IP コアの C、C++、および SystemC ファイルを検出し、これらのファイルを Vivado Design Suite での合成に使用する RTL ファイルに変換します。

Vivado HLS を使用すると、IP コアを RTL ではなく C や C++ などの高級言語で記述できます。

HLS ベースの IP コアを生成すると、C ソースのみが作成されます。HLS ベースの IP をアウト オブ コンテキスト (OOC) フローでまたは最上位デザインと共に合成する際、合成前に `compile_c` コマンドにより Vivado HLS が起動され、C ソース ファイルが RTL に変換されて、その RTL ソースがインポートされます。



推奨: Vivado Design Suite で Vivado HLS からの C コード ベースの IP が検出されると、`compile_c` コマンドが自動的に呼び出されます。手動で実行する必要はありません。

引数

`-force` (オプション): HLS ベースの IP の RTL を強制的に再生成します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): C、C++、または SystemC コードから RTL に変換する、現在のプロジェクトに含まれる IP オブジェクトを指定します。

例

次の例では、指定した IP オブジェクトの C 言語ファイルを取得し、RTL に変換しています。

```
compile_c [get_ips instance_name]
```

関連項目

- [get_ips](#)

compile_simlib

シミュレーション ライブラリをコンパイルします。

構文

```
compile_simlib [-directory <arg>] [-family <arg>] [-force]
               [-language <arg>] [-library <arg>] [-print_library_info <arg>]
               -simulator <arg> [-simulator_exec_path <arg>]
               [-source_library_path <arg>] [-no_ip_compile] [-32bit] [-quiet]
               [-verbose]
```

使用法

名前	説明
[-directory]	コンパイルした結果を保存するディレクトリ パスを指定します。デフォルトは . です。
[-family]	デバイス アーキテクチャを選択します。デフォルトは all です。
[-force]	コンパイル済みライブラリを上書きします。
[-language]	ライブラリをコンパイルする言語を指定します。デフォルトは all です。
[-library]	コンパイルするライブラリを選択します。デフォルトは all です。
[-print_library_info]	コンパイル済みライブラリの情報を表示します。
-simulator	指定したシミュレータ用にライブラリをコンパイルします。
[-simulator_exec_path]	指定したディレクトリからシミュレータの実行ファイルを使用します。
[-source_library_path]	Vivado 用の環境変数 XILINX_VIVADO で指定されているデフォルト パスを検索する前に、指定したディレクトリでライブラリソース ファイルを検索します。
[-no_ip_compile]	IP スタティック ファイルをリポジトリからコンパイルしません。
[-32bit]	32 ビットのコンパイルを実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のプロジェクトで使用されているセルおよび IP のザイリンクス® シミュレーション ライブラリをコンパイルするか、または複数のデザイン プロジェクトで使用するよう指定のディレクトリからコンパイルします。

Vivado Design Suite では、シミュレーション モデルが Vivado シミュレータで使用される動作およびタイミング モデルを含むファイルおよびライブラリとして提供されます。compile_simlib コマンドは、これらのライブラリをサードパーティ シミュレータ用にコンパイルします。ライブラリは、シミュレーション モデルをアップデートし、新しいバージョンのシミュレータをサポートするため、ツール リリースごとにコンパイルし直す必要があります。



重要: 新しいサードパーティ シミュレータ、あるいは Vivado Design Suite の新しいバージョンまたはアップデートをインストールするたびに、compile_simlib コマンドを再実行する必要があります。

現在のプロジェクトからこのコマンドを実行すると、下に説明するコマンドのデフォルト設定ではなく、そのプロジェクトで指定されたデバイス ファミリ、ターゲット言語、およびライブラリ設定がデフォルト値として使用されます。デフォルト設定は、コマンドの実行時にオプションを設定することで変更できます。

compile_simlib コマンドでは、シミュレーション ライブラリのコンパイルにシミュレーション コンパイル指示子を使用されます。デフォルトの設定を変更するには config_compile_simlib コマンドを使用します。

このコマンドを実行すると、コンパイルされたライブラリに関する情報が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-directory <arg> (オプション): コンパイルしたライブラリ結果を保存するディレクトリ パスを指定します。

注記: デフォルトでは、非プロジェクトモードではライブラリは現在の作業ディレクトリに保存され、プロジェクトモードでは <project>/<project>.cache/compile_simlib ディレクトリに保存されます。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

-family <arg> (オプション): 選択されたライブラリを指定のデバイス ファミリ用にコンパイルします。デフォルトでは、すべてのデバイス ファミリが生成されます。次のデバイス ファミリを指定できます。

- all (デフォルト、すべてのデバイス ファミリのライブラリを生成)
- versal (Versal™ ACAP デバイス)
- virtexuplus58g (Virtex® UltraScale+™ 58G デバイス)
- virtexuplus (Virtex® UltraScale+™ デバイス)
- virtexu (Virtex UltraScale™ デバイス)
- virtex7 (Virtex-7)
- virtex7l (Virtex-7 低消費電力)
- qvirtex7 (Virtex-7 防衛グレード)
- qvirtex7l (Virtex-7 低消費電力防衛グレード)
- spartan7 (Spartan-7)
- kintexuplus (Kintex® UltraScale+ デバイス)
- kintexu (Kintex UltraScale デバイス)
- kintex7 (Kintex-7)
- kintex7l (Kintex-7 低消費電力)
- qkintex7 (Kintex-7 防衛グレード)
- qkintex7l (Kintex-7 低消費電力防衛グレード)

- artix7 (Artix®-7)
- artix7l (Artix-7 低消費電力)
- qartix7 (Artix-7 防衛グレード)
- qartix7l (Artix-7 低消費電力防衛グレード)
- zynqplus (Zynq® UltraScale+ デバイス)
- zynq (Zynq デバイス)
- azynq (Zynq オートモーティブ)
- qzynq (Zynq 防衛グレード)

`-force` (オプション): コンパイル済みの現在のライブラリを上書きします。

`-language [verilog | vhdl | all]` (オプション): `-no_ip_compile` オプションを使用する場合にのみ必要なオプションで、ベース シミュレーション ライブラリを指定した言語でコンパイルします。このオプションを指定しない場合、`-simulator` オプションで指定したシミュレータに基づく言語に設定されます。混合言語シミュレータの場合は、Verilog および VHDL ライブラリがコンパイルされます。



ヒント: デフォルトでは、`compile_simlib` コマンドで IP のシミュレーション ライブラリがコンパイルされ、IP のすべての言語がコンパイルされます。

`-library <arg>` (オプション): コンパイルするシミュレーション ライブラリを指定します。デフォルトでは、`compile_simlib` コマンドですべてのシミュレーション ライブラリがコンパイルされます。有効な値は次のとおりです。

- all (デフォルト)
- unisim
- simprim

複数のライブラリを指定するには、各ライブラリに `-lib` オプションを個別に使用します。次に例を示します。

```
.. -library unisim -library simprim ..
```

`-print_library_info` (オプション): コンパイルされたシミュレーション ライブラリへのパスを指定します。

`-simulator <arg>` (必須): 指定したシミュレータ用にライブラリをコンパイルします。有効な値は、次のとおりです。

- modelsim: バージョン 2019.2 以降
- questasim: バージョン 2019.2 以降
- ies (Linux のみ): バージョン 15.20.073 以降
- xcelium (Linux のみ): バージョン 19.03.005 以降
- vcs_mx (Linux のみ): バージョン O-2018.09-SP2-1 以降
- riviera: バージョン 2019.04 以降
- active_hdl (Windows のみ): バージョン 10.5a

`-simulator_exec_path <arg>` (オプション): サードパーティ コンパイラおよびシミュレータの実行ファイルのディレクトリを指定します。このオプションは、ターゲットシミュレータが `$PATH` または `%PATH%` 環境変数で指定されていない場合や、`$PATH` または `%PATH%` 環境変数で指定されているパスとは別のパスを指定する場合に使用します。

`-source_library_path <arg>` (オプション): 環境変数 (`$XILINX` または `$XILINX_VIVADO`) で指定されているデフォルトパスを検索する前に、指定したディレクトリでライブラリソースファイルを検索します。

注記: このオプションは、ザイリンクステクニカルサポートから指示された場合以外は使用しないでください。

`-no_ip_compile` (オプション): デザインまたは指定のリポジトリに含まれる IP のシミュレーションファイルをコンパイルしないように指定します。デフォルトでは、`compile_simlib` コマンドで IP カタログ (ユーザーおよびサードパーティのリポジトリを含む) の IP すべてのスタティックシミュレーションファイルがコンパイルされます。このオプションは、この機能をオフにする場合に使用します。

`-32bit` (オプション): デフォルトの 64 ビットのコンパイルではなく、32 ビットモードのシミュレータコンパイルを実行します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、Virtex-7 デバイスを使用するデザインを ModelSim (VHDL) でシミュレーションするために UNISIM および SIMPRIM ライブラリをコンパイルしています。

```
compile_simlib -simulator modelsim -family virtex7 -library unisim \
    -library simprim -language vhdl
```

関連項目

- [config_compile_simlib](#)
- [export_simulation](#)
- [launch_simulation](#)

config_compile_simlib

compile_simlib のオプションを指定します。

構文

```
config_compile_simlib [-cfgopt <arg>] [-simulator <arg>] [-reset] [-quiet]
                      [-verbose]
```

使用法

名前	説明
[-cfgopt]	シミュレータの設定オプションを <simulator>.<language>.<library>.<options> の形式で設定し ます。
[-simulator]	指定したシミュレータの設定を表示します。
[-reset]	すべての設定をリセットします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示 します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

compile_simlib コマンドで使用するサードパーティ シミュレータ オプションを設定します。

Vivado Design Suite には compile_simlib コマンド用に定義済みの設定ファイルがあり、サポートされるサードパーティ シミュレータ、言語、ライブラリ用に設定オプションがあらかじめ定義されています。

config_compile_simlib コマンドを使用すると、シミュレータ、言語、およびライブラリの特定の組み合わせに対して設定オプションを変更できます。

現在の設定オプションすべてを返すには、config_compile_simlib コマンドを引数なしで使用します。

引数

-cfgopt <arg> (オプション): 指定のサードパーティ シミュレータ、言語、およびライブラリの組み合わせに対して設定オプションを指定します。-cfgopt の引数は、次の形式で指定します。

```
{<simulator>.<language>.<library>.<options> }
```

説明:

- <simulator>: オプションを設定するサードパーティ シミュレータを指定します。サポートされるサードパーティ シミュレータのバージョンは、compile_simlib コマンドを参照してください。有効な値は次のとおりです。
 - modelsim
 - questasim

- `ies`
- `vcs_mx`
- `riviera`
- `active_hdl`
- **<language>:** シミュレーション オプションを設定する言語を指定します。有効な値は `verilog` または `vhdl` です。
- **<library>:** コンパイルするライブラリを指定します。有効な値は次のとおりです。
 - `axi_bfm`
 - `ieee`
 - `simprim`
 - `std`
 - `unisim`
 - `vl`
- **<options>:** シミュレータ、言語、およびライブラリの組み合わせに対する設定オプションを指定します。
<simulator>.<language>.<library> の異なる組み合わせのデフォルト コンパイル オプションは、次のとおりです。
 - Active HDL: `-v2k5 (verilog)`、`+define+XIL_TIMING`、`-93 (vhdl)`、`-nowarn ELAB1_0026 (vhdl)`
 - Incisive Enterprise Simulator: `-MESSAGES`、`-NOLOG`、`-DEFINE XIL_TIMING`、`-v93 (vhdl)`、`-RELAX (vhdl)`
 - ModelSim: `-novopt`、`-quiet`、`+define+XIL_TIMING`、`-93 (vhdl)`、`-source (simprim, unisim)`
 - QuestaSim: `-novopt`、`-quiet`、`+define+XIL_TIMING`、`-93 (vhdl)`、`-source (simprim, unisim)`
 - Riviera: `-v2k5 (verilog)`、`+define+XIL_TIMING`、`-93 (vhdl)`、`-nowarn ELAB1_0026 (vhdl)`
 - VCS MX: `-sverilog (verilog)`、`-nc`、`+v2k (simprim, unisim)`、`+define+XIL_TIMING`

注記: サポートされているその他のコンパイル オプションは、サードパーティ シミュレータの資料を参照してください。

-simulator <arg> (オプション): 指定のシミュレータに関連する設定オプションのみを返します。有効なシミュレータ値は、`-cfgopt` を参照してください。

-reset (オプション): すべての設定を Vivado Design Suite のデフォルト設定オプションにリセットします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ModelSim シミュレータ、Verilog 言語、および Unisim ライブラリのコンパイル オプションを設定しています。

```
config_compile_simlib -cfgopt {modelsim.verilog.unisim: -quiet}
```

次の例では、複数のシミュレーション ライブラリのコンパイル オプションを設定しています。

```
config_compile_simlib -cfgopt {modelsim.verilog.synopsys: -quiet} \  
-cfgopt {modelsim.verilog.simpri:-source +define+XIL_TIMING}
```

関連項目

- [compile_simlib](#)

config_design_analysis

デザイン解析の一般的な機能を設定します。

構文

```
config_design_analysis [-max_common_paths <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-max_common_paths]</code>	フェーズ間で共通パスを検索するために考慮するパスの数 (< 20,000) を指定します。デフォルトは 1000 です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)

説明

`report_design_analysis` コマンドの機能を設定します。

デザイン解析レポートは、Vivado ツール フローのさまざまな段階 (合成、最適化、配置、配線) でタイミング パスを解析します。`-max_common_paths` オプションは、フローの各段階でキャプチャするセットアップ タイミング パスの数を指定します。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-max_common_paths <arg>` (オプション): インプリメンテーションのフェーズ間での共通パスの分布を調べる際に考慮するパスの数を指定します。20,000 未満の値を指定します。デフォルトは 1000 です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、タイミング解析中にパッケージ遅延を無視するよう指定しています。

```
config_design_analysis 500
```

関連項目

- [report_design_analysis](#)

config_hw_sio_gts

指定したデバイスのデバイス GT を設定します。

構文

```
config_hw_sio_gts [-dict <args>] [-quiet] [-verbose] <hw_device>
```

使用法

名前	説明
[-dict]	GT の設定に使用する名前/値のペアをリストします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_device>	ハードウェア デバイス オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

config_implementation

インプリメンテーション パラメーターを設定します。

構文

```
config_implementation [-quiet] [-verbose] [<list>]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<list>]</code>	設定する必要のあるパラメーターのリストを指定します。

カテゴリ

Configuration (コンフィギュレーション)

説明

インプリメンテーション プロセスの動作を制御するインプリメンテーション パラメーターを設定します。

設定可能なパラメーターまたはこのコマンドで設定されたパラメーターのリストは、`report_config_implementation` コマンドを使用してレポートできます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<list>` (必須): 設定するパラメーターとその値のリストを指定します。下の例に示すように、パラメーターとその値のペアを波かっこで囲み、それらをさらに波かっこで囲む必要があります。

例

次の例は、インプリメンテーション パラメーターを設定します。

```
config_implementation { {incr.ignore_user_clock_uncertainty True}
  <other_param> <other_value> }
```

関連項目

- [report_config_implementation](#)

config_ip_cache

IP インスタンスの合成キャッシュを管理します。オプションを指定しない場合、IP キャッシュ エントリがリストされます。

構文

```
config_ip_cache [-use_cache_location <arg>] [-use_project_cache]
                [-disable_cache] [-clear_output_repo] [-clear_local_cache]
                [-cache_has_match] [-cache_was_used] [-get_id] [-remove] [-vlnv <arg>]
                [-old_swvers] [-unused] [-swver <arg>] [-num_days_old <arg>]
                [-num_days_unused <arg>] [-obs_synth_crc] [-disk_usage_output_repo]
                [-report] [-rptfile <arg>] [-csvfile <arg>] [-import_from_project]
                [-filter <arg>] [-regex] [-nocase] [-purge] [-quiet] [-verbose] [<ip>]
```

使用法

名前	説明
[-use_cache_location]	現在のプロジェクトのプロパティを指定のキャッシュ ロケーションを使用するように設定します。
[-use_project_cache]	現在のプロジェクトのプロパティをデフォルトのプロジェクト IP キャッシュ ロケーションを使用するように設定します。
[-disable_cache]	キャッシュの使用をディisableにします。
[-clear_output_repo]	現在のプロジェクト用のキャッシュ (ローカルまたはリモート) に存在するすべてのキャッシュ エントリをディスクおよびメモリから削除します。
[-clear_local_cache]	現在のプロジェクトのローカル キャッシュ エントリすべてをディスクおよびメモリから削除します。
[-cache_has_match]	この IP インスタンスで機能するキャッシュ エントリのキャッシュ ID を返します。ない場合は " を返します。
[-cache_was_used]	IP の現在の合成結果を取得するのにキャッシュが使用された場合は 1、使用されなかった場合は 0 を返します。
[-get_id]	指定した <ip> の IP キャッシュ ID 文字列を計算して返します。
[-remove]	指定の IP インスタンスまたは指定の cachedInst に対応するキャッシュ エントリを削除します。正しく削除された場合はキャッシュ ID が返され、正しく削除されなかった場合は何も返されません。
[-vlnv]	-purge または -get_resource_data オプションと共に使用し、削除または検索するキャッシュ エントリの VLNv を指定します。VLNV の 1 つまたは複数のフィールドにワイルドカード (*) を使用できます。
[-old_swvers]	-purge オプションと共に使用し、以前の Vivado ツール バージョンで作成されたキャッシュ エントリを削除します。
[-unused]	-purge オプションと共に使用し、一度も使用されていないキャッシュ エントリを削除します。
[-swver]	-purge オプションと共に使用し、指定した Vivado ツール バージョン (2017.1 など) で作成されたキャッシュ エントリを削除します。
[-num_days_old]	-purge オプションと共に使用し、指定した日数以上経過しているキャッシュ エントリを削除します。

名前	説明
<code>[-num_days_unused]</code>	-purge オプションと共に使用し、指定した日数以上使用されていないキャッシュ エントリを削除します。
<code>[-obs_synth_crc]</code>	-purge オプションと共に使用し、コンポーネント合成チェックサムが IP カタログの現在のコンポーネント合成チェックサムと一致しないキャッシュ エントリを削除します。
<code>[-disk_usage_output_repo]</code>	現在のプロジェクトの <code>ip_output_repo</code> にあるすべてのキャッシュ エントリの総ディスク使用量を MB で返します。
<code>[-report]</code>	指定した IP またはキャッシュ オブジェクト、あるいは IP を指定しない場合は現在のキャッシュ位置のキャッシュ統計をレポートします。ートされます。-rptfile オプションを指定すると、統計をファイルに記述できます。-dir オプションを指定すると、そのディレクトリにあるキャッシュ エントリの統計が記述されます。
<code>[-rptfile]</code>	-report オプションと共に使用し、キャッシュ統計を記述するテキスト ファイルを指定します。
<code>[-csvfile]</code>	-report オプションと共に使用し、キャッシュ統計を記述する CSV ファイルを指定します。
<code>[-import_from_project]</code>	既存の合成済み IP をプロジェクトからキャッシュにインポートします。
<code>[-filter]</code>	-list オプションで返されたリストにフィルターを適用します。
<code>[-regexp]</code>	-filter オプションの引数に glob ではなく正規表現を使用します。
<code>[-nocase]</code>	-filter オプションの引数の大文字/小文字を区別しません。
<code>[-purge]</code>	-vlnv、-obs_swvers、-obs_synth_crc、または -swver オプションで指定されたタイプのキャッシュ エントリをすべて削除します。削除されたエントリの数が表示されます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<ip>]</code>	IP インスタンス オブジェクト、IP ファイル、または IP 名のパターンを指定します。

カテゴリ

Object (オブジェクト)、IPFlow (IP フロー)

説明

Vivado Design Suite の独立階層 (OOC) IP のキャッシュを管理します。Vivado Design Suite では、IP リポジトリのカスタマイズされた OOC IP の合成結果がキャッシュされ、同じカスタマイズ プロファイルを使用する複数の IP で OOC 合成結果を共有して実行時間を短縮できます。キャッシュされた合成結果は、1 つのプロジェクト内でプロジェクトのキャッシュから再利用するか、リモート キャッシュ ロケーションを使用して複数のプロジェクト間で再利用できます。

OOC IP の合成出力ファイルを生成する際、一致する IP のカスタマイズが IP リポジトリに見つかった場合、キャッシュされた合成結果が使用されます。適切な IP のカスタマイズが見つからない場合は、通常通りに合成出力ファイルが生成され、その結果とデザイン チェックポイント (DCP) が今後再利用できるように IP 合成キャッシュにコピーされます。



ヒント: キャッシュされた結果が使用された場合、Tcl コンソールに IP キャッシュ ロケーションが使用されたことを示す情報メッセージが表示されます。

キャッシュされた IP 合成のリポジトリには、現在のプロジェクトまたは外部ロケーションを指定できます。IP キャッシュの場所はプロジェクトの IP_OUTPUT_REPO プロパティで定義され、有効なファイル システム ディレクトリを文字列で指定します。このプロパティは、`check_ip_cache` コマンドで `-use_cache_location` または `-use_project_cache` オプションを使用して設定できます。デフォルトの IP 合成キャッシュ ロケーションは現在のプロジェクト フォルダです。

IP 合成キャッシュの使用は、`set_property` コマンドで設定可能な IP_CACHE_PERMISSIONS プロパティで制御します。有効な値は、次のとおりです。

- `disabled`: IP 合成キャッシュを使用しません。これがデフォルト設定です。
- `read`: IP 合成キャッシュから OOC 合成結果を読み出し、現在のプロジェクトに適用します。
- `write`: IP 合成キャッシュを OOC 合成結果を書き込むのには使用しますが、現在のプロジェクトには IP を読み込みません。
- `read write`: IP 合成キャッシュに結果を書き込み、それらの結果を現在のプロジェクトに使用します。

`config_ip_cache` コマンドを実行した後に `update_ip_catalog` コマンドを実行し、指定の IP キャッシュ リポジトリを Vivado ツールに読み込む必要があります。



ヒント: Vivado Design Suite IDE では、[Settings] ダイアログ ボックスで IP 合成キャッシュをイネーブルしたり、キャッシュ リポジトリを指定できます。キャッシュの使用に関する詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896) を参照してください。

`config_ip_cache` コマンドを次に示すオプションを指定せずにデフォルトで実行すると、IP 合成キャッシュ内のエントリのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-use_cache_location <arg>` (オプション): 現在のプロジェクトの IP キャッシュ リポジトリとして外部ディレクトリを指定します。

`-use_project_cache` (オプション): IP 合成結果のキャッシュにおけるデフォルト設定で、現在のプロジェクトでローカル プロジェクト ディレクトリ内のデフォルトの IP キャッシュ ロケーションを使用するよう指定します。このオプションを使用すると、デフォルト設定を復元できます。

`-disable_cache` (オプション): IP 合成キャッシュをディスエーブルにします。このオプションを使用すると、IP_CACHE_PERMISSIONS プロパティが `disabled` に設定されます。



ヒント: キャッシュを再度イネーブルにするには、IP_CACHE_PERMISSIONS プロパティを設定するか、Vivado Design Suite IDE を使用します。

`-clear_output_repo` (オプション): IP 合成キャッシュがプロジェクトのローカルにあるか複数のプロジェクトで共有されるリモート ディレクトリにあるかにかかわらず、キャッシュされている結果をすべて削除します。



注意: 削除を確認するメッセージは表示されません。

`-clear_local_cache` (オプション): IP 合成キャッシュがプロジェクトのローカルにある場合にのみ、キャッシュされている結果をすべて削除します。

`-cache_has_match` (オプション): 指定の IP に対して IP キャッシュをクエリし、一致がある場合に `Vendor:Library:Name:Version (VLNV)` を返します。

`-cache_was_used` (オプション): 指定の IP に対して IP キャッシュをクエリし、合成中にキャッシュ エンティティが使用されていた場合は `TRUE` を返します。

`-get_id` (オプション): 現在のプロジェクトの指定した `<ip>` オブジェクトに使用された、リポジトリにあるキャッシュされた IP の ID 文字列を返します。キャッシュ エンティティの ID は、VLNV 値の N の部分です。

`-remove` (オプション): `get_ips` コマンドで指定された IP コアの現在の IP キャッシュからエントリを削除します。

`-purge` (オプション): 現在の IP キャッシュから指定のエントリを削除します。エントリは、次のオプションを使用して指定します。

- `-vlnv <arg>` (オプション): Vendor:Library:Name:Version (VLNV) で指定した IP を削除します。VLNV のどのフィールドにもワイルドカード (*) を使用できます。



ヒント: VLNV は、IP の IPDEF プロパティです。

- `-swver <arg>` (オプション): 指定したツール バージョンで作成されたキャッシュ エントリを削除します。ツール バージョンは `<version>.<revision>` の形式で指定します (例: 2017.2)。
- `-old_swvers` (オプション): 現在使用しているツール バージョンより古いツール バージョンで作成されたキャッシュ エントリを削除します。
- `-unused` (オプション): 使用されたことのないキャッシュ エントリを削除します。
- `-num_days_old <arg>` (オプション): 指定した日数より古いキャッシュ エントリを削除します。
- `-num_days_unused <arg>` (オプション): 指定した日数使用されていないキャッシュ エントリを削除します。
- `-obs_synth_crc` (オプション): 現在の IP カタログのコンポーネントにあるチェックサムと合成チェックサムが異なるコンポーネントで生成されたキャッシュ エントリを削除します。

`-disk_usage_output_repo` (オプション): IP 合成キャッシュのディスク使用量を KB でレポートします。

`-report` (オプション): 表す IP および各キャッシュ エントリのヒット数を含む IP キャッシュのレポートを生成します。

`-rptfile <arg>` (オプション): `-report` オプションと共に使用し、レポートを指定のファイルに出力します。

`-import_from_project` (オプション): 現在の IP キャッシュに現在のデザインからの合成結果を保存します。既存のプロジェクトの合成結果を IP キャッシュに保存し、ほかのプロジェクトまたはデザイン イテレーションで使用できます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`config_ip_cache` で返された結果に IP キャッシュ エントリ (`ip_cache_entry`) オブジェクトのプロパティ値に基づいてフィルターを適用できます。キャッシュされた IP に設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。`ip_cache_entry` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`INSTANCE_NAME`、`CORE_VLNV`、`CUSTOMIZATION` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価(==)、不等価(!=)、一致(=~)、不一致(!~)です。数値比較演算子<、>、<=、および>=も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および **-filter** オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド **regexp** はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、**-regexp** オプションを使用した場合にのみ適用されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、**set_msg_config** コマンドで定義できます。

<ip> (オプション): 1 つの IP オブジェクトを **get_ips** コマンドで返して指定します。

例

次の例では、現在の IP キャッシュの内容を 1 行に 1 キャッシュ エントリずつ表示します。

```
join [config_ip_cache] \n
```

次の例では、外部 IP キャッシュ ロケーションを使用することを指定し、IP カタログのリポジトリ設定をアップデートしています。

```
config_ip_cache -use_cache_location C:/Data/ip
update_ip_catalog
```

次の例では、**config_ip_cache** コマンドで返された **ip_cache** オブジェクトにフィルターを適用して指定のインスタンス名のオブジェクトを返し、そのオブジェクトのプロパティをレポートしています。

```
report_property -all [config_ip_cache -filter {INSTANCE_NAME ==
base_mb_mdm_1_0}]
```

次の例では、IP キャッシュ機能をディスエーブルにし、キャッシュされた合成結果が使用されないようにしています。

```
config_ip_cache -disable_cache
```

次の例では、指定した IP コアの IP キャッシュ識別子を取得しています。

```
config_ip_cache -get_id [get_ips base_mb_mdm_1_0]  
d3e03f3ed484c174
```

次の例では、指定した IP の IP キャッシュ エントリを削除しています。

```
config_ip_cache -remove [get_ips base_mb_mdm_1_0]  
config_ip_cache -purge -vlnv [get_property IPDEF [lindex [get_ips] 0 ]]
```

関連項目

- [get_ips](#)
- [import_ip](#)
- [set_property](#)
- [synth_design](#)
- [synth_ip](#)
- [update_ip_catalog](#)

config_timing_analysis

タイミング解析の一般設定を指定します。

構文

```
config_timing_analysis [-enable_input_delay_default_clock <arg>]
                        [-enable_preset_clear_arcs <arg>] [-ignore_io_paths <arg>]
                        [-disable_flight_delays <arg>] [-merge_exceptions <arg>]
                        [-timing_early_launch_at_borrowing_latches <arg>]
                        [-enable_time_borrowing_for_max_delay_exceptions <arg>] [-quiet]
                        [-verbose]
```

使用法

名前	説明
<code>[-enable_input_delay_default_clock]</code>	内部定義されたクロックからの SDC のクロックなし入力遅延をイネーブルにします。有効な値は true、false です。このオプションは UCF 制約ではサポートされていません。
<code>[-enable_preset_clear_arcs]</code>	非同期プリセットまたはクリア タイミング アークを介するタイム パスをイネーブルにします。有効な値は true、false です。
<code>[-ignore_io_paths]</code>	プライマリ入力からのパスおよびプライマリ出力へのパスを無視します。有効な値は true、false です。
<code>[-disable_flight_delays]</code>	I/O の計算にパッケージ遅延を追加しません。有効な値は true、false です。
<code>[-merge_exceptions]</code>	タイミング エンジンによるタイミング例外の統合をイネーブル/ディスエーブルにします。有効な値は true、false です。
<code>[-timing_early_launch_at_borrowing_latches]</code>	透過ラッチを介するパスのソース イネーブルからクロック レイテンシの不必要に悪い見積もり部分を削除します。有効な値は auto、true、false で、デフォルトは auto です。
<code>[-enable_time_borrowing_for_max_delay_exceptions]</code>	set_max_delay タイミング例外が適用されるタイミング パスで時間を借りられるようにします。有効な値は true および false です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)

説明

タイミング解析の一般的な機能を設定します。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-enable_input_delay_default_clock [true | false]` (オプション): タイミング解析で内部定義されたクロックからのクロックなし入力遅延をイネーブルにします。有効な値は `true` または `false` で、デフォルト値は `false` です。

`-enable_preset_clear_arcs [true | false]` (オプション): 非同期プリセットまたはクリア タイミング アークを介するタイミング パスをイネーブルにします。有効な値は `true` または `false` で、デフォルト値は `false` です。デフォルトでは、Vivado タイミング エンジンで非同期リセットのタイミング アークはディスエーブルになります。このオプションを使用すると、タイミング解析でこれらをイネーブルにすることにより、問題がデザインに存在する非同期リセットのアサートが原因で発生しているかどうかを確認できます。

`-ignore_io_paths [true | false]` (オプション): プライマリ入力からのパスおよびプライマリ出力へのパスを無視します。このオプションを使用すると、タイミング パスから入力ポートからのネット遅延および出力ポートへのネット遅延を削除できます。

`-disable_flight_delays [true | false]` (オプション): `true` の場合、I/O の計算にパッケージ遅延を追加しません。

`-timing_early_launch_at_borrowing_latches [auto | true | false]` (オプション): 透過ラッチを介するパスのソース イネーブルからクロック レイテンシの不必要に悪い見積もり部分を削除します。イネーブルになっているラッチでは、セットアップ/ホールドを算出するタイミング パスは、ラッチの G ピンではなく D ピンから開始します。このオプションは、それらのタイミング パスに影響します。このオプションは、CRPR がイネーブルでない場合に見積もりの良いタイミングを補正するために `true` に設定できます。この場合、Vivado のタイミング エンジンでソース クロック レイテンシを算出するためにラッチが開いたときから初期のソース エッジが使用されます。CRPR がイネーブル (デフォルト) の場合は、このオプションを使用すると控えめすぎる結果となるので、`false` に設定してください。デフォルト設定は `auto` で、Vivado タイミング エンジンによりイネーブルのラッチを介するパスから CRPR を削除するかが決定されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、タイミング解析中にパッケージ遅延を無視するよう指定しています。

```
config_timing_analysis -disable_flight_delays true
```

関連項目

- [config_timing_corners](#)
- [report_timing](#)
- [report_timing_summary](#)

config_timing_corners

シングル/マルチ コーナーのタイミング解析を設定します。

構文

```
config_timing_corners [-corner <arg>] [-delay_type <arg>] [-setup] [-hold]
                        [-quiet] [-verbose]
```

使用法

名前	説明
[-corner]	変更するタイミング コーナーの名前を指定します。有効な値は、Slow または Fast です。
[-delay_type]	指定したタイミング コーナーを解析するパス遅延のタイプを指定します。有効な値は、none、max、min、min_max です。
[-setup]	セットアップ解析のタイミング コーナーをイネーブルにします (-delay_type max と同じ)。
[-hold]	ホールド解析のタイミング コーナーをイネーブルにします (-delay_type min と同じ)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)

説明

現在のデザインのシングル/マルチ コーナー タイミング解析でのスローおよびファースト タイミング コーナーを設定します。このコマンドを実行する場合は、合成済みまたはインプリメント済みデザインを開いておく必要があります。

物理的なデバイスの製造プロセスの変動、およびデバイス動作時の電圧および温度の変動の組み合わせにより、タイミング コーナーが作成されます。これらの 3 つの変数 (PVT) により、デバイス全体の遅延が決定されます。ファースト コーナーは、最小製造プロセス許容誤差、最高電圧、最低温度でのデバイスの動作を表します。スロー コーナーは、最大製造プロセス許容誤差、最低電圧、最高温度でのデバイスの動作を表します。デフォルトでは、スローおよびファースト プロセス コーナーの両方でセットアップおよびホールド解析が実行されます (クワッド解析)。

```
config_timing_corners -corner Slow -setup -hold
config_timing_corners -corner Fast -setup -hold
```

config_timing_corners コマンドを使用すると、Vivado タイミング エンジンによるデフォルトの 4 コーナー解析を制限し、タイミング パフォーマンスを向上できます。両コーナーのデフォルトの解析を変更またはディスエーブルにするには、ファースト コーナーおよびスロー コーナーの両方を設定する必要があります。

```
config_timing_corners -corner Slow -delay_type max
config_timing_corners -corner Fast -delay_type none
```

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-corner [slow | fast]` (オプション): 設定するタイミング コーナーを指定します。有効な値は、slow または fast です。`-corner` を指定しない場合、`-delay_type` が両方のコーナーに適用されます。

`-delay_type <value>` (オプション): 指定したタイミング コーナーを解析するパス遅延のタイプを指定します。有効な値は none、max、min、および min_max です。`-delay_type` オプションを none に設定すると、`-corner` で指定したコーナーがタイミング解析から除外されます。



ヒント: `-delay_type` および `-setup/-hold` はオプションですが、これらのいずれかを使用してコーナーを指定する必要があります。

`-setup` (オプション): 指定のタイミング コーナーに対してセットアップ解析を指定します。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): 指定のタイミング コーナーに対してホールド解析を指定します。これは `-delay_type min` を指定するのと同じです。



ヒント: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` と同じになります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、セットアップ解析およびホールド解析の両方でスロー タイミング コーナーを設定します。

```
config_timing_corners -corner slow -setup -hold
config_timing_corners -corner slow -delay_type min_max
```

注記: 上記のどちらでも、同じ結果が得られます。

次の例では、最小遅延解析でファースト タイミング コーナーを設定し、スロー コーナー解析をディスエーブルにしています。

```
config_timing_corners -corner fast -delay_type min
config_timing_corners -corner slow -delay_type none
```

関連項目

- [config_timing_analysis](#)
- [report_timing](#)

config_webtalk

ソフトウェア、IP、およびデバイスの使用統計をザイリンクスに送信する WebTalk をイネーブル/ディスエーブルにします。注記: WebPACK では WebTalk は常に有効です。WebPACK ライセンスを使用してビットストリームを生成する場合、ユーザーおよびインストールのプリファレンスは無視されます。WebPACK に含まれるデバイスを使用していて WebPACK ライセンスが存在する場合、常に WebPACK ライセンスが使用されます。これを変更するには、アンサー 34746 を参照してください。

構文

```
config_webtalk [-info] [-user <arg>] [-install <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-info]	現在 WebTalk がイネーブルかディスエーブルかを表示します。
[-user]	現在のユーザーに対して WebTalk をイネーブル/ディスエーブルにします。イネーブルにする場合は on、ディスエーブルにする場合は off に設定します。デフォルトは空です。
[-install]	現在のインストールのすべてのユーザーに対して WebTalk をイネーブル/ディスエーブルにします。イネーブルにする場合は on、ディスエーブルにする場合は off に設定します。off に設定した場合、個々のユーザーが -user オプションを使用して WebTalk をイネーブルにすることはできません。このオプションを使用するには、管理者権限が必要な場合があります。デフォルトは空です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

WebTalk はデザイン データを収集するためのザイリンクス ツールの機能であり、ザイリンクス デバイス、ツール、および IP がどのように使用されているかをザイリンクスが理解するのに役立ちます。

このコマンドは、現在のユーザーおよびツール インストールにおける WebTalk 機能の現在の設定を返します。また、ツール、IP、およびデバイスの使用統計をザイリンクスに送信する WebTalk の機能をイネーブル/ディスエーブルにすることもできます。WebPACK ライセンスを使用してビットストリームを生成する場合以外は、WebTalk をディスエーブルにするとデータは送信されません。

WebTalk をイネーブルにするかどうかはユーザーの自由ですが、WebPACK を使用している場合は常にイネーブルになっており、常にデータが送信されます。WebPACK ライセンスを使用してビットストリームを生成する場合、ユーザーおよびインストールのプリファレンスは無視されます。

注記: WebPACK に含まれるデバイスを使用していて WebPACK ライセンスが存在する場合、常に WebPACK ライセンスが使用されます。これを変更するには、アンサー 34746 を参照してください。

引数

`-info` (オプション): 現在の WebTalk 設定を返します。WebTalk の設定は、ユーザーおよびインストールの両方の設定によります。いずれかの設定がディスエーブルになっていると、WebTalk はディスエーブルになります。

`-user <arg>` (オプション): 現在のユーザーに対して WebTalk をイネーブルまたはディスエーブルにします。

`-install` (オプション): 現在のツール インストールに対して WebTalk をイネーブルまたはディスエーブルにします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、WebTalk の現在の設定が返されます。

```
config_webtalk -info
INFO: [Coretcl-120] Webtalk has been disabled by the current user.
INFO: [Coretcl-123] Webtalk has been enabled for the current installation.
INFO: [Coretcl-110] This combination of user/install settings means that
WebTalk is currently disabled.
```

次の例では、現在のユーザーに対して WebTalk をイネーブルにしています。

```
config_webtalk -user on
```

connect_bd_intf_net

インターフェイス ポート (intf_port) とインターフェイス ピン (intf_pin) を接続します。

構文

```
connect_bd_intf_net [-intf_net <arg>] [-boundary_type <arg>] [-quiet]
                    [-verbose] <object1> <object2> [<auto>]
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-intf_net]	すべてのオブジェクトを接続する 1 つのインターフェイス ネットを指定します。
[-boundary_type]	ソース オブジェクトが階層ブロックのインターフェイス ピン上にある場合に使用します。有効な値は、upper、lower、both です。lower に設定すると、下位階層から検索が実行されます。デフォルトは both です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<object1>	接続するインターフェイス ポートまたはインターフェイス ピンの名前を指定します。
<object2>	接続するインターフェイス ポートまたはインターフェイス ピンの名前を指定します。
[<auto>]	関連付けられているピンを自動的に接続します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター セルのインターフェイス ピンをほかのインターフェイス ピンまたは外部インターフェイス ポートに接続します。インターフェイスとは、IP インテグレーター サブシステム デザインで共通の機能を共有する信号をグループ化したものです。

このコマンドは、-intf_net オプションで指定された名前のインターフェイス ネットを作成して、指定の名前の既存のインターフェイス ネットに接続するか、または名前が指定されていない場合は名前を割り当てます。

このコマンドを実行すると、接続されたインターフェイス ネット オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-intf_net <arg>` (オプション): `create_bd_intf_net` で作成済みの既存のインターフェイス ネット名を指定するか、作成する新しいインターフェイス ネットを指定します。名前を指定しない場合、自動的にネットに名前が付けられます。

`-boundary_type [lower | upper | both]` (オプション): 階層ブロックのインターフェイス ピンに接続するオブジェクトの検索エリアを指定します。デフォルトの `both` では、現在のブロック デザイン階層から、下位に向かって接続オブジェクトを検索します。`upper` に設定すると現在の階層レベルのみが検索され、`lower` に設定すると最下位階層から上位に向かって検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<object1>` (必須): ネットを接続する最初のピンまたはポートを指定します。

`<object2>` (必須): ネットを接続する 2 番目のピンまたはポートを指定します。

例

次の例では、IP インテグレーター コアのインターフェイス ピンをサブシステム デザインのインターフェイス ポートに接続しています。

```
connect_bd_intf_net [get_bd_intf_pins clk_wiz_1/CLK_IN1_D] \
    [get_bd_intf_ports /diff_clock_rtl]
```

関連項目

- [create_bd_cell](#)
- [create_bd_intf_net](#)

connect_bd_net

ポートとピン オブジェクトを接続します。

構文

```
connect_bd_net [-net <arg>] [-boundary_type <arg>] [-quiet] [-verbose]
               <objects>...
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-net]	すべてのオブジェクトを接続する 1 つのネットを指定します。
[-boundary_type]	ソース オブジェクトが階層ブロックのピン上にある場合に使用します。有効な値は、upper、lower、both です。lower に設定すると、下位階層から検索が実行されます。デフォルトは both です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	ネットに接続するオブジェクトを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインにブロック図のポートおよびピン オブジェクトを接続する新しいネットを作成するか、既存のネットを指定のピンおよびポートに接続します。

-net オプションを指定しない場合、指定したオブジェクトを接続する新しいネットが作成されます。-net オプションを使用すると、指定のネットが必要に応じて接続または作成されます。

接続するポートおよびピン オブジェクトを指定するには、get_bd_ports および get_bd_pins コマンドを使用します。

このコマンドを使用すると、サブシステム デザインの異なる階層レベルにあるピンまたはポートを接続できますが、その場合は -net オプションは指定できません。-net を指定すると、接続が 1 つのネットではなく複数のネットになります。

このコマンドを実行すると、接続された IP インテグレーター サブシステム デザインのネット オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-net <arg> (オプション): 現在の IP サブシステム デザインに 1 つのネットを作成します。

注記: `-net` はオプションです。接続するオブジェクトが同じ階層レベルにない場合は指定しないでください。

`-boundary_type [lower | upper | both]` (オプション): 階層ブロックのピンに接続するオブジェクトの検索エリアを指定します。デフォルトの `both` では、現在のブロック デザイン階層から、下位に向かって接続オブジェクトを検索します。`upper` に設定すると現在の階層レベルのみが検索され、`lower` に設定すると最下位階層から上位に向かって検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects>: 接続する現在の IP インテグレーター サブシステム デザインのポートおよびピンを指定します。

例

次の例では、IP サブシステム デザインの異なる階層レベルにある 2 つのピンを接続しています。

```
connect_bd_net [get_bd_pins /vidOut_1/locked] \
    [get_bd_pins /newMod1/bridge_1/fid]
```

注記: `/vidOut_1/locked` と `/newMod1/bridge_1/fid` はサブシステム デザインの異なる階層レベルにあるので、`-net` オプションは指定しません。この場合、階層をまたいで接続するために複数のネットが作成されます。

関連項目

- [create_bd_net](#)
- [disconnect_bd_net](#)
- [get_bd_pins](#)
- [get_bd_ports](#)

connect_debug_cores

デバッグ スレーブ インスタンスをマスター インスタンスに接続します。有効なマスターは、[From BSCAN To DebugHub] モードに設定されたデバッグ ブリッジまたはデバッグ ハブ インスタンスです。有効なスレーブは、ILA、VIO、JTAG_to_AXI などのデバッグ コアです。connect_debug_cores では、同じ領域 (リコンフィギュラブル パーティションまたはスタティック) に存在するマスター インスタンスとスレーブ インスタンスのみを接続できます。

構文

```
connect_debug_cores -master <args> -slaves <args> [-quiet] [-verbose]
```

戻り値

デバッグ マスター インスタンスおよびスレーブ インスタンス

使用法

名前	説明
-master	[From BSCAN To DebugHub] モードに設定された有効なデバッグ ブリッジまたはデバッグ ハブ インスタンスを指定します。指定できるマスター インスタンスは 1 つのみです。
-slaves	スレーブ インスタンスを 1 つまたは複数指定します。有効なスレーブ インスタンスは、ILA、VIO、JTAG_to_AXI などのデバッグ コアです。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Debug \(デバッグ\)](#)、[XDC](#)

説明

デバッグ スレーブ インスタンスを指定したマスター インスタンスに接続します。このコマンドでは、指定したスレーブを既存のデバッグ チェーンに追加できます。その場合、既にチェーンに存在するデバッグ スレーブに影響を与えずに、指定したスレーブをデバッグ ハブまたはブリッジに接続できます。

デバッグ マスターには、デバッグ ハブまたはデバッグ ブリッジを指定できます。Vivado デバッグ ハブ コアは、ザイリンクス デバイスの JTAG バウンダリスキャン (BSCAN) インターフェイスと、ILA (Integrated Logic Analyzer)、VIO (Virtual Input/Output)、JTAG-to-AXI などの Vivado デバッグ コアとの間のインターフェイスを提供します。Vivado Debug Bridge は、フラット デザインまたはパーシャル リコンフィギュレーション (PR) デザイン内のデバッグ コアと通信するためのオプションを複数提供するコントローラーです。Debug Bridge は、JTAG ケーブルを使用するか、あるいは JTAG ケーブルを使用せずにイーサネット、PCIe、またはその他のインターフェイスを介してリモートでデザインをデバッグするよう設定できます。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。



重要: パーシャル リコンフィギュレーション (PR) デザインの場合、connect_debug_cores コマンドではスタティック領域内または同じリコンフィギュラブル パーティション内にあるマスター インスタンスとスレーブ インスタンスのみを接続できます。

引数

`-master <arg>` (必須): PR デザインの [From BSCAN To Debug HUB] モードに設定されたデバッグ ハブ インスタンスまたはデバッグ ブリッジを指定します。指定できるマスターは 1 つのみです。

`-slaves <arg>` (必須): 1 つまたは複数のデバッグ コア スレーブ インスタンスを指定します。有効なスレーブ インスタンスは、ILA、VIO、または JTAG_to_AXI デバッグ コアです。

注記: 指定したスレーブが既に別のマスターに接続されている場合は、そのマスターへの接続が解除されてから新しいマスターに接続されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定した ILA コアをデバッグ ブリッジに接続しています。

```
connect_debug_cores -master [get_cells inst_count/debug_bridge_0] \
-slaves [list [get_cells inst_count/ila_0] [get_cells inst_count/ila_1] ]
```

関連項目

- [create_debug_core](#)
- [get_cells](#)
- [get_debug_cores](#)
- [get_hw_axis](#)
- [get_hw_ilas](#)
- [get_hw_vios](#)

connect_debug_port

ネットとピンをデバッグ ポート チャンネルに接続します。

構文

```
connect_debug_port [-channel_start_index <arg>] [-quiet] [-verbose] <port>
                   <nets>...
```

使用法

名前	説明
<code>[-channel_start_index]</code>	チャンネル インデックスからネットを接続します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><port></code>	デバッグ ポート名を指定します。
<code><nets></code>	ネットまたはピンを指定します。

カテゴリ

[Debug \(デバッグ\)](#)、[XDC](#)

説明

ネットリスト デザインからの信号を、`create_debug_core` コマンドを使用してデザインに追加された ILA デバッグ コアのポートに接続します。信号は、ポートの特定のチャンネル インデックスに接続するか、ポートで使用可能なチャンネルに接続できます。

ポートに接続する信号が多すぎたり、接続をサポートするだけのチャンネルがない場合は、エラー メッセージが表示されます。

デバッグ コアにポートを追加するには `create_debug_port` コマンド、既存ポートで使用可能なチャンネルを増加するには `set_property port_width` コマンドを使用します。例を参照してください。

ポートから信号の接続を解除するには、`disconnect_debug_port` コマンドを使用します。

デバッグ コアを定義して接続すると、デバッグ コアをブロックとしてインプリメントし、ネットリスト デザインに含めることができます。コアをインプリメントするには、`implement_debug_core` コマンドを使用します。

引数

`-channel_start_index <arg>` (オプション): 接続に使用するチャンネル インデックスを指定します。複数の信号を指定した場合、このオプションで指定したチャンネル インデックス番号から接続が追加されます。チャンネル インデックスの番号は 0 から開始します。

注記: このオプションを指定しない場合、最初に使用可能なチャンネル インデックスに接続されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<port>` (必須): 信号を接続するポートの名前を指定します。ポートは、`core_name/port_name` で指定する必要があります。

`<nets>` (必須): 指定したデバッグ ポートに接続する、ネットリスト デザインからのネット名を 1 つまたは複数指定します。

例

次の例では、`myCore` デバッグ コアに新しい PROBE ポートを作成し、ポートの `PORT_WIDTH` プロパティを増加して接続される信号数を受信できるようにし、信号を 3 番目のチャンネル位置 (インデックス 2) からポートに接続します。

```
create_debug_port myCore PROBE
set_property PORT_WIDTH 8 [get_debug_ports myCore/PROBE1]
connect_debug_port myCore/PROBE1 [get_nets [list m0_ack_o m0_cyc_i \
    m0_err_o m0_rty_o m0_stb_i m0_we_i ]] -channel_start_index 2
```

注記: ポートで使用可能なチャンネルに接続するネット数が多すぎると、エラー メッセージが表示され、ポートは接続されません。

関連項目

- [create_debug_core](#)
- [create_debug_port](#)
- [disconnect_debug_port](#)
- [get_debug_ports](#)
- [get_nets](#)
- [implement_debug_core](#)
- [set_property](#)

connect_hw_server

ハードウェア サーバーへの接続を開きます。

構文

```
connect_hw_server [-url <arg>] [-cs_url <arg>] [-quiet] [-verbose]
```

戻り値

ハードウェア サーバー

使用法

名前	説明
[-url]	ハードウェア サーバー URL を指定します。デフォルトは 3121 です。
[-cs_url]	cs_server に使用する URL を指定します。デフォルト URL が使用されてそこにサーバーがない場合、デフォルト URL の cs_server が自動的に起動します。デフォルトは TCP:localhost:3042 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Hardware (ハードウェア)

説明



重要: このコマンドを使用する前に、open_hw コマンドを使用して Vivado Design Suite でハードウェア マネージャーを開く必要があります。

デザインをプログラムおよびデバッグするため 1 つ以上のサイリンクス デバイスで構成される JTAG チェーンを含むハードウェア ターゲットを開くには、まず Vivado ツール ハードウェア サーバー (hw_server) に接続してハードウェア ターゲット (hw_target) への接続を制御する必要があります。

hw_server は、物理プログラミング ターゲットへの接続を制御します。ハードウェア プログラムまたはテスト ボードにローカルまたはリモートで接続されているマシンで実行する必要があります。hw_server コマンドを別のアプリケーションとして起動する必要があります。このコマンドは、Vivado Design Suite インストール ディレクトリの / bin フォルダーに含まれます。

ハードウェア サーバーに接続するには、hw_server アプリケーションが実行中で、connect_hw_server コマンドの -url オプションでホスト名およびポート番号を指定する必要があります。hw_server プロセスのデフォルト URL は localhost:3121 です。Vivado ハードウェア サーバーの設定および実行に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

Vivado Design Suite の 1 つのインスタンスを複数のハードウェア サーバーに接続でき、異なるデバイス コンフィギュレーションのプログラムおよびデバッグがサポートされます。ただし、ホスト名とポート番号の組み合わせで指定された特定のハードウェア サーバーに対して開くことのできる接続は 1 つのみです。既に接続されているサーバーに対して接続を開こうとすると、エラーが返されます。

`current_hw_server` コマンドを使用して現在のハードウェア サーバーを変更していない場合、最後に接続されたハードウェア サーバーが現在のハードウェア サーバーとなります。サーバーの接続を解除するには、`disconnect_hw_server` コマンドを使用します。

このコマンドを実行すると、接続されたハードウェア サーバーのホスト名が返されます。

引数

`-url <arg>` (オプション): 実行中の `hw_server` アプリケーションの URL を指定します。URL は `<hostname>:<port_number>` の形式で指定します。デフォルトは `localhost:3121` です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、リモート ホスト `trumpet3` にある `hw_server` にポート 3121 を介して接続しています。

```
connect_hw_server -url trumpet3:3121
```

次の例では、ローカルで実行されている `hw_server` にデフォルト URL で接続しています。

```
connect_hw_server
```

注記: 接続の前に、`hw_server` コマンドを別に実行しておく必要があります。

関連項目

- [current_hw_server](#)
- [disconnect_hw_server](#)
- [get_hw_servers](#)
- [get_hw_targets](#)
- [refresh_hw_server](#)

connect_net

ネットをピンまたはポートに接続します。

構文

```
connect_net [-hierarchical] [-basename <arg>] [-net <args>]
            [-objects <args>] [-net_object_list <args>] [-dict <args>] [-quiet]
            [-verbose]
```

使用法

名前	説明
[-hierarchical]	階層接続を可能にし、必要に応じてネットおよびピンを作成します (-basename を参照)。
[-basename]	階層接続を実行する場合に必要なネット/ピンのベース名を指定します (-hierarchical を参照)。デフォルト値は、接続されるネットの名前に基づきます (-net を参照)。
[-net]	指定したオブジェクトに接続するネットを指定します。
[-objects]	接続するピンおよびポート オブジェクトを 1 つまたは複数指定します。
[-net_object_list]	ネットとピン/ポート ペアのリストを指定します。ピンまたはポート リストの各エレメントが対応するネットに接続されます。-net オプションと共に使用することはできません。-net_object_list オプションを使用している場合、-objects オプションは無視されます。
[-dict]	-net_object_list オプションの代わりに使用します。-net_object_list オプションよりも高速ですが、ネットとピン/ポート (名前またはほかの Tcl オブジェクトではなくオブジェクトで指定) のペアを 1 つまたは複数指定する必要があります。各ピンまたはポートのリスト エレメントが該当するネットに接続されます ({ \$net_1 \$pin_1 \$net_2 \$pin_2 } など)。-net オプションと共に使用することはできません。-dict オプションを使用すると、-objects オプションは無視されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Netlist (ネットリスト)

説明

指定のネットを、開いている合成済みデザインまたはインプリメント済みデザインのネットリストに含まれる 1 つ以上のピンまたはポートに接続します。

connect_net コマンドは、デザインの階層レベルをまたがるネットも接続します。この際、必要に応じてピンおよび階層ネットが追加されます。追加されたネットおよびピンには、識別しやすいようにカスタム ベース名を付けることもできますが、指定しない場合は Vivado ツールで自動的にベース名が割り当てられます。



ヒント: `-net_object_list` または `-dict` オプションを使用すると、1つの `connect_net` コマンドで複数のネット、ピンとポートのリストを指定できるので、複数のネットを追加する操作が大幅に高速化します。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-hierarchical` (オプション): 階層の異なるレベルにあるネットを接続します。



重要: `-hierarchical` オプションを指定しない場合は、階層ピンは接続されず、警告メッセージが表示されます。

`-basename <arg>` (オプション): 階層レベルをまたがる指定のネットを接続するのに必要な階層ネットまたはピンに使用するカスタム名を指定します。このオプションを使用しない場合、自動的に接続されるネットに基づくベース名が付けられます。

`-net <arg>` (必須): 接続するネットを指定します。



ヒント: `create_net` コマンドの `-from` および `-to` オプションを使用してバスを作成できますが、バスの各ビットを `connect_net` コマンドを使用してそれぞれ接続する必要があります。

`-objects <args>` (必須): ネットを接続するピンまたはポートを指定します。ネットは1つ以上のピンまたはポートオブジェクトに接続できます。

`-net_object_list <args>` (オプション): ネットを接続する複数のネット、ピン、およびポートを指定します。このオプションを使用すると、1つの `connect_net` コマンドで複数のネットを接続できます。ネット、ピン、およびポートは、名前で指定するか、または `get_nets`、`get_pins`、および `get_ports` コマンドを使用してオブジェクトとして指定します。ネットとピン/ポートのリストは、`{net1 {pin1 pin2...pinN} net2 {pin1 pin2} ...netN {pin1 pin2...pinN}}` という形式で指定します。



ヒント: `-net_object_list` または `-dict` オプションを使用する場合は、`-net` および `-objects` オプションは使用しないでください。これらのオプションは無視されます。`-net`、`-objects`、`-net_object_list`、`-dict` はすべてオプションと示されていますが、これらのいずれかを使用して接続するネットおよびオブジェクトを指定する必要があります。

`-dict <args>` (オプション): `-net_object_list` オプションとは異なる、パフォーマンスの面で多少有利な構文を使用できるようにします。ネット/ピンまたはネット/ポート オブジェクト ペアは、名前ではなく、オブジェクトで指定する必要があります。次に例を示します。

```
set myNetA [get_nets netA]
set myPin1 [get_pins pin1]
set myPin2 [get_pins pin2]
set myPin3 [get_pins pin3]
connect_net -dict { $myNetA $myPin1 $myNetA $myPin2 $myNetA $myPin3 }
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ポートを作成し、myDMA インスタンスにピンを作成し、myEnable というネットを作成した後、そのネットを作成したポートおよびピンに接続しています。

```
create_port -direction IN enableIn
create_pin -direction IN myDMA/en
create_net myEnable
connect_net -net myEnable -objects {enableIn myDMA/en}
```

次の例では、32 ビットのバス ポート、ピン、およびネットを作成し、`loop` コマンドを使用して各ビットを 1 つずつ接続します。

```
create_port -from 0 -to 31 -direction IN dataIN
create_pin -from 0 -to 31 -direction IN myDMA/data
create_net -from 0 -to 31 dataBus
for {set x 0} {$x<32} {incr x} { \
    connect_net -net dataBus[$x] -objects [list dataIN[$x] myDMA/data[$x]] }
```

注記: `dataBus` を接続しようとすると、ネットが見つからないというエラーが発生します。バスの各ビットを個別に接続する必要があります。

次の例では、新しいセルを作成し、`-net_object_list` オプションを使用して 1 つの `connect_net` コマンドで複数のネットを接続しています。

```
create_cell -ref inv a2_i
connect_net -net_object_list {top_I[2] {I[2] a2_i/I} \
    top_O[2] {O[2] a2_i/O} top_clk a2_i/clk}
```

関連項目

- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [disconnect_net](#)
- [remove_net](#)
- [resize_net_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

convert_ips

指定の IP をコア コンテナ フォーマットに、またはコア コンテナ フォーマットの IP をコア コンテナでないフォーマットに変換します。

構文

```
convert_ips [-force] [-to_core_container] [-from_core_container] [-quiet]
            [-verbose] <objects>
```

使用法

名前	説明
[-force]	IP がロックされている場合でも強制的に変換を実行します。
[-to_core_container]	指定の IP をコア コンテナ フォーマットに変換します。
[-from_core_container]	IP をコア コンテナでないフォーマットに変換します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	変換する IP のオブジェクトを指定します。IP またはソース ファイル オブジェクトを指定します。

カテゴリ

IPFlow (IP フロー)

説明

既存の IP をコア コンテナ フォーマットに、またはコア コンテナ IP をコア コンテナでない展開フォーマットに変換します。

IP のコア コンテナ フォーマットは圧縮された ZIP ファイルで、デザインのファイル構造を削減し、ツール パフォーマンスを向上します。

デフォルトでは、IP はザイリンクス IP カタログからコア コンテナ フォーマットを使用してデザインに追加されます。convert_ips コマンドを使用すると、既存のデザインに含まれる IP を変換してコア コンテナ フォーマットの利点を活かすことができます。convert_ips コマンドでは、圧縮されたコア コンテナ フォーマットをコア コンテナでない展開フォーマットの IP に戻すこともできます。



ヒント: -to_core_container または -from_core_container オプションのいずれも指定しない場合は、convert_ips コマンドで IP が現在のフォーマットから逆のフォーマットに変換されます。つまり、コア コンテナ フォーマットの IP はコア コンテナでないフォーマットに、コア コンテナでないフォーマットの IP はコア コンテナ フォーマットに変換されます。

ユーザー管理の IP は、フォーマットを変換することはできません。ロックされている IP を変換するには、-force オプションを指定する必要があります。IP の編集および IS_LOCKED と IS_MANAGED プロパティの詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896)を参照してください。

このコマンドを実行すると、実行されたアクションが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): ロックされている IP (IS_LOCKED) を強制的に変換します。

`-to_core_container` (オプション): 展開フォーマットの IP をコア コンテナ フォーマットに変換します。指定した IP が既にコア コンテナ フォーマットである場合は、その IP は無視されます。

`-from_core_container` (オプション): コア コンテナ フォーマットの IP をコア コンテナでない展開フォーマットに変換します。指定した IP が既にコア コンテナでないフォーマットである場合は、その IP は無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 変換する IP オブジェクトまたは IP ファイルを指定します。IP ファイルは `get_files` コマンドを使用して指定でき、IP オブジェクトは `get_ips` コマンドを使用して指定できます。

例

次の例では、現在のプロジェクトに含まれるすべての IP をコア コンテナ フォーマットに変換しています。

```
convert_ips -to_core_container [get_ips]
```

注記: 既にコア コンテナである IP は無視されます。

次の例では、指定した IP ファイルをコア コンテナ フォーマットに変換しています。

```
convert_ips -to_core_container \  
[get_files C:/Data/wave1/wave1.srscs/sources_1/ip/char_fifo/char_fifo.xci]
```

次の例では、デザインに含まれるすべての IP を現在のフォーマットから変換します。コア コンテナ フォーマットの IP はコア コンテナでないフォーマットに、コア コンテナでないフォーマットの IP はコア コンテナ フォーマットに変換されます。

```
convert_ips [get_ips]
```

関連項目

- [get_files](#)
- [get_ips](#)
- [get_property](#)
- [set_property](#)

convert_ngc

NGC ファイルをサポートされるフォーマットに変換します。

構文

```
convert_ngc [-output_dir <arg>] [-format <arg>] [-add_to_project] [-force]
            [-quiet] [-verbose] <files>
```

戻り値

なし

使用法

名前	説明
<code>[-output_dir]</code>	すべての出力を配置するディレクトリをしています。デフォルトでは、NGC ファイルのディレクトリに配置されます。
<code>[-format]</code>	出力フォーマットを指定します。有効な値は Verilog または EDIF で、デフォルトは EDIF です。
<code>[-add_to_project]</code>	出力ファイルを現在のプロジェクトに追加します。プロジェクトが開いていない場合は、このオプションは無視されます。
<code>[-force]</code>	ディスクに既に存在するファイルを上書きし、add_to_project オプションが指定されている場合はプロジェクトのファイルを置き換えます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	変換する NGC ファイルを指定します。

カテゴリ

[xilinx_tclstore](#) (ザイリンクス Tcl Store)、[projutils](#) (プロジェクト ユーティリティ)

説明

NGC ファイルをサポートされるフォーマットに変換します。

引数

`-output_dir <arg>` (オプション): 出力を保存するディレクトリを指定します。指定しない場合、出力は入力 NGC ファイルと同じディレクトリに配置されます。

`-format [Verilog | EDIF]` (オプション): 出力フォーマットを指定します。デフォルトは EDIF です。

`-add_to_project` (オプション): 出力ファイルを現在のプロジェクトに追加します。プロジェクトが開いていない場合は、このオプションは無視されます。

`-force` (オプション): ディスクに既に存在するファイルを上書きし、`-add_to_project` オプションが指定されている場合は現在のプロジェクトのファイルを置き換えます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 変換する NGC ファイルを指定します。

例

次の例では、`test.ngc` を `test.edn` に変換し、`-verbose` オプションですべてのメッセージを表示しています。`test.edn` ファイルが現在開いているプロジェクトに追加されます。

```
convert_ngc ./test.ngc -add_to_project -verbose
```

次の例では、`test.ngc` を `test.edn` に変換しています。`test.edn` ファイルは、`./output` ディレクトリに保存されます。`./output/test.edn` が既に存在する場合は、上書きされます。

```
convert_ngc ./test.ngc -output_dir output -force
```

次の例では、現在のディレクトリおよびサブディレクトリにあるすべての NGC ファイルを変換しています。

```
convert_ngc [ glob ./**/*.ngc ] [ glob ./*.ngc ]
```

関連項目

- [add_files](#)
- [current_project](#)
- [get_files](#)
- [read_edif](#)

copy_bd_objs

オブジェクトのコピーを作成し、指定の階層セルに追加します。

構文

```
copy_bd_objs [-prefix <arg>] [-from_design <arg>] [-quiet] [-verbose]
             <parent_cell> <objects>...
```

戻り値

正しく処理された場合は 0、エラーが発生した場合は ""。

使用法

名前	説明
[-prefix]	セルに追加する接頭辞を指定します。
[-from_design]	元のオブジェクトを含むデザインを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<parent_cell>	親セルを指定します。
<objects>	コピーするオブジェクトを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

開いているサブシステム デザインの IP インテグレーター オブジェクトを 2 番目のサブシステム デザインにコピーします。選択したオブジェクトは、現在のサブシステム デザインの最上位または既存の階層モジュール内にコピーできます。

get_bd_cells などのコマンドは現在のサブシステム デザインに対して実行されるので、下の例に示すように、コピーするオブジェクトを Tcl 変数に保存する必要があります。現在のサブシステム デザインをソース デザインに設定し、コピーするオブジェクトのグループを選択して、Tcl 変数に保存します。その後、current_bd_design コマンドを使用して現在のサブシステム デザインをターゲット デザインに変更し、選択したオブジェクトをコピーします。この場合、-from_design オプションを使用する必要があります。

このコマンドは、現在のサブシステム デザインのある階層レベルにあるオブジェクトを別の階層レベルにコピーするためにも使用できます。この場合は、-from_design オプションを使用する必要はありません。

このコマンドが正常に実行された場合は 0 が返され、正常に実行されなかった場合はエラーが返されます。

引数

-prefix <arg> (オプション): 階層モジュールにコピーするセルに適用する接頭辞を指定します。

`-from_design <arg>` (オプション): 指定するオブジェクトが存在する IP インテグレーター サブシステム デザイン の名前を指定します。デザインは IP インテグレーターで開いておく必要があります。`-from_design` オプションを指定しない場合、オブジェクトは `current_bd_design` コマンドで指定されている現在のデザインで検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<parent_cell>` (必須): 指定のオブジェクトのコピー先となる階層モジュールの名前を指定します。現在のサブシステム デザインの最上位には `/` を指定できます。

`<objects>` (必須): 指定の `<parent_cell>` にコピーするセルおよびネットを `get_bd_cells` および `get_bd_nets` コマンドで指定します。

例

次の例では、現在のサブシステム デザインを設定し、オブジェクトのグループを選択して、Tcl 変数に保存した後、現在のサブシステム デザインを変更し、選択したオブジェクトを現在のデザインの最上位にコピーしています。

```
current_bd_design myDesign
myDesign
set copyObjs [get_bd_cells {vidOut_1 bridge_1}]
/vidOut_1 /bridge_1
current_bd_design design_1
design_1
copy_bd_objs -from_design myDesign / $copyObjs
0
```

注記: `get_bd_cells` コマンドでは現在のサブシステム デザインからのセルのみが返されるので、`current_bd_design` でターゲット デザインを変更する前に、Tcl 変数にこれらのオブジェクトを保存します。

関連項目

- [current_bd_design](#)
- [get_bd_cells](#)
- [get_bd_nets](#)

copy_ip

既存の IP をコピーします。

構文

```
copy_ip -name <arg> [-dir <arg>] [-quiet] [-verbose] <objects>...
```

戻り値

プロジェクトに追加された IP ファイル オブジェクト

使用法

名前	説明
-name	コピーの IP の名前を指定します。
[-dir]	プロジェクト外で作成し、管理するリモート IP へのディレクトリ パスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	コピーする IP を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

現在のプロジェクトにインスタンス化されている IP コアのコピーを作成します。

引数

-name <arg> (必須): 作成する新しい IP の名前を指定します。

-dir <arg> (オプション): 新しく作成した IP を保存する、ローカル プロジェクト外のディレクトリへのパスを指定します。指定するディレクトリは、既に存在している必要があります。ディレクトリが存在しない場合は、エラーが返されます。

注記: ディレクトリを指定しない場合、新しい IP はローカル プロジェクト ディレクトリ構造の <project_name>.srcs/sources_1/ip に追加されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<object> (必須): コピーする IP オブジェクトを指定します。`copy_ip` コマンドでは、一度に 1 つの IP コアのみコピーできます。IP は `get_ips` コマンドで指定する必要があります。名前で指定することはできません。

例

次の例では、現在のプロジェクトにインスタンス化されている FIFO コアをコピーし、指定のディレクトリに保存しています。

```
copy_ip -name newFIFO -dir C:/Data/new_IP [get_ips oldFIFO]
```

関連項目

- [create_ip](#)
- [get_ips](#)
- [import_ip](#)
- [read_ip](#)

copy_run

既存の run (ソース run) から run をコピーし、その run の新しいコピー (デスティネーション run) を作成します。

構文

```
copy_run [-parent_run <arg>] [-verbose] -name <arg> [-quiet] <run>
```

戻り値

新しい run オブジェクト

使用法

名前	説明
[-parent_run]	新しいインプリメンテーション run 用の合成 run を指定します。名前または run オブジェクトを指定できます。デフォルトは、ソース run と同じです。
[-verbose]	コピー実行中の詳細を表示します。
-name	新しい run の名前を入力します。
[-quiet]	コマンド エラーを表示しません。
<run>	コピーする run を指定します。名前または run オブジェクトを指定できます。

カテゴリ

[xilinxclstore](#) (ザイリンクス Tcl Store)、[projutils](#) (プロジェクト ユーティリティ)

説明

既存の合成またはインプリメンテーション run をコピーします。

引数

-name <arg> (オプション): 新しい run の名前を指定します。

-parent_run <arg> (オプション): 新しいインプリメンテーション run の合成 run を指定します。run 名で指定するか、または get_runs または current_run コマンドでオブジェクトとして返すことにより指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<run> (必須): コピーする run を指定します。run 名で指定するか、または `get_runs` または `current_run` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、`synth_1` という run をコピーして新しく `synth_2` という run を作成しています。

```
copy_run -name synth_2 [get_runs synth_1]
```

次の例では、`impl_1` という run をコピーして新しく `impl_2` という run を作成し、新しい run の親として `synth_2` を指定しています。

```
copy_run -name impl_2 [get_runs impl_1] -parent_run synth_2
```

関連項目

- [current_run](#)
- [get_runs](#)

create_bd_addr_seg

新しいセグメントを作成します。

構文

```
create_bd_addr_seg -range <arg> -offset <arg> [-quiet] [-verbose]
                  [<parent_addr_space>] [<slave_segment>] <name>
```

戻り値

新しく作成されたセグメント オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
-range	セグメントの範囲 (4096、4K、16M、1G など) を指定します。
-offset	セグメントのオフセット (0x00000000 など) を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<parent_addr_space>]	セグメントの親アドレス空間を指定します。
[<slave_segment>]	作成したセグメントのスレーブ セグメントを指定します。
<name>	作成するセグメントの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに新しいアドレス セグメント オブジェクト (bd_addr_seg) を作成します。

このコマンドを実行すると、新しく作成されたマスター アドレス セグメント オブジェクトが返されるか、正常に実行されなかった場合は何も返されません。

引数

-range <arg> (必須): 作成するアドレス セグメントの範囲またはサイズを整数または 16 進数値で指定します。範囲は 2 のべき数で表されたビット数 (4096 など) または割り当てるメモリ量 (4K、16M、1G など) で指定します。

-offset <arg> (必須): アドレス セグメントのオフセットを指定します。整数または 16 進数値 (0x00000000 など) で指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<parent_addr_space> (必須): セグメントの親アドレス空間を定義します。

<slave_segment> (必須): 作成するマスター アドレス セグメントのスレーブ アドレス セグメントをマップします。

<name> (必須): 作成するマスター アドレス セグメントの名前を指定します。

例

次の例では、MicroBlaze コアのデータおよび命令アドレス空間のアドレス セグメントを作成しています。

```
create_bd_addr_seg -range 0x10000 -offset 0x41200000 \
    [get_bd_addr_spaces microblaze_1/Data] \
    [get_bd_addr_segs microblaze_1_axi_intc/s_axi/Reg] SEG1

create_bd_addr_seg -range 0x40000000 -offset 0x0 \
    [get_bd_addr_spaces microblaze_1/Instruction] \
    [get_bd_addr_segs microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB/Mem] \
    SEG1
```

関連項目

- [exclude_bd_addr_seg](#)
- [get_bd_addr_segs](#)
- [get_bd_addr_spaces](#)
- [include_bd_addr_seg](#)

create_bd_cell

IP カタログから IP セルを追加するか、新規階層ブロックを作成します。

構文

```
create_bd_cell [-vlnv <arg>] [-type <arg>] [-reference <arg>]
               [-revision <arg>] [-quiet] [-verbose] <name>
```

戻り値

新しく作成されたセル オブジェクト、コマンドが正しく実行されなかった場合はなし。

使用法

名前	説明
[-vlnv]	IP カタログから追加する IP セルの <Vendor>:<Library>:<Name>:<Version> を指定します。
[-type]	作成するセルのタイプを指定します。有効な値は IP、hier、および module で、デフォルトは IP です。
[-reference]	セルを作成する際に参照するモジュールの最上位名またはファイルパスを指定します。
[-revision]	コアのリビジョンを指定します。デフォルトは -1 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するセルの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado カタログからセルを現在のサブシステム デザインに追加するか、サブシステム デザインに追加する新しい階層モジュールを作成するか、HDL ソース ファイルからモジュール定義を参照して新しいモジュールを作成します。

カタログから IP コアを追加する場合は、-vlnv オプションが必要です。

新しい階層ブロック デザイン モジュールを作成する場合は、-type hier オプションが必要です。

RTL モジュールまたはエンティティ 宣言を参照するブロック デザイン モジュールを作成する場合は、-type module オプションと -reference オプションが必要です。モジュール参照機能を使用すると、ブロック デザインに RTL ファイル (Verilog または VHDL) からのモジュール定義を追加できます。モジュール参照を作成する前に、モジュール定義を含むソース ファイルをプロジェクトに追加するか、デザインに読み込んでおく必要があります。モジュール参照の詳細は、『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』(UG994) を参照してください。

このコマンドを実行すると、新しく作成されたセル オブジェクトの名前が返されるか、正常に実行されなかった場合は何も返されません。

引数

`-vlnv <arg>` (必須): このオプションは、`-type IP` (デフォルト) では必須で、`-type hier` ではオプションです。IP インテグレーター カタログから追加するセルの `<Vendor>:<Library>:<Name>:<Version>` 属性を指定します。VLNV 属性は、IP インテグレーター カタログ内のオブジェクトを識別します。新しい階層モジュールを作成する場合は、このオプションは必要ありません。

注記: IP インテグレーター カタログからの IP の `-vlnv` プロパティは、Vivado Design Suite インストール ディレクトリの `data/ip/xilinx` にあるファイルを参照します。

`-type [IP | hier | module]` (オプション): セルのタイプを指定します。

- `IP`: Vivado IP カタログから IP の IP。これは作成されたブロック デザインのデフォルト タイプであり、`-vlnv` オプションが必要です。
- `hier`: 現在のデザインに追加し、作成する新しい空の階層ブロック デザイン モジュール。
- `module`: ソース ファイルセットに読み込まれている Verilog または VHDL ファイルから参照される階層モジュール。参照モジュールを指定する場合は、`-reference` オプションを使用する必要があります。

注記: `-vlnv` および `-type` はオプションですが、いずれかを指定する必要があります。Vivado IP カタログから追加する IP コアを指定する場合は `-vlnv`、新しい階層モジュールを作成または参照する場合は `-type` を使用します。

`-reference <arg>` (オプション): 読み込まれた RTL デザイン ソース ファイルから参照するモジュールの名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: 現在の IP サブシステム デザインに追加する IP セルの名前を指定します。

例

次の例では、IP インテグレーター カタログから AXI FIFO コアを現在のサブシステム デザインに指定した名前で追加しています。

```
create_bd_cell -vlnv xilinx.com:ip:axi_fifo_mm_s:4.0 axi_fifo_1
```

注記: `-vlnv` オプションで Vivado カタログから追加するコアを指定しています。

次の例では、ブロック デザインに新しいモジュールを作成し、読み込み済みの RTL ソース ファイルから指定のモジュール定義を参照しています。

```
create_bd_cell -type module -reference rtlRam rtlRam_0
```

次の例では、新しい階層モジュール myModule1 を作成し、前の例の AXI FIFO を新しいモジュール内に移動します。その後、myModule1 をサブシステム デザインの現在のインスタンスとして設定し、新しいモジュール myModule2 を作成して、現在のインスタンスに追加します。最後に、現在のインスタンスをサブシステム デザインの最上位に戻します。

```
create_bd_cell -type hier myModule1
/myModule1
move_bd_cells /myModule1 [get_bd_cells /axi_fifo_1]
/myModule1
current_bd_instance /myModule1
/myModule1
create_bd_cell -type hier myModule2
/myModule1/myModule2
current_bd_instance
/
```

関連項目

- [copy_bd_objs](#)
- [current_bd_instance](#)
- [move_bd_cells](#)
- [update_module_reference](#)

create_bd_design

新規デザインと、同じ名前の最上位セルを作成します。

構文

```
create_bd_design [-dir <arg>] [-cell <arg>] [-quiet] [-verbose] <name>
```

戻り値

新しく作成されたデザイン オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
[-dir]	プロジェクト外で作成し、管理するリモート BD へのディレクトリ パスを指定します。
[-cell]	サブデザインを新規デザインにコピーする階層セル名を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するデザインの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在のプロジェクトに追加したり、Vivado Design Suite の IP インテグレーターで使用する新規 IP サブシステム デザインを作成します。

空の IP サブシステム モジュールが作成され、現在のプロジェクトのソース ファイルに追加されます。サブシステム モジュールおよびファイルは、指定の <name> で、現在のプロジェクトの次の場所に作成されます。

```
<project_name>/<project_name>.srcs/sources_1/bd/<name>/<name>.bd
```

このコマンドが正常に実行されると、ファイル パスと IP サブシステム デザインの名前が返されます。正常に実行されなかった場合はエラーが返されます。

引数

-dir <arg> (オプション): ブロック デザイン ファイルを保存するディレクトリを指定します。このオプションを使用すると、現在のプロジェクト ディレクトリ構造外で BD ファイルを作成および管理でき、複数のプロジェクト間でブロック デザインを再利用しやすくなります。

-cell <arg> (オプション): 新規ブロック デザインのソースとして使用する階層ブロック デザイン セル (bd_cell) を指定します。既存のデザインの一部から新規ブロック デザインを作成する場合にこのオプションを使用します。セルは、インスタンス名で指定するか、または get_bd_cells コマンドを使用してオブジェクトとして指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: 作成する IP サブシステム デザインの名前を指定します。

例

次の例では、`design_1` という新しい空の IP サブシステム モジュールを作成して現在のプロジェクトに作成し、プロジェクトの `sources` ディレクトリに `design_1.bd` というファイルを作成しています。

```
create_bd_design design_1
```

関連項目

- [close_bd_design](#)
- [current_bd_design](#)
- [open_bd_design](#)
- [save_bd_design](#)

create_bd_intf_net

新規インターフェイス ネット (intf_net) を作成します。

構文

```
create_bd_intf_net [-quiet] [-verbose] <name>
```

戻り値

新しく作成された intf_net オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	作成する intf_net の名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP サブシステム デザインの新しい IP インテグレーター インターフェイスを作成します。

このコマンドを実行すると、新しく作成されたインターフェイス ネット オブジェクトが返されるか、正常に実行されなかった場合は何も返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: 作成するインターフェイス ネットの名前を指定します。

例

次の例では、現在のサブシステム デザインにインターフェイス ネットを作成しています。

```
create_bd_intf_net diff_clock_rtl
```

関連項目

- [connect_bd_intf_net](#)

create_bd_intf_pin

新規インターフェイス ピン (intf_pin) を作成します。

構文

```
create_bd_intf_pin -vlnv <arg> -mode <arg> [-quiet] [-verbose] <name>
```

戻り値

新しく作成された intf_pin オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
-vlnv	バスの VLNv を指定します。
-mode	バス インターフェイスのモードを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するインターフェイス ピンの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター階層モジュールに新しいインターフェイス ピンを作成します。IP インテグレーター インターフェイスとは、共通の機能を共有する信号をグループ化したもので、関連の機能を共有する個別の信号およびバスの両方を含めることができます。たとえば、AXI4-Lite マスターは、多数の個別の信号と複数のバスを含むインターフェイスです。

1 つの接続ピンまたは標準バス ピンを作成するには、create_bd_pin コマンドを使用します。

インターフェイス ピンは、互換性のあるほかのインターフェイス ピンまたはインターフェイス ポートに接続します。インターフェイス ピンは、階層モジュール内ではモジュールの外部に接続するためポートとして追加され、階層モジュール上ではピンとして追加されます。

current_bd_instance コマンドを使用して、階層モジュールを IP インテグレーター サブシステム デザインの現在のインスタンスとして指定する必要があります。create_bd_intf_pin コマンドは、現在のインスタンスに対して実行されます。

このコマンドを実行すると、新しく作成されたインターフェイス ピン オブジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-vlnv <arg>` (必須): サブシステム デザインに追加するインターフェイス ピン オブジェクトの `<Vendor>:<Library>:<Name>:<Version>` (VLVN) 属性を指定します。VLNV 属性は、IP インテグレーター カタログ内のオブジェクトを識別します。

注記: インターフェイス ピンおよびポートの `-vlnv` プロパティは、Vivado Design Suite インストール ディレクトリにあるファイルを参照します。たとえば、`-vlnv xilinx.com:interface:lmb_rtl:1.0` は Vivado Design Suite インストール ディレクトリの `data/ip/interfaces/lmb_v1_0` にあります。

`-mode <arg>` (必須): インターフェイス ピンのモードを指定します。有効な値は Master、Slave、System、MirroredMaster、MirroredSlave、MirroredSystem、Monitor です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 現在のインスタンスに追加するインターフェイス ピンの名前を指定します。

例

次の例では、階層モジュール `newMod1` を IP インテグレーター サブシステム デザインの現在のインスタンスとして設定し、そのモジュールに新しいインターフェイス ピンを作成しています。

```
current_bd_instance [get_bd_cells /newMod1]
create_bd_intf_pin -mode Slave -vlnv xilinx.com:user:dma_rtl:1.0 data_in
```

関連項目

- [connect_bd_intf_net](#)
- [create_bd_port](#)
- [get_bd_intf_ports](#)

create_bd_intf_port

新規インターフェイス ポートを作成します。

構文

```
create_bd_intf_port -vlnv <arg> -mode <arg> [-board_intf <arg>] [-quiet]
[-verbose] <name>
```

戻り値

新しく作成されたインターフェイス ポート オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
-vlnv	バスの VLNv を指定します。
-mode	バス インターフェイスのモードを指定します。
[-board_intf]	現在のボードの指定したインターフェイスを使用して、このオプションで指定した外部バス インターフェイスのポート マップを作成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するポートの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP サブシステム デザインの新しいインターフェイス ポートを作成します。IP インテグレーター インターフェイスとは、共通の機能を共有する信号をグループ化したもので、関連の機能を共有する個別の信号およびバスの両方を含めることができます。たとえば、AXI4-Lite マスターは、多数の個別の信号と複数のバスを含むインターフェイスです。

1 つの接続ポートまたは共通バス ポートを作成するには、`create_bd_port` コマンドを使用します。

このコマンドを実行すると、新しく作成されたインターフェイス ポート オブジェクトの名前が返されるか、正常に実行されなかった場合は何も返されません。

引数

-vlnv <arg> (必須): サブシステム デザインに追加するインターフェイス ポート オブジェクトの <Vendor>:<Library>:<Name>:<Version> (VLNV) 属性を指定します。VLNV 属性は、IP インテグレーター カタログ内のオブジェクトを識別します。

注記: インターフェイス ピンおよびポートの `-vlnv` プロパティは、Vivado Design Suite インストール ディレクトリの `./data/ip/interfaces` にあるファイルを参照します。たとえば、`-vlnv xilinx.com:interface:lmb_rtl:1.0` は Vivado Design Suite インストール ディレクトリの `data/ip/interfaces/lmb_v1_0` にあります。

`-mode <arg>` (必須): インターフェイス ピンのモードを指定します。有効な値は Master、Slave、System、MirroredMaster、MirroredSlave、MirroredSystem、Monitor です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): サブシステム デザインに追加するインターフェイス ポートの名前を指定します。

例

次の例では、新しい IP インテグレーター インターフェイス ポートを作成し、現在のサブシステム デザインに追加しています。

```
create_bd_intf_port -vlnv xilinx.com:interface:diff_clock_rtl:1.0 \
    -mode Slave diff_clock_rtl
```

関連項目

- [connect_bd_intf_net](#)
- [create_bd_intf_pin](#)
- [create_bd_port](#)
- [get_bd_intf_ports](#)

create_bd_intf_tlm_port

新しい TLM インターフェイス ポートを作成します。

構文

```
create_bd_intf_tlm_port -vlnv <arg> -mode <arg> [-quiet] [-verbose] <name>
```

戻り値

新しく作成された TLM インターフェイス ポート オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
-vlnv	TLM インターフェイスの VLNv を指定します。
-mode	TLM インターフェイスのモードを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	TLM インターフェイスの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

create_bd_net

新規ネットを作成します。

構文

```
create_bd_net [-quiet] [-verbose] <name>
```

戻り値

新しく作成されたネット オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	作成するネットの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに新しいネットを作成します。

このコマンドを実行すると、新しく作成されたネット オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成するネットの名前を指定します。ネット名は最上位から指定するか、名前のみ (net1) で指定するか、階層ネット名 (cell1/cellA/net1) を指定してデザイン階層内で指定できます。

例

次の例では、新しいネットを作成しています。

```
create_bd_net net1
```

関連項目

- [connect_bd_intf_net](#)
- [connect_bd_net](#)
- [create_bd_cell](#)
- [create_bd_intf_net](#)
- [create_bd_intf_pin](#)
- [create_bd_intf_port](#)
- [create_bd_pin](#)
- [create_bd_port](#)
- [current_bd_design](#)

create_bd_pin

新規ピンを作成します。

構文

```
create_bd_pin [-from <arg>] [-to <arg>] -dir <arg> [-type <arg>] [-quiet]
              [-verbose] <name>
```

戻り値

新しく作成されたピン オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
[-from]	開始インデックスを指定します。デフォルトは指定なしです。
[-to]	終了インデックスを指定します。デフォルトは指定なしです。
-dir	ピンの方向を指定します。
[-type]	ピンのタイプを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するピンの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター階層モジュールに追加する新しいピンを作成します。

このコマンドを実行すると、新しく作成されたピン オブジェクトの名前が返されるか、正常に実行されなかった場合はエラー メッセージが返されます。

引数

-from <arg> (オプション): 標準バス ピンの開始インデックスを指定します。1 ビット ピンの場合は指定しません。

-to <arg> (オプション): 標準バス ピンの終了インデックスを指定します。1 ビット ピンの場合は指定しません。

-dir [I | O | IO] (必須): ピンの方向を指定します。入力ピンの場合は I、出力ピンの場合は O、双方向ピンの場合は IO に設定します。

-type <arg> (オプション): ピンのタイプを指定します。クロック ピンの場合は CLK、リセット ピンの場合は RST、クロック イネーブル ピンの場合は CE、割り込みピンの場合は INTR、データ ピンの場合は DATA に設定します。ピンのタイプを指定しない場合、未定義 (UNDEF) になります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 作成するサブシステムのピンの名前を指定します。名前は、ピンを追加する階層モジュールを基準に参照されます (/modName/pinname)。

例

次の例では、現在の IP インテグレーター サブシステム デザインの指定したモジュールに新しい入力ピンを作成しています。

```
create_bd_pin -dir I -type rst /newMod1/rst
/newMod1/rst
```

関連項目

- [create_bd_cell](#)
- [create_bd_intf_pin](#)
- [create_bd_intf_port](#)
- [create_bd_port](#)

create_bd_port

IP サブシステム デザインの新しいポートを作成します。

構文

```
create_bd_port [-from <arg>] [-to <arg>] -dir <arg> [-type <arg>]
               [-freq_hz <arg>] [-quiet] [-verbose] <name>
```

戻り値

新しく作成されたポート オブジェクト、コマンドが正しく実行されなかった場合はなし。

使用法

名前	説明
[-from]	開始インデックスを指定します。デフォルトは指定なしです。
[-to]	終了インデックスを指定します。デフォルトは指定なしです。
-dir	ポートの方向を指定します。有効な値は I、O、IO です。
[-type]	ポートのタイプを指定します。有効な値は clk、ce、rst、intr、data です。
[-freq_hz]	クロック ポートの周波数を Hz で指定します。デフォルトでは指定されません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するポートの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター サブシステム デザインに追加する新しいポートを作成します。ポートは、サブシステム デザインの外部信号への接続です。

このコマンドを実行すると、新しく作成されたポート オブジェクトの名前が返されるか、正常に実行されなかった場合はエラー メッセージが返されます。

引数

-from <arg> (オプション): 標準バス ポートの開始インデックスを指定します。1 ビット ポートの場合は指定しません。

-to <arg> (オプション): 標準バス ポートの終了インデックスを指定します。1 ビット ポートの場合は指定しません。

-dir [I | O | IO] (必須): ポートの方向を指定します。入力ポートの場合は I、出力ポートの場合は O、双方向ポートの場合は IO に設定します。

`-type <arg>` (オプション): ポートのタイプを指定します。クロック ポートの場合は CLK、リセット ポートの場合は RST、クロック イネーブル ポートの場合は CE、割り込みポートの場合は INTR、データ ポートの場合は DATA に設定します。ポートのタイプを指定しない場合、未定義 (UNDEF) になります。

`-freq_hz <arg>` (オプション): `-type` オプションで定義されたクロック ポートの周波数を指定します。単位は Hz です。たとえば 150 MHz クロックの場合は 150000000 を指定します。指定しない場合、クロック周波数は 100 MHz に設定されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成するサブシステムのポートの名前を指定します。

例

次の例は、現在の IP インテグレーター サブシステム デザインに新しい双方向バス ポートを作成します。

```
create_bd_port -from 0 -to 32 -dir IO -type data addr
/addr
```

次の例は、125 MHz のクロック ポートを作成します。

```
create_bd_port -dir I -type clk -freq_hz 125000000 my_clock
```

関連項目

- [create_bd_cell](#)
- [create_bd_intf_pin](#)
- [create_bd_intf_port](#)

create_bd_tlm_port

IP サブシステム デザインの新しい TLM ポートを作成します。

構文

```
create_bd_tlm_port [-quiet] [-verbose] <name>
```

戻り値

新しく作成された TLM ポート オブジェクト。コマンドが正しく実行されなかった場合はなし。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	作成するポートの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

create_cell

現在のデザインにセルを作成します。

構文

```
create_cell -reference <arg> [-black_box] [-quiet] [-verbose] <cells>...
```

使用法

名前	説明
-reference	セルが参照するライブラリ セルまたはデザインを指定します。
[-black_box]	ブラック ボックス インスタンスを作成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<cells>	作成するセルの名前を指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

現在の合成済みデザインまたはインプリメント済みデザインのネットリストにセルを追加します。

注記: セルをライブラリ マクロまたはマクロ プリミティブに追加することはできません。

新しいセル インスタンスは、デザインの最上位、またはデザインのモジュール内の階層に追加できます。インスタンスは、ライブラリまたはデザイン ソース ファイルから既存のセルを参照するか、作成されていないセルを参照するブラック ボックス インスタンスを追加することもできます。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

このコマンドを実行すると、作成したセル インスタンスの名前が返されます。

引数

`-reference <arg>` (必須): 新しく作成するセル インスタンスが参照するライブラリ セルまたはソース ファイル モジュールを指定します。

`-black_box` (オプション): 指定した参照セルのブラック ボックス インスタンスを定義します。このオプションは、トップダウン設計手法において、参照セルがまだ存在しておらず、セルのブラック ボックス インスタンスを作成する場合に使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<cells> (必須): 作成するセルのインスタンス名を指定します。インスタンス名は、デザインの最上位からの階層名で指定できます。この場合、階層インスタンス名に階層区切り文字を使用する必要があります。現在の階層区切り文字を確認するには、`get_hierarchy_separator` コマンドを使用します。

例

次の例では、`or1200_cpu` モジュールの 3 つのセル インスタンスを指定の名前で作成しています。

```
create_cell -reference or1200_cpu myCell11 myCell12 myCell13
```

次の例では、まず階層区切り文字を設定し、階層インスタンス名を指定して参照セルのブラック ボックス インスタンスを作成しています。

```
set_hierarchy_separator |
create_cell -reference dmaBlock -black_box usbEngine0|myDMA
```

注記: `-black_box` を使用しているときに `-reference` セルが既に存在していると、エラーが返されます。

関連項目

- [connect_net](#)
- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [remove_cell](#)
- [rename_cell](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

create_clock

クロック オブジェクトを作成します。

構文

```
create_clock -period <arg> [-name <arg>] [-waveform <args>] [-add] [-quiet]
           [-verbose] [<objects>]
```

戻り値

新しいクロック オブジェクト

使用法

名前	説明
-period	クロック周期を指定します。0 より大きい値を設定します。
[-name]	クロック名を指定します。
[-waveform]	クロック エッジを指定します。
[-add]	ソース オブジェクトの既存クロックに追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	クロック ソース ポート、ピン、またはネットを指定します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

ナノ秒 (ns) で定義された指定の周期または波形でクロック オブジェクトを作成します。このコマンドは、タイミングエンジンでクロック エッジの遅延伝搬の始点として使用されるプライマリ クロックを定義します。定義したクロックは、既存のクロックの定義に追加したり、既存のクロックを上書きできます。

デザイン内にソースがない仮想クロックも作成できます。仮想クロックは、入力および出力遅延を設定する際の時間の基準として使用できますが、デザインに物理的には存在しません。

クロックは、既存の物理クロックから、そのマスター クロックのプロパティの多くを継承して生成することも可能です。既存の物理クロックからクロックを生成するには、`create_generated_clock` コマンドを使用します。



重要: `create_generated_clock` コマンドではなく `create_clock` コマンドを使用して生成クロックを作成すると、作成されたクロックにソース クロックのプロパティが継承されません。親クロックの挿入遅延およびジッターが生成クロックに伝搬されず、タイミングが正しく算出されません。

`create_clock` コマンドを実行すると、作成したクロック オブジェクトの名前が返されます。

引数

`-period <arg>` (必須): 作成するクロック オブジェクトのクロック周期を指定します。値はナノ秒 (ns) で指定し、0 より大きい値である必要があります。

`-name <arg>` (オプション): 作成するクロック オブジェクトの名前を指定します。名前を指定しない場合は、指定されたソース オブジェクト (<objects>) に基づいてシステムにより名前が生成されます。ソース オブジェクト <objects> を指定せずに `-name` オプションを使用すると、デザインの物理的なソースに関連付けられていない仮想クロックを作成できます。

`-waveform <arg1 arg2 ...>` (オプション): 作成するクロックの、1 クロック サイクル内での立ち上がりエッジと立ち下がりエッジの時間をナノ秒で指定します。複数の立ち上がりエッジと立ち下がりエッジを使用して波形の特性を定義できますが、波形の立ち上がりエッジと立ち下がりエッジの両方を表すエッジを偶数個指定する必要があります。最初に指定した時間 (`arg1`) が最初の立ち上がり遷移、2 番目に指定した時間 (`arg2`) が立ち下がりエッジとなります。立ち下がりエッジの値が立ち上がりエッジの値より小さい場合、立ち下がりエッジが立ち上がりエッジの前に発生することを意味します。

注記: 波形を指定しない場合、デフォルトの波形は立ち上がりエッジが時間 0.0、立ち下がりエッジが指定した周期の $1/2$ (`-period/2`) で発生するよう設定されます。

`-add` (オプション): 異なるクロック波形で同時解析を実行する場合に、同じソースに複数のクロックを定義します。追加するクロックを `-name` を使用して指定する必要があります。このオプションを使用しないと、`create_clock` コマンドで同じ名前のクロックが存在する場合に上書きされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (オプション): 指定したクロックのソースとなるポート、ピン、またはネットを指定します。既にクロックが指定されているソース オブジェクトにクロックを指定した場合、`-add` オプションを使用しなければ元のクロックが新しいクロックで上書きされます。クロック オブジェクトを接続する <objects> を指定しない場合、クロックは仮想クロックとして作成されます。

注記: 指定したネットの最初のドライバーがクロックのソースとして使用されます。

例

次の例では、`bftClk` という物理クロックを作成し、クロック周期を設定しています。

```
create_clock -name bftClk -period 5.000 [get_ports bftClk]
```

注記: この例でオブジェクトを定義する `get_ports` コマンドを使用しない場合、物理クロックではなく仮想クロックが作成されます。

次の例では、入力ポート bftClk に周期が 10 ns、立ち上がりエッジが 2.4 ns、立ち下がりエッジが 7.4 ns の clk というクロックを作成しています。

```
create_clock -name clk -period 10.000 -waveform {2.4 7.4} [get_ports bftClk]
```

次の例では、クロック ソースを指定していないので、仮想クロックが作成されます。

```
create_clock -name virtual_clock -period 5.000
```

次の例では、立ち下がりエッジが 2 ns、立ち上がりエッジが 7 ns のクロックを作成しています。

```
create_clock -name clk -period 10.000 -waveform {7 2} [get_ports bftClk]
```

注記: -waveform オプションで定義されている立ち下がりエッジは立ち上がりエッジより前の時間なので、arg2 として指定されていても波形では最初になります。

関連項目

- [all_clocks](#)
- [create_generated_clock](#)
- [get_clocks](#)
- [report_clocks](#)
- [report_clock_interaction](#)
- [report_clock_networks](#)
- [report_clock_utilization](#)
- [set_clock_groups](#)
- [set_clock_latency](#)
- [set_clock_uncertainty](#)
- [set_input_delay](#)
- [set_output_delay](#)
- [set_propagated_clock](#)

create_dashboard_gadget

プロジェクト サマリ ダッシュボードのガジェットを作成します。

構文

```
create_dashboard_gadget -name <arg> -type <arg> [-quiet] [-verbose]
```

使用法

名前	説明
-name	ガジェットの名前を指定します。
-type	ガジェットのタイプを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明



重要: このコマンドは、Vivado IDE のプロジェクト サマリで使用するためのもので、[Add Gadget] コマンドと同じ操作を実行します。

合成済みまたはインプリメント済みデザイン run の異なる情報を表示するプロジェクト サマリ ダッシュボードなどのダッシュボードに新しいガジェットを作成します。表示できるデザインの情報には、タイミング情報、リソース利用率、DRC および設計手法違反、消費電力解析などがあります。

ダッシュボード ガジェットには、表示する情報およびその表示方法を指定するプロパティがあります。これらのプロパティは、set_property コマンドを使用して設定できます。ガジェットのプロパティを確認するには、次のコマンドを使用します。

```
report_property -all [get_dashboard_gadget <gadget_name>]
```

ダッシュボード ガジェットのプロパティは、set_property コマンドを使用して設定できます。ガジェットに設定可能なプロパティの一部を次に示します。

- TYPE: ガジェットに表示する情報のタイプを指定します。-type オプションで定義します。
- ROW: ダッシュボード内でガジェットを配置する行を指定します。
- COL: ダッシュボード内でガジェットを配置する列を指定します。
- REPORTS: ガジェットに関連付けるレポートを指定します。
- RUN.STEP および TYPE: ガジェットに表示する合成またはインプリメンテーション run の段階を指定します。
- VIEW.TYPE および ORIENTATION: 情報をグラフまたは表のどちらで表示するか、およびデータの向きを指定します。

引数

-name <arg> (必須): 作成するするガジェットの名前を指定します。

-type <arg> (必須): 作成するするダッシュボード ガジェットのタイプを指定します。指定可能なタイプは、DRC、Methodology、Power、Timing、および Utilization です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、消費電力ガジェットを作成し、ガジェットに表示するレポートおよび run の段階を指定しています。

```
create_dashboard_gadget -name gadget_power -type power
set_property reports {impl_1#impl_1_route_report_power_0}
[get_dashboard_gadgets [list {gadget_power}]]
set_property active_reports {impl_1#impl_1_route_report_power_0}
[get_dashboard_gadgets [list {gadget_power}]]
set_property run.step all_stages [get_dashboard_gadgets [list
{gadget_power}]]
```

関連項目

- [delete_dashboard_gadgets](#)
- [get_dashboard_gadgets](#)
- [move_dashboard_gadget](#)

create_debug_core

新しい Integrated Logic Analyzer デバッグ コアを作成します。

構文

```
create_debug_core [-quiet] [-verbose] <name> <type>
```

戻り値

新しいデバッグ コア オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	新しいデバッグ コア インスタンスの名前を指定します。
<code><type></code>	新しいデバッグ コア インスタンスのタイプを指定します。

カテゴリ

[Debug \(デバッグ\)](#)、[XDC](#)

説明

新しい Integrated Logic Analyzer (ILA) デバッグ コアを作成し、現在のプロジェクトのネットリスト デザインに追加します。ILA デバッグ コアは、`open_hw` コマンドを使用して起動できる Vivado Design Suite のロジック解析機能でデザインのデバッグ用にネットを接続するポートを定義します。

ILA デバッグ コアは、ザイリンクス IP カタログからデザインの RTL ソース ファイルに追加するか、合成後にこのコマンドを使用してネットリストに追加できます。ILA デバッグ コアの使用の詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

注記: デバッグ コアは、開いているネットリスト デザインにのみ追加できます。

デフォルトでは、ILA コアは CLK ポートおよび PROBE ポートと共に作成されます。CLK ポートは ILA コアのクロックドメインを定義し、そのドメインに共通の信号をプローブできます。CLK ポートでサポートされるクロック信号は 1 つだけなので、クロック ドメインごとに別のデバッグ コアを作成する必要があります。PROBE ポートは、MARK_DEBUG プロパティでデバッグ用にマークされたネットのプローブ ポイントを提供します。PROBE ポートには複数のチャンネルがあり、1 つの ILA コアから複数のネットをプローブできます。

`create_debug_port` コマンドを使用して既存の ILA コアに新しいポートを追加し、`connect_debug_port` コマンドを使用してそのポートに信号を接続できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): プロジェクトに追加する ILA デバッグ コアの名前を指定します。

`<type>` (必須): 挿入するデバッグ コアのタイプを指定します。Vivado ツールで現在サポートされているのは ILA デバッグ コアのみです。ILA デバッグ コアは接続されたネットに別のロードを追加するだけで、その他の変更はありません。

注記: ILA コアをプロジェクトに追加すると、1 つまたは複数の ILA コアのコンテナとして Debug Hub コア (`labtools_xsdbm_v1`) が 1 つ追加されます。ただし、Debug Hub コアをプロジェクトに直接追加することはできません。

例

次の例では、合成 run を開き、指定のネットリスト デザイン名を作成して、そのデザインに新しい ILA デバッグ コアを作成します。

```
open_run -name netlist_1 synth_1
create_debug_core myCore ila
```

デバッグ コアのプロパティは、次の例のように `set_property` コマンドを使用するとカスタマイズできます。

```
set_property C_DATA_DEPTH 2048 [get_debug_cores myCore]
```

次の例では、`MARK_DEBUG` プロパティを使用してネットのシーケンスをデバッグするようマークし、新しいデバッグ コアを作成して CLK ポートを適切なクロック ドメインに接続し、デバッグ ネットをコア上の PROBE ポートに割り当てます。

```
set_property MARK_DEBUG true [get_nets [list {control_reg[0]}
{control_reg[1]} \
{control_reg[2]} {control_reg[3]} {control_reg[4]} {control_reg[5]} \
{control_reg[6]} {control_reg[7]} {control_reg[8]} {control_reg[9]} \
{control_reg[10]} {control_reg[11]} {control_reg[12]} {control_reg[13]}
\
{control_reg[14]} {control_reg[15]} {control_reg[16]} {control_reg[17]}
\
{control_reg[18]} {control_reg[19]} {control_reg[20]} {control_reg[21]}
\
{control_reg[22]} {control_reg[23]} {control_reg[24]} {control_reg[25]}
\
{control_reg[26]} {control_reg[27]} {control_reg[28]} {control_reg[29]}
\
{control_reg[30]} {control_reg[31]}]]]
create_debug_core u_ila_0 ila
set_property port_width 1 [get_debug_ports u_ila_0/CLK]
connect_debug_port u_ila_0/CLK [get_nets [list wbClk ]]
set_property port_width 32 [get_debug_ports u_ila_0/PROBE0]
connect_debug_port u_ila_0/PROBE0 [get_nets [list {control_reg[0]}
{control_reg[1]} {control_reg[2]} {control_reg[3]} {control_reg[4]} \
{control_reg[5]} {control_reg[6]} {control_reg[7]} {control_reg[8]} \
{control_reg[9]} {control_reg[10]} {control_reg[11]} {control_reg[12]}
\

```

```
{control_reg[13]} {control_reg[14]} {control_reg[15]} {control_reg[16]}  
\  
{control_reg[17]} {control_reg[18]} {control_reg[19]} {control_reg[20]}  
\  
{control_reg[21]} {control_reg[22]} {control_reg[23]} {control_reg[24]}  
\  
{control_reg[25]} {control_reg[26]} {control_reg[27]} {control_reg[28]}  
\  
{control_reg[29]} {control_reg[30]} {control_reg[31]} ]]
```

関連項目

- [connect_debug_port](#)
- [create_debug_port](#)
- [delete_debug_core](#)
- [get_debug_cores](#)
- [implement_debug_core](#)
- [open_run](#)
- [report_debug_core](#)
- [report_property](#)
- [set_property](#)

create_debug_port

新規デバッグ ポートを作成します。

構文

```
create_debug_port [-quiet] [-verbose] <name> <type>
```

戻り値

新しいデバッグ ポート オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	デバッグ コア インスタンスの名前を指定します。
<code><type></code>	デバッグ ポートのタイプを指定します。

カテゴリ

[Debug \(デバッグ\)](#)、[XDC](#)

説明

`create_debug_core` コマンドを使用してデザインに追加された ILA デバッグ コアに追加する新しいポートを定義します。このポートは ILA コアへの接続ポイントで、デバッグ目的でデザインのネットに接続するために使用されます。

`create_debug_core` コマンドを使用して新しいデバッグ コアを作成すると、デフォルトで `clk` と `probe` ポートが含まれますが、トリガー入力/出力ポート タイプも追加できます。ポートのタイプと目的の詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

ポートに接続ポイントを複数設定し、デバッグ用に複数のネットをサポートできます。デフォルトでは、新しいポートの幅は 1 で定義されるので、接続できるネットは 1 つのみです。`probe` ポートの幅を変更して複数の信号をサポートできるようにするには、`set_property port_width` コマンドを使用します (例を参照)。

注記: `clk`、`trig_in`、`trig_in_ack`、`trig_out`、および `trig_out_ack` ポートの幅は 1 にしか設定できません。

信号をポートに接続するには `connect_debug_port` コマンド、既存のプロブ接続を変更するには `modify_debug_ports` コマンド、信号の接続を解除するには `disconnect_debug_port` コマンドを使用します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 新しいポートを接続する ILA デバッグ コアの名前を指定します。デバッグ コアは、`create_debug_core` で既に作成され、存在する必要があります。

<type> (必須): 挿入するデバッグ ポートのタイプを指定します。サポートされるポート タイプは、次のとおりです。

- `clk`: ILA デバッグ コアをクロック ドメインに接続するクロック ポートを定義します。各デバッグ コアには `clk` ポートを 1 つしか含めることができず、各 `clk` ポートは 1 つのクロック ドメインにしか接続できないので、異なるクロック ドメインからの信号をプローブするには複数の ILA コアを使用する必要があります。



重要: 接続されているクロックがフリーランニング クロックであることを確認してください。そうでないと、デザインがデバイスに読み込まれたときに、デバッグ コアと通信できなくなる可能性があります。

- `probe`: MARK_DEBUG プロパティでデバッグ用にマークされた信号に接続するためのプローブ ポイントを提供します。ILA デバッグ コアには複数の `probe` ポートを含めることができ、コアにポートを追加したときに Vivado ツールにより自動的に番号が付けられます。各 `probe` ポートには 1 つまたは複数のチャネル (接続ポイント) を含めることができ、`PORT_WIDTH` プロパティで定義します。
- `trig_in/trig_in_ack` および `trig_out/trig_out_ack`: ILA コアの `probe` 入力における特定の比較条件を検出するために使用される ILA プローブ トリガー コンパレータです。`trig_in` と `trig_in_ack`、`trig_out` と `trig_out_ack` は、ポート ペアとしてデバッグ コアに追加する必要があります。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

例

次の例では、新しいデバッグ コアを作成して `probe` ポートを追加し、ポート幅を 8 に設定して、その `probe` ポートに信号を接続しています。

```
create_debug_core myCore ila
create_debug_port myCore probe
set_property PORT_WIDTH 8 myCore/probe1
connect_debug_port -channel_start_index 1 myCore/probe1 \
{m1_cyc_i m1_ack_o m1_err_o m1_rty_o}
```

注記: ILA コアにはデフォルトで `clk` ポートおよび `probe` ポートが含まれるので、新しく追加したプローブ ポートには番号が付けられます (`probe1`)。

関連項目

- [connect_debug_port](#)
- [create_debug_core](#)
- [disconnect_debug_port](#)
- [set_property](#)

create_drc_check

ユーザー定義の DRC ルールを作成します。

構文

```
create_drc_check [-hiername <arg>] -name <arg> [-desc <arg>] [-msg <arg>]
               -rule_body <arg> [-severity <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-hiername]	ルールの階層名を指定します。DRC の GUI パネルで、ルールを配置するメニュー階層にこの名前が使用されます。メニュー階層を分離するには、ピリオド (.) を使用します。デフォルトは User Defined です。
-name	DRC ルールの名前を指定します。PREFIX-id という形式で指定します (PREFIX は 4 ～ 6 文字の略称、id は特定のルールを識別する整数)。類似したチェックには同じ略称を使用し、固有の id で識別するようにします。
[-desc]	DRC ルールの簡単な説明を指定します。これはオプションであり、デフォルトは <User rule - default description> です。
[-msg]	DRC ルールの詳細な説明を指定します。置換キーを含めることができます。使用可能な置換キーは、%MSG_STRING、%NETLIST_ELEMENT、%SITE_GROUP、%CLOCK_REGION、%BANK、%BEL_GROUP です。
-rule_body	ルールの本体を表す文字列を指定します。評価する Tcl プロシージャ名または Tcl コードの文字列を指定します。
[-severity]	DRC ルールの重要度を指定します。デフォルトは WARNING です。有効な値は、ERROR、"CRITICAL WARNING"、WARNING、ADVISORY です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

DRC、Object (オブジェクト)

説明

report_drc コマンドを実行したときに使用するユーザー定義の DRC ルール チェック (drc_check) を作成します。

ユーザー定義ルール チェックには固有の名前または略称を指定でき、オプションで複数のルールを階層にグループ化したり、ルールの説明や違反が発生したときの汎用プレースホルダー メッセージを指定したり、report_drc コマンドで実行するデザイン ルール チェックに関連付ける Tcl コードを参照できます。

このコマンドで指定された汎用プレースホルダー メッセージは、Tcl チェッカー プロシージャおよび create_drc_violation コマンドで検出されたデザイン オブジェクトと違反に関連する特定の情報に置き換えられます。

プロセスは次のとおりです。

- ユーザー定義ルールをチェックする際に適用するメソッドと、ルールでチェックするオブジェクトを定義する Tcl チェッカー プロシージャを記述します。Tcl チェッカー プロシージャは別の Tcl スクリプトで定義し、`report_drc` を実行する前に `source` コマンドで読み込む必要があります。
- Tcl チェッカーの `create_drc_violation` を使用して、デザインでルールをチェックした際に検出される違反を特定します。
- `create_drc_check` コマンドを使用して、`-rule_body` オプションで指定した Tcl チェッカー プロシージャを呼び出すユーザー定義 DRC ルール チェックを定義します。
- `create_drc_ruledeck` コマンドを使用してルール デックを作成し、`add_drc_checks` コマンドを使用してユーザー定義ルール チェックをルール デックに追加します。
- ルール デックまたはユーザー定義ルール チェックを指定して `report_drc` を実行し、違反がないかどうかをチェックします。

指定の名前の `drc_check` が既に存在する場合は、エラーが返されます。既存の `drc_check` を変更または上書きする場合は、`delete_drc_check` コマンドを使用してまずチェックを削除してください。

DRC ルール チェック オブジェクトには `is_enabled` プロパティがあり、`set_property` コマンドを使用して TRUE または FALSE に設定できます。新しいルール チェックを作成すると、`is_enabled` プロパティはデフォルトで TRUE に設定されます。`report_drc` コマンドを実行したときにルール チェックが使用されないようにするには、`is_enabled` プロパティを FALSE に設定します。これにより、新しい DRC チェックを作成し、`add_drc_checks` コマンドを使用してルール デックに追加した場合に、そのチェックをルール デックから削除せずにイネーブルにしたりディスエーブルにしたりできます。

各ユーザー定義 DRC ルール チェックには `USER_DEFINED` プロパティが設定されており、ユーザー定義ルール チェックを簡単に特定および選択できます。

引数

`-hiername <arg>` (オプション): 新しいルールのグループを指定します。デフォルトは User Defined です。このオプションで指定した値は、GUI での DRC ルールのリストにおける第一階層となります。新しく作成されたすべての DRC チェックは、すべての階層に追加され、`report_drc` コマンドによりデフォルトで使用されます。

`-name <arg>` (必須): デザイン ルールの名前を指定します。これは、`-rule_body` オプションで指定された Tcl チェッカー プロシージャの `create_drc_violation` コマンドで使用された名前と一致している必要があります。この名前が DRC レポートに関連の違反と共に表示されます。名前は、4～6 文字のルール グループの略称に、同じグループのルールを識別する ID を付けたものにする必要があります (ABCD-1、ABCD-23 など)。

`-desc <arg>` (オプション): ルールの簡単な説明を指定します。デフォルトは User Rule です。この説明は、GUI で DRC ルールをリストしたときに表示されます。DRC レポートおよびサマリでも使用されます。

`-msg <arg>` (オプション): ルールの違反が検出されたときに表示するメッセージを指定します。メッセージには、ルールの違反で検出されたデザイン エLEMENT で動的に置換するプレースホルダーを含めることができます。実際のデザイン データは、`report_drc` を実行したときにメッセージに挿入されます。各置換キーには、長い形と短縮形があります。有効な置換キーは次のとおりです。

- `%MSG_STRING (%STR)`: `create_drc_violation` コマンドで `-msg` オプションを使用して違反用に定義したメッセージ文字列。
- **注記:** `create_drc_check` コマンドで `-msg` オプションを指定しない場合、`%STR` がデフォルト メッセージとなり、`create_drc_violation` コマンドの `-rule_body` オプションで定義されたメッセージが DRC レポートに渡されます。
- `%NETLIST_ELEMENT (%ELG)`: セル、ピン、ポート、ネットなどのネットリスト エLEMENT。
- `%SITE_GROUP (%SIG)`: デバイス サイト。

- %CLOCK_REGION (%CRG): クロック領域。
- %BANK (%PBG): パッケージ I/O バンク。

-rule_body <arg> (必須): ルール チェック機能を定義する Tcl プロシージャの名前を指定します。Tcl プロシージャを -rule_body オプションに含めるか、別の Tcl スクリプトで定義して、ツールを起動したときまたは report_drc コマンドを実行する前に source コマンドで読み込みます。

Tcl チェッカー プロシージャでは、create_drc_violation コマンドを使用して、デザイン ルール違反に関連するデザイン エレメントを含む DRC 違反オブジェクトを作成できます。-msg オプションで定義されたメッセージの置換キーは、違反オブジェクトからのデザイン エレメントで置換されます。

-severity <arg> (オプション): 作成するルールの重要度を指定します。デフォルト値は WARNING です。有効な値は、次のとおりです。

- ERROR
- "CRITICAL WARNING"
- WARNING
- ADVISORY

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、RAMW-1 という名前のデザイン ルール チェックを定義しています。階層名および説明を定義し、デフォルトの重要度である警告を使用し、チェックを実行したときに dataWidthCheck プロシージャが呼び出されるようにします。

```
create_drc_check -name {RAMW-1} -hiername {RAMB} \
  -desc {Data Width Check} -rule_body dataWidthCheck -severity Advisory
```

次の Tcl スクリプトは dataWidthCheck プロシージャを定義しています。このプロシージャは、RAMW-1 チェックの -rule_body オプションで呼び出されます。この Tcl スクリプト ファイルは、report_drc コマンドを実行する前に source コマンドを使用して読み込む必要があります。

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B
# wider than 36.
proc dataWidthCheck {} {
  # list to hold violations
  set vios {}

  # iterate through the objects to be checked
  foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
    set bwidth [get_property WRITE_WIDTH_B $bram]
    if { $bwidth > 36 } {
```

```
# define the message to report when violations are found
set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
lappend vios $vio
}
}
if {[llength $vios] > 0} {
    return -code error $vios
} else {
    return {}
}
}

create_drc_check -name {RAMW-1} -hiername {RAMB Checks} \
    -desc {Data Width Check} -rule_body dataWidthCheck \
    -severity Advisory
```

注記: スクリプト ファイルには、Tcl チェッカー プロシージャと、report_drc コマンドで使用するためにそれを定義する create_drc_check コマンドの両方を含めることができます。この場合、Tcl スクリプト ファイルを読み込んだときに dataWidthCheck プロシージャと RAMW-1 デザイン ルール チェックの両方が読み込まれます。

関連項目

- [add_drc_checks](#)
- [create_drc_ruledeck](#)
- [create_drc_violation](#)
- [delete_drc_check](#)
- [get_drc_checks](#)
- [get_drc_violations](#)
- [report_drc](#)

create_drc_ruledeck

1 つまたは複数のユーザー定義 DRC ルール デック オブジェクトを作成します。

構文

```
create_drc_ruledeck [-quiet] [-verbose] <ruledecks>...
```

戻り値

drc_ruledeck オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><ruledecks></code>	作成する DRC ルール デックの名前を指定します。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

`report_drc` コマンドを実行したときに使用する 1 つまたは複数のユーザー定義ルール デックを作成します。

DRC ルール デック (drc_ruledeck) オブジェクトはデザイン ルール チェックのグループで、I/O プランニングや配置などの FPGA デザイン フローの異なる段階で実行されます。ツールには、定義済みのルール デックが含まれています。`get_drc_ruledecks` コマンドを使用すると、定義済みのルール デックを取得できます。

このコマンドで作成したルール デックは空で、チェックは含まれません。`add_drc_checks` コマンドを使用して、ルール デックにルール チェックを追加する必要があります。ルール デックからチェックを削除するには、`remove_drc_checks` コマンドを使用します。ルール デックに含めることができるデザイン ルール チェックのリストを取得するには、`get_drc_checks` コマンドを使用します。

このコマンドを実行すると、作成された drc_ruledeck のリストが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<ruledecks> (必須): 作成する 1 つまたは複数のユーザー定義 DRC ルール デックの名前を指定します。

例

次の例では、2 つの drc_ruledeck オブジェクトを作成しています。

```
create_drc_ruledeck my_rules project_rules
```

関連項目

- [add_drc_checks](#)
- [delete_drc_ruledeck](#)
- [get_drc_checks](#)
- [get_drc_ruledecks](#)
- [remove_drc_checks](#)
- [report_drc](#)

create_drc_violation

DRC 違反を作成します。

構文

```
create_drc_violation -name <arg> [-severity <arg>] [-msg <arg>] [-quiet]
                        [-verbose] [<objects>...]
```

使用法

名前	説明
-name	DRC ルールの名前を指定します。通常、4 ～ 6 文字で指定します。
[-severity]	DRC ルールの重要度を指定します。デフォルトは WARNING です。有効な値は、FATAL、ERROR、CRITICAL_WARNING、WARNING、ADVISORY です。
[-msg]	DRC ルールのメッセージ文字列を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	クエリするセル、ポート、ピン、ネット、クロック領域、サイト、パッケージ バンクを指定します。

カテゴリ

[DRC、Report \(レポート\)](#)

説明

report_drc コマンドでのレポート用に DRC 違反オブジェクトを作成し、その違反に関連するデザイン オブジェクトのリストを制御します。

create_drc_violation コマンドは、create_drc_check コマンドで作成されたユーザー定義デザイン ルールチェックのチェック機能を定義および実行する Tcl チェッカー プロシージャの一部として指定されます。違反オブジェクトは、そのデザイン ルール違反が検出されるたびに Tcl チェッカー プロシージャにより作成されます。

プロセスは次のとおりです。

- ユーザー定義ルールをチェックする際に適用するメソッドと、ルールでチェックするオブジェクトを定義する Tcl チェッカー プロシージャを記述します。Tcl チェッカー プロシージャは別の Tcl スクリプトで定義し、report_drc を実行する前に source コマンドで読み込む必要があります。
- Tcl チェッカーの create_drc_violation を使用して、デザインでルールをチェックした際に検出される違反を特定します。
- create_drc_check コマンドを使用して、-rule_body オプションで指定した Tcl チェッカー プロシージャを呼び出すユーザー定義 DRC ルール チェックを定義します。
- create_drc_ruledeck コマンドを使用してルール デックを作成し、add_drc_checks コマンドを使用してユーザー定義ルール チェックをルール デックに追加します。

- ルール デックまたはユーザー定義ルール チェックを指定して `report_drc` を実行し、違反がないかどうかをチェックします。

違反は `report_drc` コマンドでレポートされ、違反オブジェクトは `get_drc_violations` コマンドで返されます。DRC 違反オブジェクトに関連付けられているデザイン オブジェクトは、該当する `get_*` コマンド (`get_cells`、`get_nets`、`get_ports` など) で `-of_objects` オプションを使用して取得できます。

```
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

引数

`-name <arg>` (必須): 違反に関連するデザイン ルール チェックの名前を指定します。これは、`-rule_body` で関連の Tcl チェッカー プロシージャを呼び出す `create_drc_check` コマンドで使用された名前と一致している必要があります。 `create_drc_violation` コマンドからのメッセージは、同じ `-name` の `drc_check` に渡されます。

`-severity <arg>` (オプション): 作成する違反の重要度を指定します。個々の DRC 違反に設定した重要度が、ルール チェックのデフォルトの重要度の代わりに使用されます。ユーザー定義 DRC のデフォルトの重要度は、`create_drc_check` コマンドの `-severity` で指定した重要度になります。有効な値は、次のとおりです。

- ERROR
- "CRITICAL WARNING"
- WARNING
- ADVISORY

注記: 指定した重要度は、DRC 違反オブジェクトに関連付けられた DRC ルールの `SEVERITY` プロパティとして保存されます。

`-msg <arg>` (オプション): `create_drc_check` で定義したプレースホルダー メッセージで使用されている汎用文字列変数 (`$STR`) を置換する違反特定のメッセージを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): Tcl チェッカー プロシージャで検出された違反に関連するセル、ポート、ピン、ネット、クロック領域、サイト、パッケージ I/O バンク オブジェクトで、DRC オブジェクトのプレースホルダー メッセージに同じ `-name` で置換されるものを指定します。デザイン オブジェクトは、メッセージの置換キーに次のように対応します。

- %ELG: セル、ポート、ピン、ネットなどのネットリスト エレメント。
- %CRG: クロック領域。
- %SIG: デバイス サイト。
- %PBG: パッケージ I/O バンク。

注記: `create_drc_violation` コマンドで渡される <objects> の順序とタイプが、`create_drc_check` の `-msg` の指定と一致している必要があります。

例

次の Tcl スクリプトは `dataWidthCheck` プロシージャを定義しています。このプロシージャは、RAMW-1 チェックの `-rule_body` オプションで呼び出されます。この Tcl スクリプト ファイルは、`report_drc` コマンドを実行する前に `source` コマンドを使用して読み込む必要があります。

この Tcl チェッカー プロシージャの機能は、次のとおりです。

- 違反を格納するために作成されるリスト変数 (\$vios)
- 違反が検出されるたびに違反オブジェクトが作成され、リスト変数に追加されます。
- \$msg 文字列のプレースホルダー キー %ELG は、違反に関連する \$bram セルに置換されます。
- 違反が検出されると (\$vios > 0)、`dataWidthCheck` プロシージャによりエラー コードが返され、`report_drc` コマンドにチェックの結果を知らせます。
- 違反のリストは戻りコードと共に渡され、違反は `report_drc` でレポートされます。

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B
# wider than 36.
proc dataWidthCheck {} {
    # list to hold violations
    set vios {}

    # iterate through the objects to be checked
    foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
        set bwidth [get_property WRITE_WIDTH_B $bram]
        if { $bwidth > 36 } {
            # define the message to report when violations are found
            set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
            set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
            lappend vios $vio
        }
    }
    if {[llength $vios] > 0} {
        return -code error $vios
    } else {
        return {}
    }
}

create_drc_check -name {RAMW-1} -hiername {RAMB Checks} \
    -desc {Data Width Check} -rule_body dataWidthCheck \
    -severity Advisory
```

注記: スクリプト ファイルには、Tcl チェッカー プロシージャと、`report_drc` コマンドで使用するためにそれを定義する `create_drc_check` コマンドの両方を含めることができます。この場合、Tcl スクリプト ファイルを読み込んだときに `dataWidthCheck` プロシージャと RAMW-1 デザイン ルール チェックの両方が読み込まれます。

関連項目

- [add_drc_checks](#)
- [create_drc_ruledeck](#)
- [create_drc_check](#)
- [get_cells](#)

- [get_drc_checks](#)
- [get_drc_violations](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)
- [get_sites](#)
- [report_drc](#)
- [set_property](#)

create_fileset

新規ファイルセットを作成します。

構文

```
create_fileset [-constrset] [-simset] [-blockset] [-clone_properties <arg>]
               -define_from <arg> [-quiet] [-verbose] <name>
```

戻り値

新しいファイルセット オブジェクト

使用法

名前	説明
<code>[-constrset]</code>	ファイルセットを制約ファイルセットとして作成します (デフォルト)。
<code>[-simset]</code>	ファイルセットをシミュレーション ソース ファイルセットとして作成します。
<code>[-blockset]</code>	ファイルセットをブロック ソース ファイルセットとして作成します。
<code>[-clone_properties]</code>	指定したファイルセットからプロパティをコピーします。
<code>-define_from</code>	ブロックセットの最上位となるソース ファイルセットのモジュール名を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	作成するファイルセットの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

デザイン プロジェクト内に新しいファイルセットを定義します。ファイルは、`add_files` コマンドを使用して新しく作成したファイルセットに追加できます。

ファイルセットは、プロジェクト内で特定の機能を持つ複数のファイルです。1 つ以上の制約ファイルは制約ファイルセット (`-constrset`)、1 つ以上のシミュレーション テストベンチはシミュレーション ファイルセット (`-simset`) で指定できます。`create_fileset` コマンドを使用する際に指定できるファイルセット オプションは 1 つのみです。タイプを指定しない場合、デフォルトで制約ファイルセットが作成されます。

`create_fileset -blockset` コマンドを使用して、IP コアまたはデザインの階層モジュールを独立階層 (OOC) ブロックとして設定することも可能です。ブロック ファイルセット (ブロックセット) は、`-define_from` オプションで指定された IP またはモジュールの階層ファイル コレクションを作成します。指定の階層モジュールに関連するファイルは、現在のファイルセットから新しいブロックセットに移動されます。ブロックセットを作成すると、そのブロック用に OOC 合成およびインプリメンテーション `run` も定義されます。合成済みデザイン チェックポイント (DCP) および必要な構造シミュレーション ネットリストを含む OOC モジュールの出力ファイルは、ブロックセット内に保存されます。構造シミュレーション ネットリストは、ブロックのビヘイビア モデルがない場合、またはターゲット シミュレータでサポートされる言語のビヘイビア モデルがない場合に必要です。必要に応じて IP またはモジュールの OOC 制約ファイルを定義でき、ブロック ファイルセットに追加できます。



ヒント: OOC デザインの詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896) または『Vivado Design Suite ユーザー ガイド: 階層デザイン』(UG905) を参照してください。

`create_fileset` コマンドを実行すると、新しく作成されたファイルセットの名前が返されるか、正常に実行されなかった場合はエラー メッセージが返されます。

引数

`-constrset` (オプション): 1 つ以上の制約ファイルを含める制約ファイルセットを作成します。`-constrset`、`-simset`、または `-blockset` オプションのいずれも指定しない場合、デフォルトで制約ファイルセットが作成されます。

`-simset` (オプション): 1 つ以上のシミュレーション ソース ファイルを含めるシミュレーション ファイルセットを作成します。ファイルセットのタイプには、`-constrset` または `-simset` オプションのいずれか 1 つしか設定できません。両方とも指定した場合は、エラー メッセージが表示されます。

`-blockset` (オプション): OOC デザインの IP コアまたは階層モジュールを設定するブロック ファイルセットを作成します。



重要: `-blockset` オプションを使用する場合、`-define_from` オプションを使用してブロックセットの最上位として使用する IP またはモジュールを指定する必要があります。

`-clone_properties <arg>` (オプション): 指定のファイルセットのプロパティを、新しく作成するファイルセットに追加します。このオプションを使用すると、新しいファイルセットが `USED_IN` などの必要なプロパティを使用して作成されます。

`-define_from <arg>` (オプション): ブロック ファイルセットに定義する IP コアの最上位モジュールを指定します。



重要: このオプションは、`-blockset` オプションを使用する場合は必須です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成するファイルセットの名前を指定します。

例

次の例では、constraints2 という名前の新しい制約ファイルセットを作成しています。

```
create_fileset -constrset -quiet constraints2
```

注記: -quiet オプションが指定されているので、指定したファイルセットの作成中にエラーが発生しても、エラーメッセージは表示されません。

次の例では、-define_from オプションで指定された階層モジュールの OOC ブロックセットを作成しています。

```
create_fileset -blockset -define_from dac_spi dac_spi
```

次の例では、sim_1 という名前の新しいシミュレーション ファイルセットを作成しています。

```
create_fileset -simset sim_1
```

関連項目

- [add_files](#)
- [current_fileset](#)
- [synth_ip](#)

create_generated_clock

既存のクロックからクロック オブジェクトを生成します。

構文

```
create_generated_clock [-name <arg>] [-source <args>] [-edges <args>]  
    [-divide_by <arg>] [-multiply_by <arg>] [-combinational]  
    [-duty_cycle <arg>] [-invert] [-edge_shift <args>] [-add]  
    [-master_clock <args>] [-quiet] [-verbose] <objects>
```

戻り値

新しいクロック オブジェクト

使用法

名前	説明
[-name]	生成クロックの名前を指定します。
[-source]	マスター クロック ソース オブジェクト ピン/ポートを指定します。
[-edges]	エッジを指定します。
[-divide_by]	分周係数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
[-multiply_by]	通係数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
[-combinational]	組み合わせロジックを介して 1 で分周されるクロックを作成します。
[-duty_cycle]	クロックのデューティ サイクルを指定します。有効な値は 0.0 ~ 100.0 で、デフォルト値は 50.0 です。
[-invert]	信号を反転します。
[-edge_shift]	エッジ シフトを指定します。
[-add]	ソース オブジェクトの既存クロックに追加します。
[-master_clock]	マスター ピンに複数のクロックがある場合に、使用するクロックを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	クロック ソース ポート、ピン、またはネットを指定します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

デザインに含まれる既存の物理クロック オブジェクトから新しいクロックを生成します。

クロックは、次の 3 つの方法のいずれかを使用してデザインに追加できます。

- `create_clock` コマンドでプライマリ クロックまたは仮想クロックを定義します。
- `create_generated_clock` コマンドを使用してプライマリ物理クロックから派生クロックを定義します。
- クロックが MMCM、PLL、BUFR を介して伝搬されると、Vivado Design Suite により自動的に派生クロックが生成されます。

`create_generated_clock` コマンドを使用して、Vivado Design Suite により MMCM、PLL、BUFR から自動的に派生されたクロックの名前を変更できます。この場合、新しいクロックは作成されず、指定のソース オブジェクトで定義されている既存のクロックの名前が指定の名前に変更されます。これには、`-name` および `<object>` を指定する必要があります。ソース オブジェクトに複数のクロックが存在する場合に `-source` および `-master_clock` オプションを使用して名前を変更するクロックをさらに特定できます。自動生成されたクロックの名前の変更については、『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903) を参照してください。



重要: 名前を変更するコマンドを実行する際に、ほかの制約で既に使用されているクロックの名前は変更できません。そのクロックが XDC ファイルで使用される前に、クロック名を変更する必要があります。

このコマンドを実行すると、作成されたクロック オブジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-name <arg>` (オプション): 指定のオブジェクトに作成する生成クロックの名前を指定するか、指定のオブジェクト上の既存のクロックに割り当てられている名前を変更します。名前を指定しない場合、生成クロックの名前はそれが割り当てられるオブジェクトの名前になります。複数のオブジェクトに割り当てる場合は、最初のオブジェクトの名前になります。

`-source <arg>` (オプション): 新しいクロックのマスターとなるクロックのピンまたはポートを指定します。マスタークロックには、仮想クロックではなく定義済みの物理クロックを使用する必要がありますが、プライマリクロックまたは別の生成クロックを使用できます。ソース ピンまたはポートに複数のクロックが定義されている場合は、`-master_clock` オプションを使用して生成クロックを定義するのに使用するソース上のクロックを指定する必要があります。

`-edges <arg>` (オプション): 生成クロックの遷移を定義するマスタークロックのエッジを指定します。生成クロックの遷移ポイント (1、2、3) を、マスタークロックのエッジ カウントで指定します。生成クロックの遷移のシーケンスは、クロックの周期とデューティ サイクルを定義します。位置 1 が生成クロックの最初の立ち上がりエッジを定義し、位置 2 は生成クロックの最初の立ち下がりエッジとなるのでデューティ サイクルを定義し、位置 3 は生成クロックの 2 番目の立ち上がりエッジとなるのでクロック周期を定義します。複数のエッジ番号は波かっこ {} で囲みます。エッジ番号の指定方法は、例を参照してください。

`-divide_by <arg>` (オプション): マスタークロックの周波数を分周する値を指定し、生成クロック オブジェクトの周波数を設定します。1 以上の整数を指定する必要があります。

`-multiply_by <arg>` (オプション): マスタークロックの周波数を逡倍する値を指定し、生成クロック オブジェクトの周波数を設定します。1 以上の整数を指定する必要があります。デフォルトは 1 です。

`-combinational` (オプション): 生成クロックのソース ピンとマスター クロックのソース ピンの間の組み合わせパスのみの遅延をトレースして、生成クロックのレイテンシを算出します。デフォルトでは、シーケンシャル パスと組み合わせパスの両方がトレースされて、生成クロックのレイテンシが算出されます。

`-duty_cycle <arg>` (オプション): `-multiply_by` オプションと共に使用し、生成クロックのデューティ サイクルを新しいクロック周期のパーセントで指定します。有効な値は 0.0 ~ 100 です。デフォルト値は 50.0 です。

`-invert` (オプション): マスター クロックの位相を反転した生成クロックを作成します。

`-edge_shift <arg>` (オプション): 生成クロックのエッジを、マスター クロックを基準に指定の値だけシフトします。エッジ シフトの指定方法は、例を参照してください。

`-add` (オプション): 異なるクロック波形で同時解析を実行する場合に、同じソースに複数のクロックを定義します。追加するクロックを `-name` を使用して指定する必要があります。このオプションを使用しないと、`create_clock` コマンドで自動的に名前が割り当てられ、同じ名前のクロックが存在する場合は上書きされます。

注記: `-master_clock` および `-name` オプションは、`-add` オプションと共に指定する必要があります。

`-master_clock <arg>` (オプション): ソース ピンまたはポートに複数のクロックがある場合に、新しいクロック オブジェクトの生成に使用するマスター クロックを指定します。

注記: `-add` および `-name` オプションは、`-master_clock` オプションと共に指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 生成クロックを割り当てるピンまたはポート オブジェクトを指定します。指定したオブジェクトに既にクロックが定義されている場合は、`-add` オプションを使用して新しい生成クロックが追加されるようにし、オブジェクト上の既存のクロックが削除されないようにします。

例

次の例では、指定のクロック ピン上のマスター クロックを分周してクロックを生成しています。`-name` が指定されていないので、生成クロックの名前は割り当てられるピンと同じになります。

```
create_generated_clock -divide_by 2 -source \
    [get_pins clkgen/cpuClk] [get_nets fftEngine/CLK]
```

次の例では、指定のソース クロックから CLK1 という生成クロックを定義し、生成クロックの遷移ポイントとして使用するマスター クロックのエッジを指定し、エッジを指定の量シフトしています。`-edges` オプションでソース クロックの 2 番目のエッジを生成クロックの最初のクロック エッジ、ソース クロックの 3 番目のエッジを生成クロックの最初の立ち下がりエッジ、ソース クロックの 8 番目のエッジを生成クロックの 2 番目の立ち上がりエッジに指定しています。これらの値により、生成クロックの周期はソース クロックのエッジ 2 からエッジ 8 までの時間に定義され、デューティ サイクルはソース クロックのエッジ 2 からエッジ 3 までの時間の周期に対するパーセントに定義されます。また、生成クロックの各エッジは指定の量シフトされます。

```
create_generated_clock -name CLK1 -source CMB/CLKIN -edges {2 3 8} \
    -edge_shift {0 -1.0 -2.0} CMB/CLKOUT
```

注記: 生成クロックの波形は、`-edges` オプションで定義された遷移に基づいて繰り返されます。

次の例では、MUX の出力から 2 つの生成クロックを作成します。クロックを割り当てるオブジェクトに複数のクロックが割り当てられているので、`-master_clock` を使用して使用するクロックを指定し、`-add` を使用してフリップフロップの Q ピンに生成クロックを割り当て、`-name` を使用して生成クロックの名前を定義しています。

```
create_generated_clock -source [get_pins muxOut] -master_clock M_CLKA \
-divide_by 2 -add -name gen_CLKA [get_pins flop-Q]
create_generated_clock -source [get_pins muxOut] -master_clock M_CLKB \
-divide_by 2 -add -name gen_CLKB [get_pins flop-Q]
```

次の例では、Vivado Design Suite により MMCM クロック出力に自動生成されたクロックの名前を変更しています。

```
create_generated_clock -name CLK_DIV2 [get_pins mmcm/CLKOUT1]
```

関連項目

- [check_timing](#)
- [create_clock](#)
- [get_generated_clocks](#)
- [get_pins](#)
- [set_clock_latency](#)
- [set_clock_uncertainty](#)
- [set_propagated_clock](#)

create_gui_custom_command

GUI のカスタム コマンドを作成します。

構文

```
create_gui_custom_command -name <arg> [-menu_name <arg>]
                             [-description <arg>] [-show_on_toolbar] [-run_proc <arg>]
                             [-toolbar_icon <arg>] [-command <arg>] [-tcl_file <arg>] [-quiet]
                             [-verbose]
```

使用法

名前	説明
-name	作成するコマンドの名前を指定します。
[-menu_name]	カスタム コマンドのメニュー名を指定します。
[-description]	カスタム メニュー コマンドおよびオプションでツールバー ボタンに表示する説明テキストを指定します。
[-show_on_toolbar]	作成するコマンドをツールバーに追加します。
[-run_proc]	-command および -tcl_file オプションを両方とも指定する場合に必要です。true の場合は -command オプションで指定されたコマンドを実行し、それ以外の場合は -tcl_file オプションで指定されたソース スクリプトを実行します。
[-toolbar_icon]	ツールバー ボタンに表示する PNG または JPEG ファイルへの完全パスを指定します。
[-command]	実行するコマンドを指定します。
[-tcl_file]	実行する Tcl ファイルへの完全パスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

ユーザー カスタム コマンドの GUI メニュー項目を作成します。メニュー コマンドは、Vivado IDE の [Tools]→[Custom Commands] メニューに表示されます。

カスタム GUI コマンドは、ツールバーに表示することもできます。GUI ボタンをクリックすると、カスタム コマンドが実行されます。GUI ボタンをクリックしたときに、Tcl ファイルが読み込まれるようにすることもできます。

`get_gui_custom_commands` を使用すると、ユーザー定義のカスタム コマンドのリストを取得できます。

引数

-name (必須): 作成するコマンドの名前を指定します。

-menu_name (オプション): カスタム コマンドのメニュー名を指定します。

`-description` (オプション): カスタム メニュー コマンドおよびオプションでツールバー ボタンに表示する説明テキストを指定します。

`-show_on_toolbar` (オプション): 作成するコマンドをツールバーに追加します。

`-run_proc` (オプション): `-command` と `-tcl_file` オプションの両方を指定している場合に必要です。true の場合は `-command` オプションで指定されたコマンドを実行し、それ以外の場合は `-tcl_file` オプションで指定された Tcl スクリプトを実行します。

`-toolbar_icon` (オプション): ツールバー ボタンに表示する PNG または JPEG ファイルへの完全パスを指定します。

`-command` (オプション): 実行するコマンドを指定します。

`-tcl_file` (オプション): 実行する Tcl ファイルへの完全パスを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、`print_versionabc` という名前の GUI カスタム コマンドを作成し、ツールバーに追加します。

```
create_gui_custom_command -name print_version -command "version" \
-description "Gets tool version" -show_on_toolbar
```

関連項目

- [create_gui_custom_command_arg](#)
- [get_gui_custom_command_args](#)
- [get_gui_custom_commands](#)

create_gui_custom_command_arg

GUI のカスタム コマンドのカスタム コマンド引数を作成します。

構文

```
create_gui_custom_command_arg -command_name <arg> -arg_name <arg>
[-default <arg>] [-comment <arg>] [-optional] [-quiet] [-verbose]
```

使用法

名前	説明
-command_name	引数を作成するカスタム コマンドの名前を指定します。
-arg_name	作成するカスタム コマンド引数の名前を指定します。
[-default]	カスタム コマンド引数のデフォルト値を指定します。
[-comment]	カスタム コマンド引数のコメントを指定します。
[-optional]	カスタム コマンド引数をオプションにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

特定のカスタム GUI コマンドの引数を定義します。

`get_gui_custom_commands` を使用すると、定義済みのカスタム コマンドを取得できます。
`get_gui_custom_command_args` を使用すると、特定の GUI カスタム コマンドの GUI カスタム コマンド引数のリストを取得できます。

引数

-command_name (必須): 引数を作成するカスタム GUI コマンドの名前を指定します。

-arg_name (必須): 作成する引数の名前を指定します。

-default (オプション): 作成する引数のデフォルト値を指定します。

-comment (オプション): カスタム コマンド引数のコメントを指定します。

-optional (オプション): カスタム コマンド引数をオプションにします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

この例は、カスタム GUI コマンド `print_version` に `quiet` という引数を作成します。この引数はオプションとして定義され、デフォルト値は `-quiet` です。

```
create_gui_custom_command_arg -command_name print_version -arg_name quiet \  
-default "-quiet" -comment "Ignore commands errors" -optional
```

関連項目

- [create_gui_custom_command](#)
- [get_gui_custom_command_args](#)
- [get_gui_custom_commands](#)

create_hw_axi_txn

ハードウェア AXI トランザクション オブジェクトを生成します。

構文

```
create_hw_axi_txn [-address <arg>] [-data <arg>] [-size <arg>] -type <arg>
                  [-len <arg>] [-burst <arg>] [-cache <arg>] [-id <arg>] [-force]
                  [-quiet] [-verbose] <name> <hw_axi>
```

戻り値

新しいハードウェア AXI トランザクション オブジェクト。

使用法

名前	説明
[-address]	AXI の読み出しまたは書き込みアドレスを指定します。デフォルトはアドレス 0 です。
[-data]	トランザクション データを指定します。デフォルトはすべて 0 です。
[-size]	(廃止予定) データのワード サイズをビット数で指定します。これは、IP コアに基づいて自動的に設定されるようになっています。
-type	読み出し (READ) または書き込み (WRITE) を指定します。
[-len]	トランザクションの長さをデータ ワード数で指定します。デフォルトは 1 です。
[-burst]	バースト タイプを指定します。有効な値は INCR、FIXED、または WRAP で、デフォルトは INCR です。
[-cache]	AXI キャッシュ タイプを指定します。デフォルトは 3 です。
[-id]	アドレス ID を指定します。デフォルトは 0 です。
[-force]	指定した名前のトランザクションが存在する場合はそのトランザクションを上書きし、存在しない場合は新しいトランザクションを作成します。デフォルトは 0 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	新しいオブジェクトの名前を指定します。
<hw_axi>	ハードウェア AXI コア オブジェクトを関連付けます。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

get_hw_axis コマンドで指定した JTAG to AXI Master コア (hw_axi オブジェクト) の読み出しまたは書き込みトランザクションを定義します。

JTAG to AXI Master コアは、AXI トランザクションを駆動し、ハードウェア デバイス内の AXI 信号を駆動する AXI マスターとして機能するカスタマイズ可能な IP コアです。JTAG to AXI Master コアは、AXI4-Stream を除くメモリ マップ AXI インターフェイスをすべてサポートし、AXI4-Lite プロトコルをサポートします。JTAG to AXI Master コアの詳細は、『LogiCORE IP JTAG to AXI Master 製品ガイド』 (PG174) を参照してください。

AXI トランザクションは、JTAG to AXI Master コアからコアに接続されている AXI 信号への読み出し/書き込みバースト トランザクションです。AXI トランザクションでは、送信するデータや送信先アドレスなど、読み出しまたは書き込み トランザクションの特性を設定できます。これらの定義された トランザクションは、指定の `hw_axi` オブジェクトのプロパティとして保存され、`run_hw_axi` および `report_hw_axi_txn` コマンドを使用して実行およびレポートされます。

このコマンドを実行すると、作成されたハードウェア AXI トランザクション (`hw_axi_txn`) オブジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-address <arg>` (オプション): `hw_axi` オブジェクト上の読み出しまたは書き込みを実行するレジスタのアドレスを指定します。デフォルト アドレスは 0000 です。

`-data <arg>` (オプション): 書き込み トランザクションで `hw_axi` のアドレス ロケーションに書き込むデータ値を 16 進数フォーマットで指定します。デフォルトのデータ値はすべて 0 です。

`-type [READ | WRITE]` (必須): 指定のアドレスに対して読み出し (READ) を実行するか書き込み (WRITE) を実行するかを指定します。

`-len <arg>` (オプション): `-size` オプションに基づいて、読み出しまたは書き込み トランザクションの長さをデータワード数で指定します。デフォルトは 1 です。

`-burst <arg>` (オプション): 実行する AXI バーストのタイプを指定します。バーストは、INCR、FIXED、または WRAP に指定できます。デフォルト データ バーストはインクリメンタル (INCR) です。

`-cache <arg>` (オプション): インプリメントする AXI コマンド キャッシュを 10 進数で指定します。デフォルト値は 3 です。読み出し/書き込みのキャッシュ設定の詳細は、『JTAG to AXI Master LogiCORE IP 製品ガイド』 (PG174) を参照してください。

`-id <arg>` (オプション): AXI トランザクションに割り当てる ID を 10 進数で指定します。これにより、さまざまな読み出しおよび書き込み トランザクションへの応答をツールで特定できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 新しく作成する AXI トランザクション オブジェクトの名前を指定します。この名前を使用して、トランザクションにアクセスしたり、トランザクションを呼び出すことができます。

`<hw_axi>` (必須): トランザクションを定義する `hw_axi` オブジェクトを指定します。`hw_axi` は、`get_hw_axi` コマンドを使用してオブジェクトとして指定する必要があります。名前指定することはできません。

例

次の例では、32 ビット データ ワードの 2 データ ワード長トランザクションで、16 進数値で指定した 10 進数 10 のデータ値を書き込むトランザクションを作成しています。

```
create_hw_txn write_1 -type WRITE -data 0000_0000_0000_000A -size 32 -len 2 \
[get_hw_axis hw_axi_1]
```

次の例では、指定の 128 ビット データ ストリームを hw_axi オブジェクトの指定のアドレスに書き込む新しいハードウェア AXI トランザクション オブジェクト (hw_axi_txn) を作成しています。新しい AXI トランザクションの名前は write_txn です。

```
create_hw_axi_txn write_txn [get_hw_axis hw_axi_1] -type WRITE \
-len 4 -data {4444444444_33333333_22222222_11111111}
```

次の例では、AXI 読み出しおよび書き込みトランザクションを作成し、hw_axi を実行して、その結果をレポートしています。

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
-data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
-len 4 -type read

run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report

run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report

close_hw_target;
disconnect_hw_server;
```

次の例では、読み出しタイプの hw_axi トランザクションのシーケンスを作成し、実行しています。

```
# Read registers
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$MM2S_VDMACR_OFFSET]] -type read txn00 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$MM2S_VDMASR_OFFSET]] -type read txn01 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$MM2S_REG_INDEX_OFFSET]] -type read txn02 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$SPARK_PTR_REG_OFFSET]] -type read txn03 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$VERSION_OFFSET]] -type read txn04 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$S2MM_VDMACR_OFFSET]] -type read txn05 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$S2MM_VDMASR_OFFSET]] -type read txn06 [get_hw_axis hw_axi_1]
create_hw_axi_txn -address [format %08x [expr $baseaddr + \
$S2MM_VDMA_IRQ_OFFSET]] -type read txn07 [get_hw_axis hw_axi_1]
run_hw_axi -quiet [get_hw_axi_txns]
```

関連項目

- [delete_hw_axi_txn](#)
- [get_hw_axis](#)
- [get_hw_axi_txns](#)
- [refresh_hw_axi](#)
- [report_hw_axi_txn](#)
- [reset_hw_axi](#)
- [run_hw_axi](#)

create_hw_bitstream

ビットストリーム ファイルをメモリに読み込みます。

構文

```
create_hw_bitstream -hw_device <arg> [-mask <arg>] [-nky <arg>]
[-detect_partial] [-quiet] [-verbose] [<file>]
```

使用法

名前	説明
-hw_device	ターゲット hw_device 接続を指定します。
[-mask]	hw_device のマスク ファイルを指定します。
[-nky]	hw_device の暗号化ファイルを指定します。
[-detect_partial]	パーシャル ビットストリームを検出します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	ビットストリーム ファイル名を指定します。

カテゴリ

Hardware (ハードウェア)

説明

`write_bitstream` コマンドで作成したビットストリーム ファイルを読み込み、ハードウェア ビットストリーム (`hw_bitstream`) オブジェクトを作成して、そのオブジェクトを Vivado Design Suite のハードウェア マネージャー 機能のハードウェア デバイス (`hw_device`) オブジェクトに関連付けます。

ハードウェア ビットストリーム (`hw_bitstream`) オブジェクトは、デバイスの `PROGRAM.HW_BITSTREAM` プロパティを使用して指定のハードウェア デバイスに関連付けられます。このプロパティは、`create_hw_bitstream` コマンドにより自動的に設定されます。`PROGRAM.FILE` プロパティに指定のビットストリーム ファイルのファイル パスも設定されます。

注記: `program_hw_devices` コマンドを使用した場合にも、`hw_bitstream` オブジェクトも自動的に作成され、`hw_device` オブジェクトに関連付けられます。

`write_bitstream -mask` コマンドを使用してビットストリーム ファイルと共に記述されたマスク ファイルは、`MASK` プロパティにより `hw_bitstream` オブジェクトに関連付けられます。暗号化されたビットストリームに必要な暗号キー ファイルは、`ENCRYPTION.FILE` プロパティにより `hw_bitstream` オブジェクトに関連付けられます。これらのファイルは、`-mask` および `-nky` オプションを使用して `hw_bitstream` オブジェクトに関連付けられます。

作成した `hw_bitstream` オブジェクトを削除するには、`delete_hw_bitstream` コマンドを使用します。

このコマンドを実行すると、作成された `hw_bitstream` オブジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-hw_device <arg>` (必須): ハードウェア ビットストリーム オブジェクトに関連付けるハードウェア デバイス オブジェクトを指定します。hw_device は、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

`-mask <arg>` (オプション): 検証用にビットストリームのどのビットをリードバック データと比較するべきかを示すためにデバイスに使用するマスク ファイルを指定します。マスク ファイルは `write_bitstream -mask_file` コマンドを使用して記述し、`verify_hw_bitstream` コマンドでプログラムされたビットストリームが正しいことを検証する際に使用されます。指定したマスク ファイルは、作成された hw_bitstream オブジェクトの MASK プロパティに定義されます。

`-nky <arg>` (オプション): eFUSE レジスタまたはバッテリー バックアップ SRAM (BBR) にプログラムする暗号キー ファイルを指定します。暗号キーは `write_bitstream` コマンドにより NKY または NKZ ファイルに記述され、暗号化されたビットストリームを使用する際に必要です。指定した暗号キー ファイルは、作成された hw_bitstream オブジェクトの ENCRYPTION.FILE プロパティに定義されます。暗号キーの値はこのファイルから抽出され、ENCRYPTION.KEY プロパティに定義されます。

`-detect_partial` (オプション): `write_bitstream -reference_bitfile` コマンドで作成されたパーシャル ビットストリームを検出します。このオプションを使用すると、パーシャル ビットストリーム ファイルを使用してハードウェア デバイスのインクリメンタル プログラムが可能になります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 読み込むビットストリーム ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、新しいハードウェア ビットストリーム オブジェクトを作成し、現在のハードウェア デバイス オブジェクトに関連付けます。

```
create_hw_bitstream -hw_device [current_hw_device] C:/Data/design1.bit
```

次の例では、現在の hw_device の hw_bitstream オブジェクトを作成し、そのビットストリームに関連付けるマスク ファイルと暗号キー ファイル (NKY) を指定しています。

```
create_hw_bitstream -hw_device [current_hw_device] \
-mask ./project_cpu_encrypt.runs/impl_1/top.msk \
-nky ./project_cpu_encrypt.runs/impl_1/top.nky \
./project_cpu_encrypt.runs/impl_1/top.bit
```

関連項目

- [current_hw_device](#)

- [delete_hw_bitstream](#)
- [get_hw_devices](#)
- [program_hw_devices](#)
- [set_property](#)
- [write_bitstream](#)

create_hw_cfgmem

コンフィギュレーション メモリ (cfgmem) ファイルをメモリに読み込みます。

構文

```
create_hw_cfgmem -hw_device <arg> [-quiet] [-verbose] <mem_device>
```

使用法

名前	説明
-hw_device	hw_cfgmem オブジェクトに関連付ける hw_device オブジェクトを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<mem_device>	フラッシュ メモリの名前を get_cfgmem_parts コマンドを使用して指定します。

カテゴリ

Hardware (ハードウェア)

説明

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成し、指定のハードウェア デバイスに関連付けます。

ザイリンクス® FPGA にビットストリーム データを読み込む (プログラムする) プロセスは、コンフィギュレーションと呼ばれます。コンフィギュレーションは異なるアプリケーションのニーズに対応するため柔軟に設計し、できる限り既存のシステム リソースを利用してシステム コストを最小限に抑えます。

ザイリンクス FPGA をコンフィギュレーションするには、ハードウェア デバイス (hw_device) の内部メモリにデザイン特定のコンフィギュレーション データをビットストリーム ファイルの形で読み込みます。hw_cfgmem は、ザイリンクス FPGA デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。create_hw_cfgmem オブジェクトを作成し、ハードウェア デバイスに関連付けたら、program_hw_cfgmem コマンドを使用してコンフィギュレーション メモリをビットストリームおよびその他のデータでプログラムできます。

hw_cfgmem オブジェクトは、デバイス オブジェクトの PROGRAM.HW_CFGMEM プロパティを使用して指定のハードウェア デバイス オブジェクトに関連付けられます。hw_cfgmem オブジェクトを操作するには get_hw_cfgmems コマンドを使用し、hw_device からオブジェクトを取得するには get_property コマンドを使用します。

```
get_property PROGRAM.HW_CFGMEM [current_hw_device]
```



ヒント: 新しい hw_cfgmem オブジェクトを作成する際、下の例に示すように、そのオブジェクトを Tcl 変数に関連付けることもできます。この変数を参照することにより、オブジェクトのプロパティを設定したり、program_hw_cfgmem または readback_hw_cfgmem などの Tcl コマンドで使用できます。

このコマンドを実行すると、作成された hw_cfgmem オブジェクトが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-hw_device <arg>` (必須): メモリ コンフィギュレーション デバイスを関連付ける `hw_device` オブジェクトを指定します。`hw_device` は、`get_hw_devices` コマンドまたは `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<mem_device>` (必須): 関連付けられた `hw_device` のコンフィギュレーションに使用するフラッシュ メモリ デバイスを指定します。コンフィギュレーション メモリは、`get_cfgmem_parts` コマンドを使用して `cfgmem_part` オブジェクトとして指定する必要があります。

注記: コンフィギュレーション メモリ パーツ (`cfgmem_part`) オブジェクトの `COMPATIBLE_PARTS` プロパティは、コンフィギュレーション メモリがどのザイリンクス デバイスと互換性があるかを特定します。

例

次に例を示します。

1. 現在のハードウェア デバイス (`current_hw_device`) に関連付けられている PART を取得します。
2. デバイス パーツと互換性のある `cfgmem_parts` オブジェクトのリストを取得します。
3. 互換性のあるパーツの 1 つを使用して、新しい `hw_cfgmem` オブジェクトを作成します。
4. `write_cfgmem` コマンドで作成された `cfgmem` ファイルに現在の `hw_cfgmem` オブジェクトの `PROGRAM.FILE` プロパティを設定します。
5. 現在の `hw_cfgmem` オブジェクトをプログラムします。

```
set devPart [get_property PART [current_hw_device]]
set cfgParts [get_cfgmem_parts -of [get_parts $devPart]]
create_hw_cfgmem -hw_device [current_hw_device] [lindex $cfgParts 0]
set_property PROGRAM.FILE {C:/Data/cfgmem_file.mcs} [current_hw_cfgmem]
program_hw_cfgmem [current_hw_cfgmem]
```

関連項目

- [current_hw_cfgmem](#)
- [current_hw_device](#)
- [delete_hw_cfgmem](#)
- [get_cfgmem_parts](#)
- [get_hw_cfgmems](#)
- [get_property](#)
- [program_hw_cfgmem](#)
- [readback_hw_cfgmem](#)
- [write_cfgmem](#)

create_hw_device

開いているターゲットに hw_device (JTAG チェーン) を作成します。

構文

```
create_hw_device [-idcode <arg>] [-irlength <arg>] [-mask <arg>]
                 [-part <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-idcode]	16 進数のデバイス ID コードを指定します。
[-irlength]	デバイスの IR 幅を 10 進数で指定します。
[-mask]	16 進数のデバイス マスク値を指定します。
[-part]	作成するデバイスのパーツ タイプを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado ハードウェア マネージャーでは、SVF (Serial Vector Format) ファイルを使用したハードウェア デバイスのプログラムがサポートされています。SVF ファイルは、プログラム命令とコンフィギュレーション データを含む ASCII ファイルです。これらのファイルは、ATE マシンおよびエンベデッド コントローラーでバウンダリスキャン操作を実行するために使用されます。SVF ファイルには、ビットストリームをザイリンクス デバイスに直接プログラム、またはフラッシュ メモリ デバイスに間接プログラムするのに必要な JTAG コマンドが含まれます。SVF ファイルは、`write_hw_svf` コマンドを使用して記述し、`execute_hw_svf` コマンドを使用して開いているハードウェア ターゲット (hw_target) に適用します。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

hw_svf ファイルを作成するプロセスは、次のとおりです。

1. `create_hw_target` コマンドを使用して SVF ターゲットを作成します。
2. SVF ターゲットを開きます。
3. `create_hw_device` コマンドを使用して、SVF ターゲットに 1 つまたは複数のデバイスを作成します。
4. `program_hw_devices` などのコマンドを使用してデバイスをプログラムします。
5. `write_hw_svf` コマンドを使用して操作コマンドの SVF ファイルを記述します。

`create_hw_device` コマンドは、開いている SVF ターゲットに hw_device オブジェクトを作成し、JTAG チェーンに追加します。このデバイスは、ほかの hw_target と同様に、`get_hw_devices` や `program_hw_devices` などのコマンドを使用してクエリおよびプログラムできます。

開いている SVF hw_target の JTAG チェーンに追加するザイリンクス デバイスおよびユーザー定義のパーツの両方を作成できます。ザイリンクス デバイスの場合は、認識される製品番号を指定すると、Vivado ツールで適切な詳細が定義されます。ユーザー定義のパーツの場合は、適切なオプションを使用して JTAG ID コード、IR 幅、およびマスクの詳細を指定する必要があります。ユーザー定義のパーツは、JTAG チェーンおよび SVF hw_target 上にプレースホルダーとして追加されます。ユーザー パーツを `get_hw_devices` コマンドを使用して取得したり、`report_property` コマンドを使用してそのプロパティをクエリしたりすることはできますが、プログラムすることはできません。



重要: JTAG チェーンに対して操作を実行する前に、SVF ターゲットの JTAG チェーンを定義するすべてのデバイスを作成する必要があります。`create_hw_device` コマンドと `program` コマンドを混合すると、SVF ファイルで参照される JTAG チェーンが正しく定義されず、`execute_hw_svf` を実行したときに機能しません。

SVF ターゲット上に hw_device を作成した後、ビットストリーム (.bit) の割り当ておよびデバイスのプログラムなどのデバイスにサポートされる操作を実行できます。

```
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

`create_hw_device` コマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-idcode <arg>` (オプション): ユーザー定義デバイスの JTAG ID を指定します。ザイリンクス デバイスを指定する場合は、このオプションは必要ありません。

`-irlength <arg>` (オプション): ユーザー定義デバイスの JTAG 命令レジスタ (IR) 幅を指定します。ザイリンクス デバイスを指定する場合は、このオプションは必要ありません。

`-mask arg` (オプション): JTAG チェーンのデバイスからデータ レジスタ (DR) を読み出すために使用するマスクを指定します。このマスクは、デバイス応答のどのビットが有効であるかを定義します。マスクされたビットは無視されます。ザイリンクス デバイスを指定する場合は、このオプションは必要ありません。

`-part <arg>`: デバイスとしてザイリンクス パーツを指定するか、ユーザー定義のパーツ名を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、SVF ターゲットを作成し、そのターゲットを開いて、ターゲット上に新しい hw_device を作成しています。

```
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
```

次の例では、SVF ターゲット上に複数のデバイスを作成する際の正しい順序を示します。SVF ターゲットが作成されて開かれ、現在のターゲットにサイリンクス デバイス、ユーザー パーツ、および 2 つ目のサイリンクス デバイスが作成されています。2 つのサイリンクス デバイスに対してビットストリーム プロパティが定義され、デバイスがプログラムされて、SVF ファイルが記述されます。

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
create_hw_device -idcode 01234567 -irlength 8 -mask ffffffff -part
userPart1
create_hw_device -part xc7k9p
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [lindex [get_hw_devices]
0]
set_property PROGRAM.FILE {C:/Data/ku_design.bit} [lindex [get_hw_devices]
2]
program_hw_devices [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 2]
write_hw_svf C:/Data/myDesign.svf
```

関連項目

- [create_hw_target](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)
- [program_hw_devices](#)
- [set_property](#)

create_hw_probe

ハードウェア プロブ オブジェクトを作成します。

構文

```
create_hw_probe [-no_gui_update] [-map <arg>] [-quiet] [-verbose] <name>
               <core>
```

戻り値

新しいハードウェア プロブ オブジェクト。

使用法

名前	説明
<code>[-no_gui_update]</code>	GUI をアップデートしません。
<code>[-map]</code>	ビットを定義します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	新しいオブジェクトの名前を指定します。バス プロブには範囲が付けられます。
<code><core></code>	関連のハードウェア ILA コア オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado ロジック解析でトリガーを定義し、データを表示するため、指定の ILA コアに新しいユーザー定義のプロブを作成します。新しいプロブでは、既存プロブの特定のビット値をまとめて、波形ビューアで表示されるデータを単純または明確にできます。ユーザー定義プロブからのデータ サンプルは、`list_hw_samples` コマンドを使用してレポートできます。

ユーザー定義のプロブでは、ILA コアからの単一の物理的プロブからビット値をマップしたり、複数の物理的プロブからのビット値を単一のユーザー定義のプロブにまとめたりできます。単一のプロブからのビットをマップするプロブは、トリガーを作成したり、データを表示するために使用できます。複数の物理プロブからのビットを組み合わせたプロブは、Vivado ロジック解析でデータを表示するためにのみ使用可能です。

ユーザー定義プロブを削除するには、`delete_hw_probe` コマンドを使用します。

`create_hw_probe` コマンドを実行すると、正常に実行された場合はユーザー定義プロブの名前が返され、正常に実行されなかった場合はエラーが返されます。

引数

`-no_gui_update` (オプション): ユーザー定義プロブの追加を Vivado ロジック解析の GUI に反映しません。

`-map <arg>` (オプション): 物理プローブ ポートの名前と、新しいユーザー定義プローブにマップする信号ビットを指定します。物理プローブ ポートは指定のハードウェア ILA (`hw_ila`) オブジェクト上のポートであり、ILA コアでプローブされる信号に関連しています。`-map` の引数は、次のように物理プローブ ポート名とビットのリストとして指定します。

```
-map {0011 probe3[19] probe3[6]}
```



ヒント: このマップには、上記の例のように定数を含めることもできます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: ユーザー定義プローブ名を名前とプローブ幅の組み合わせ (例: `probeName[0:3]`) で指定します。幅を指定しない場合、プローブ幅は 1 ビットになります。

注記: ユーザー定義プローブの幅は、`-map` オプションで指定したビット数に一致している必要があります。

`<core>`: ILA コア オブジェクトを指定します。`hw_ila` は、名前で指定するか、または `get_hw_ilas` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、複数の物理プローブからのビットを、ILA コア上の新しいユーザー定義プローブにマップしています。

```
create_hw_probe -map {0011 probe5[3:0] probe8 probe9} myProbeAR[9:0]
hw_ila_1
```



ヒント: `-map` オプションで新しいプローブに 10 ビットがマップされているので、プローブ名ではそれに一致するポート幅が指定されています。

次の例では、30 ビットの信号の最上位ビットをコピーして符号拡張することにより `hw_probe` を作成し、ほかの 32 ビット符号付き信号に揃えています。

```
create_hw_probe -map {probe0[29] probe0[29] probe0[29:0]} \
mySignExtendedProbe[31:0] [get_hw_ilas hw_ila_1]
```

関連項目

- [get_hw_ilas](#)
- [get_hw_probes](#)

create_hw_sio_link

ハードウェア RX と TX エンドポイントの間に新しいリンクを作成します。ハードウェア TX または RX エンドポイントを少なくとも 1 つ指定する必要があります。いずれかがない場合、エンドポイントは不明として処理されます。不明のエンドポイントは、リンク プロパティで名前を変更できます。

構文

```
create_hw_sio_link [-description <arg>] [-quiet] [-verbose] [<hw_sio_rx>]  
                  [<hw_sio_tx>]
```

戻り値

新しいハードウェア SIO リンク

使用法

名前	説明
[-description]	リンクの説明を指定します。デフォルトはリンク オブジェクト名です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_sio_rx>]	RX エンドポイントを指定します。デフォルトはありません。
[<hw_sio_tx>]	TX エンドポイントを指定します。デフォルトはありません。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

現在のハードウェア デバイス上にインプリメントされている IBERT デバッグ コアの GT 上のトランスミッター (TX) とレシーバー (RX) オブジェクト間の通信リンクを定義します。

Vivado シリアル I/O 解析はリンク ベースの解析機能であり、IBERT デザイン内の任意のトランスミッターとレシーバーをリンクできます。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。リンクを設定するには、`set_property` コマンドを使用してリンク オブジェクトのプロパティ値を指定します。リンクの設定に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

このコマンドを実行すると、作成されたハードウェア SIO リンク (hw_sio_link) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-description <arg>` (オプション): リンクのラベルとして使用する簡単な説明を指定します。この説明は、hw_sio_link オブジェクトの DESCRIPTION プロパティに保存されます。



ヒント: `hw_sio_link` オブジェクトの NAME プロパティは IBERT デバッグ コア上のトランスミッター (TX) GT から レシーバー (RX) GT までのリンクの完全パス デスティネーションで、ハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`) オブジェクトが含まれます。-description オプションで指定した説明は、リンクを探しやすくするためのショートカットとして使用できます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_sio_rx> (オプション): 通信リンクに含める GT のレシーバー エlementを指定します。レシーバーは、`get_hw_rxs` コマンドを使用してオブジェクトとして指定する必要があります。

<hw_sio_tx> (オプション): 通信リンクに含める GT のトランスミッター エlementを指定します。トランスミッターは、`get_hw_txs` コマンドを使用してオブジェクトとして指定する必要があります。



重要: レシーバーとトランスミッターの指定はオプションですが、リンクを作成するためには少なくともいずれか 1 つを指定する必要があります。トランスミッターまたはレシーバーのみのリンクは、オープンエンド リンクと認識されます。

例

次の例では、指定の RX と TX 間に通信リンクを作成しています。

```
create_hw_sio_link -description Link_12 [get_hw_sio_txs *MGT_X0Y12*] \
[get_hw_sio_rxs *MGT_X0Y12*]
```



ヒント: 上記の例では RX の前に TX を指定していますが、順序は重要ではありません。

関連項目

- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_linkgroups](#)

create_hw_sio_linkgroup

新しいハードウェア SIO リンク グループを作成します。

構文

```
create_hw_sio_linkgroup [-description <arg>] [-quiet] [-verbose]  
                        <hw_sio_links>
```

戻り値

新しいハードウェア SIO リンク グループ

使用法

名前	説明
[-description]	リンク グループの説明を指定します。デフォルトはリンク グループ オブジェクト名です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_sio_links>	ハードウェア SIO リンク

カテゴリ

Hardware (ハードウェア)

説明

現在のデバイス上にインプリメントされている IBERT デバッグ コアの TX と RX 間の指定の通信リンクに関連付ける新しいグループを作成します。

Vivado シリアル I/O 解析機能はリンク ベースです。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。リンク グループ (hw_sio_linkgroup) オブジェクトを使用すると、複数のリンクにまとめてプロパティを設定したりスキャンを実行するため、リンクをグループとして関連付けることができます。

このコマンドを実行すると、作成されたリンク グループの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-description <arg> (オプション): リンク グループのラベルとして使用する簡単な説明を指定します。この説明は、hw_sio_linkgroup オブジェクトの DESCRIPTION プロパティに保存されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_links>` (必須): グループとして関連付けるハードウェア SIO リンク (`hw_sio_link`) オブジェクトを 1 つ以上指定します。`hw_sio_links` は、`create_hw_sio_link` または `get_hw_sio_links` コマンドを使用してオブジェクトとして指定する必要があります。

注記: 別のリンク グループに既に含まれている `hw_sio_link` オブジェクトを指定すると、エラーが返されます。

例

次の例では、`hw_sio_link` オブジェクトの `DESCRIPTION` プロパティを使用して新しいリンク グループに追加するリンクを特定しています。

```
create_hw_sio_linkgroup -description "LoopBack Links" \
    [get_hw_sio_links -filter {DESCRIPTION == Link_12 || DESCRIPTION == \
        "Link 9" || DESCRIPTION == "Link 10" || DESCRIPTION == "Link 11" }]
```

関連項目

- [create_hw_sio_link](#)
- [current_hw_device](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_linkgroups](#)

create_hw_sio_scan

新しいハードウェア SIO スキャンを作成します。リンク オブジェクトを指定する場合は、そのリンク オブジェクトに RX エンドポイント オブジェクトが含まれている必要があります。

構文

```
create_hw_sio_scan [-description <arg>] [-link_settings <arg>] [-quiet]
                  [-verbose] <scan_type> <hw_sio_object>
```

戻り値

新しいハードウェア SIO スキャン

使用法

名前	説明
[-description]	スキャンの説明を指定します。デフォルトはスキャン オブジェクト名です。
[-link_settings]	スキャンを実行する前に設定するリンク プロパティと値を指定します。デフォルトはありません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<scan_type>	スキャン タイプを指定します。有効な値は 1d_bathtub、2d_full_eye です。
<hw_sio_object>	スキャンを実行する RX エンドポイントまたはリンク オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

IBERT デバッグ コア上の指定の通信リンクにシリアル I/O 解析スキャン オブジェクトを作成します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用のアイ スキャン ハードウェアを使用してリンクのスキャンを実行すると有益です。Vivado シリアル I/O 解析機能では、リンク スキャンを作成、実行、および保存できます。

このコマンドは、run_hw_sio_scan コマンドで指定のリンクまたは GT レシーバーに対して解析を実行するのに使用可能なリンク スキャン オブジェクトを作成し、返します。write_hw_sio_scan コマンドを使用して、スキャンをディスクに保存することもできます。

作成したスキャン オブジェクトを削除するには、remove_hw_sio_scan コマンドを使用します。

このコマンドを実行すると、作成したハードウェア SIO スキャン (hw_sio_scan) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-description <arg>` (オプション): シリアル I/O 解析スキャンのラベルとして使用する簡単な説明を指定します。説明は、`<hw_sio_scan>` オブジェクトを特定するのに使用できます。たとえば、多数のポートをスキャンする際にスキャン プロットがどのポートのものであるかがわかるように、レシーバー ポートを特定できます。

`-link_settings <arg>` (オプション): スキャンを実行する前に設定するリンク プロパティと値を指定します。リンク設定を指定しない場合は、デフォルト設定が使用されます。スキャンのプロパティおよび設定については、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<scan_type>` (必須): スキャン タイプを指定します。有効なタイプは次のとおりです。

- `1d_bathtub`: 0 垂直軸のすべての水平サンプリング ポイントをスキャンします。
- `2d_full_eye`: すべての水平および垂直サンプリング ポイントをスキャンしてアイを作成します。

`<hw_sio_object>` (必須): スキャン オブジェクトを定義する IBERT デバッグ コア リンク (`hw_sio_link`) またはレシーバー (`hw_sio_rx`) を指定します。リンクまたはレシーバーは、`get_hw_sio_links` または `get_hw_sio_rxs` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `create_hw_sio_scan` コマンドでは、`hw_sio_object` を 1 つのオブジェクトのリストとして指定する必要があります。

例

次の例では、指定のリンクのスキャンを定義しています。

```
set xil_newScan [create_hw_sio_scan -description {LoopBack} 2d_full_eye \
[lindex [get_hw_sio_links *MGT_X0Y10/TX*] 0 ]]
run_hw_sio_scan [get_hw_sio_scans $xil_newScan]
```

関連項目

- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_scan](#)
- [run_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

create_hw_sio_sweep

新しいハードウェア SIO スイープを作成します。リンク オブジェクトを指定する場合は、そのリンク オブジェクトに RX エンドポイント オブジェクトが含まれている必要があります。

構文

```
create_hw_sio_sweep [-description <arg>] [-iteration_settings <arg>]
                    [-quiet] [-verbose] <scan_type> [<hw_sio_link>]
```

戻り値

新しいハードウェア SIO スイープ

使用法

名前	説明
[-description]	スイープの説明を指定します。デフォルトはスイープ オブジェクト名です。
[-iteration_settings]	スイープを実行する前に設定する各スキャンのリンク設定を指定します。デフォルトはありません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<scan_type>	スイープ タイプを指定します。有効な値は 1d_bathtub および 2d_full_eye です。
[<hw_sio_link>]	スイープを実行するリンク オブジェクトを指定します。デフォルトはありません。

カテゴリ

Hardware (ハードウェア)

説明

ある値の範囲で複数のスキャンを実行するためのシリアル I/O 解析リンク スイープ オブジェクトを作成します。

リンクのマージンを解析するには、ザイリンクス® UltraScale™ デバイスまたは 7 シリーズ FPGA 専用の機能を使用してリンクのスキャンを実行すると有益です。また、リンクに対して GT の異なる設定を使用して複数のスキャンを実行するのも有益です。デザインにどの設定が最適かを判断するのに役立ちます。Vivado® シリアル I/O 解析機能では、リンク スイープ (ある値の範囲で実行するリンク スキャンのグループ) を定義、実行、および保存できます。

このコマンドは、run_hw_sio_sweep コマンドで指定のリンクまたは GT レシーバーに対して解析を実行するのに使用可能なリンク スイープ オブジェクトを作成し、返します。write_hw_sio_sweep コマンドを使用して、スイープをディスクに保存することもできます。

作成したスイープ オブジェクトを削除するには、remove_hw_sio_sweep コマンドを使用します。

このコマンドを実行すると、作成したハードウェア SIO スイープ (hw_sio_sweep) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-description <arg>` (オプション): シリアル I/O 解析スイープ スキャンのラベルとして使用する簡単な説明を指定します。

`-iteration_settings <arg>` (オプション): 複数のスキャン間で変更するプロパティを指定します。反復設定の詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<scan_type>` (必須): スキャン タイプを指定します。有効なタイプは次のとおりです。

- `1d_bathtub`: 0 垂直軸のすべての水平サンプリング ポイントをスキャンします。
- `2d_full_eye`: すべての水平および垂直サンプリング ポイントをスキャンしてアイを作成します。



ヒント: bathtub スキャンの結果は `write_hw_sio_scan` コマンドを使用してファイルに保存できますが、プロットを `display_hw_sio_scan` コマンドを使用して Vivado シリアル I/O 解析機能で表示することはできません。

`<hw_sio_link>` (オプション): スイープ オブジェクトを定義するハードウェア SIO リンク (`hw_sio_link`) オブジェクトを指定します。リンクは、`get_hw_sio_links` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、作成した `hw_sio_sweep` オブジェクトを変数に割り当て、そのスイープ スキャンを実行しています。

```
set xil_newSweep [create_hw_sio_sweep -description {Sweep 0} 2d_full_eye \
    [lindex [get_hw_sio_links *MGT_X0Y10/TX*] 0 ]]
run_hw_sio_sweep [get_hw_sio_sweeps $xil_newSweep]
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_scan](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

create_hw_target

hw_target (JTAG チェーン) を作成して名前を指定します。

構文

```
create_hw_target [-copy <arg>] [-quiet] [-verbose] <target_name>
```

戻り値

ハードウェア ターゲット

使用法

名前	説明
[-copy]	ハードウェア ターゲットのコピーを指定します。デフォルトは、現在のターゲットのコピーです。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<target_name>	作成するハードウェア ターゲットの名前を指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado ハードウェア マネージャーでは、SVF (Serial Vector Format) ファイルを使用したハードウェア デバイスのプログラムがサポートされています。SVF ファイルは、プログラム命令とコンフィギュレーション データを含む ASCII ファイルです。これらのファイルは、ATE マシンおよびエンベデッド コントローラーでバウンダリスキャン操作を実行するために使用されます。SVF ファイルには、ビットストリームをザイリンクス デバイスに直接プログラム、またはフラッシュ メモリ デバイスに間接プログラムするのに必要な JTAG コマンドが含まれます。SVF ファイルは、`write_hw_svf` コマンドを使用して記述し、`execute_hw_svf` コマンドを使用して開いているハードウェア ターゲット (hw_target) に適用します。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

hw_svf ファイルを作成するプロセスは、次のとおりです。

1. `create_hw_target` コマンドを使用して SVF ターゲットを作成します。
2. SVF ターゲットを開きます。
3. `create_hw_device` コマンドを使用して、SVF ターゲットに 1 つまたは複数のデバイスを作成します。
4. `program_hw_devices` などのコマンドを使用してデバイスをプログラムします。
5. `write_hw_svf` コマンドを使用して操作コマンドの SVF ファイルを記述します。

`create_hw_target` コマンドは、デバイスをプログラムし、プログラム コマンドを SVF ファイルにエクスポートするためのプラットフォームとして使用可能な SVF `hw_target` オブジェクトを現在の `hw_server` 上に作成します。SVF ターゲットは、ほかの `hw_target` と同様に、`get_hw_targets` や `current_hw_target` などのコマンドを使用してクエリおよび管理できます。

注記: SVF フローを使用する場合は、SVF ターゲットは実際のハードウェア ボードまたはデバイスに接続する必要はないので、ご使用のシステム上のローカル `hw_server` に接続することをお勧めします。

SVF `hw_target` は、`get_property` または `report_property` コマンドを使用して返すことができる `IS_SVF` ブールプロパティで特定できます。このプロパティは、SVF ターゲットに対しては `TRUE` に設定されています。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-copy <arg>` (オプション): 既存の `hw_target` をコピーして新しい SVF `hw_target` を作成します。引数には、物理 `hw_target` または SVF `hw_target` を `get_hw_targets` コマンドを使用して指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<target_name>`: 新しい SVF `hw_target` の名前を指定します。

例

次の例では、指定の `hw_target` をコピーして SVF `hw_target` オブジェクトを作成しています。

```
create_hw_target -copy [get_hw_targets *210203327996A] svfTarget
```

次の例では、現在定義されている SVF `hw_target` オブジェクトを取得しています。

```
get_hw_targets -filter {IS_SVF}
```

次の例では、SVF フローに必要なすべてのコマンドを示しています。まず Vivado ハードウェア マネージャーを開いてローカル `hw_server` に接続し、SVF `hw_target` を作成して開き、`hw_device` を追加してこのデバイスにビットストリームをプログラムして、このデバイス用のプログラム コマンドを SVF ファイルに記述しています。

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
write_hw_svf my_xc7k325t.svf
close_hw_target
```

関連項目

- [connect_hw_server](#)
- [create_hw_device](#)
- [current_hw_target](#)
- [get_hw_targets](#)
- [get_property](#)
- [program_hw_devices](#)
- [report_hw_targets](#)
- [report_property](#)
- [set_property](#)
- [write_hw_svf](#)

create_interface

I/O ポート インターフェイスを作成します。

構文

```
create_interface [-parent <arg>] [-quiet] [-verbose] <name>
```

戻り値

新しいインターフェイス オブジェクト

使用法

名前	説明
[-parent]	新しいインターフェイスを割り当てる親インターフェイスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	新しい I/O ポート インターフェイスの名前を指定します。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

スカラー ポートまたは差動 I/O ポートをグループ化するための新しいインターフェイスを作成します。

引数

-parent <arg> (オプション): 新しいインターフェイスを割り当てる親インターフェイスを指定します。

注記: 指定した親インターフェイスが存在しない場合、エラー メッセージが表示されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<name> (必須): 作成する I/O ポート インターフェイスの名前を指定します。

例

次の例では、新しい USB インターフェイスが作成されます。

```
create_interface USB0
```

次の例では、指定した親インターフェイスにイーサネット インターフェイスが作成されます。

```
create_interface -parent Top_Int ENET0
```

関連項目

- [delete_interface](#)
- [create_port](#)
- [make_diff_pair_ports](#)
- [place_ports](#)
- [remove_port](#)
- [set_package_pin_val](#)
- [split_diff_pair_ports](#)

create_ip

設定可能な IP のインスタンスを作成して、ファイルセットに追加します。

構文

```
create_ip [-vlnv <arg>] -module_name <arg> [-dir <arg>] [-force]
          [-allow_hidden] [-vendor <arg>] [-library <arg>] [-name <arg>]
          [-version <arg>] [-revision <arg>] [-quiet] [-verbose]
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
[-vlnv]	新しい IP の作成元となる IP カタログの VLNv 文字列 (<Vendor>:<Library>:<Name>:<Version>) を指定します。
-module_name	プロジェクトに追加する新しい IP の名前を指定します。
[-dir]	プロジェクト外で作成し、管理するリモート IP へのディレクトリパスを指定します。
[-force]	既存の IP インスタンスを上書きします。-dir オプションを使用している場合にのみ使用できます。
[-allow_hidden]	隠しコアをインスタンス化できるようにします。
[-vendor]	IP ベンダーの名前を指定します。
[-library]	IP ライブラリの名前を指定します。
[-name]	IP 名
[-version]	IP バージョン
[-revision]	IP コアのリビジョンを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

IPFlow (IP フロー)

説明

IP カタログからの設定可能な IP の XCI ファイルを作成し、現在のプロジェクトのソース ファイルに追加します。ネットリストに IP コアのインスタンスを作成するには、作成した IP ソース オブジェクトを HDL ファイルにインスタンス化する必要があります。

同じコアのインスタンスが複数必要な場合は、HDL デザインにコアのモジュールを必要な数だけインスタンス化します。同じ IP コアを異なる設定で使用する場合は、create_ip コマンドを使用して別の IP ソース オブジェクトを作成する必要があります。

`create_ip` は、現在の IP カタログから IP コアをインポートするために使用します。カタログに IP を追加せずに XCI および XCO ファイルを直接読み込むには、`import_ip` コマンドを使用します。

このコマンドは IP 生成プロセスと、インポートされた IP コア ファイルのパスと名前を返します。

注記: IP コアは Vivado にネイティブであり、Vivado ツールでカスタマイズおよび再生成できます。`convert_ip` コマンドを使用すると、レガシ IP を Vivado でサポートされているネイティブ IP に変換できます。

引数

`-vlnv <arg>` (オプション): 新しい IP の作成元となる IP カタログの VLNv 文字列を指定します。VLNV は `<Vendor>:<Library>:<Name>:<Version>` 形式の文字列で、カタログ内の IP を識別します。VLNV 文字列は IP コアの IPDEF プロパティにマップされています。

注記: `-vlnv` を指定するか、`-vendor`、`-library`、`-name`、および `-version` すべてを指定する必要があります。

`-module_name <arg>` (必須): 作成する新しい IP インスタンスの名前を指定します。モジュールは、`<module_name>/<module_name>.xci` という名前で作成されます。

`-dir <arg>` (オプション): IP コア ファイルを保存するディレクトリを指定します。このオプションを指定しない場合、IP ファイル (`.xci`、`.veo` など) は `<project_name>.srcs` ディレクトリの階層に保存されます。

`-force` (オプション): 指定したディレクトリに `<module_name>` で指定したのと同じ名前の IP インスタンスが存在する場合に上書きします。既存の IP に関連付けられているファイルもすべて、新しく作成される IP に関連付けられているファイルで上書きされます。このオプションは、`-dir` オプションを使用している場合にのみ使用可能です。

`-vendor <arg>` (オプション): IP を作成したベンダーの名前を指定します。

`-library <arg>` (オプション): コアの追加元の IP ライブラリを指定します。

`-name <arg>` (オプション): カタログ内での IP コアの名前を指定します。

`-version <arg>` (オプション): IP コアのバージョン場号を指定します。

注記: `-vlnv` を指定するか、`-vendor`、`-library`、`-name`、および `-version` すべてを指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`-vlnv` オプションで指定した IP コアをインポートし、現在のプロジェクトでのモジュール名を指定しています。

```
create_ip -vlnv xilinx.com:ip:c_addsub:11.0 -module_name test_addr
```

次の例では、Vivado で `-vendor`、`-library`、`-name`、`-version` を指定して IP ブロックを作成し、モジュール名を指定しています。IP を作成したら、`set_property` コマンドを使用して IP の属性をカスタマイズし、インスタレーション テンプレートおよび合成ターゲットを生成します。

```
create_ip -name c_addsub -version 11.0 -vendor xilinx.com -library ip \
  -module_name c_addsub_v11_0_0
set_property CONFIG.COMPONENT_NAME c_addsub_v11_0_0 \
  [get_ips c_addsub_v11_0_0]
set_property CONFIG.A_WIDTH 32 [get_ips c_addsub_v11_0_0]
set_property CONFIG.B_WIDTH 32 [get_ips c_addsub_v11_0_0]
set_property CONFIG.ADD_MODE Add_Subtract [get_ips c_addsub_v11_0_0]
set_property CONFIG.C_IN true [get_ips c_addsub_v11_0_0]
generate_target {instantiation_template synthesis} \
  [get_files C:/Data/c_addsub_v11_0_0/c_addsub_v11_0_0.xci \
    -of_objects [get_filesets sources_1]]
```

関連項目

- [generate_target](#)
- [import_ip](#)
- [upgrade_ip](#)
- [validate_ip](#)

create_ip_run

指定した IP の run を作成します。

構文

```
create_ip_run [-force] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-force]</code>	指定した IP の生成ファイルを強制的に再生成します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	run を生成する必要があるすべての IP オブジェクトを指定します (get_ips または get_files で指定)。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

get_ips コマンドを使用して指定した IP オブジェクト、または get_files コマンドを使用して指定した IP コア ファイル (XCI) の合成 run およびインプリメンテーション run を指定します。

IP run は、アウト オブ コンテキスト (OOC) IP フローをサポートするために合成済みデザイン チェックポイント ファイル (DCP) を生成するか、または OOC 階層デザイン フローで IP モジュールを合成およびインプリメントするために使用します。

合成用に 1 つ、インプリメンテーション用に 1 つ、合計 2 つの run が作成されます。run の名前は、IP コアと run タイプから <ip-name>_synth_1 および <ip-name>_impl_1 になります。

run を合成するのに必要な IP ソース ファイルは、IP run ディレクトリにコピーされます。run の属性は、set_property コマンドを使用して設定できます。

このコマンドを実行すると、IP モジュール用に作成された合成 run の名前が返されます。

引数

`-force` (オプション): 指定の IP に対する出力ファイルが生成されており、すべて最新であっても、IP run を強制的に作成します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<object> (必須): `get_ips` コマンドを使用して IP オブジェクトを指定するか、`get_files` コマンドを使用して IP コア ファイル (XCI) を指定します。



ヒント: 1 つの IP のみを指定します。

例

次の例では、指定した IP モジュールの合成 run およびインプリメンテーション run を作成しています。

```
create_ip_run [get_ips add1]
```

関連項目

- [create_run](#)
- [get_files](#)
- [get_ips](#)
- [set_property](#)

create_macro

マクロを作成します。

構文

```
create_macro [-quiet] [-verbose] <name>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	作成するマクロの名前を指定します。

カテゴリ

[XDC](#)

説明

セルを相対配置するためのマクロを作成します。

マクロは、リソース効率を改善し内部接続を高速にするため、関連のセルをグループ化して一緒に配置するために主に使用します。`create_macro` コマンドを使用すると、RTL ソース ファイルで RLOC 制約を使用して定義されている RPM と同様に、合成済みデザインまたはインプリメント済みデザインで `place_design` コマンドで相対配置されるようマクロを定義できます。相対配置マクロの定義方法は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

マクロを作成したら、`update_macro` コマンドを使用して特定のセルをマクロに割り当てることができます。定義済みのマクロを変更するには、`delete_macro` コマンドを使用してマクロを削除してからマクロを再作成し、新しい内容でマクロをアップデートする必要があります。既存のマクロを単に上書きすることはできません。

定義済みのマクロを削除するには、`delete_macro` コマンドを使用します。デザインで現在定義されているマクロをリストするには、`get_macros` コマンドを使用します。

このコマンドを実行しても、何も返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 作成するマクロの名前を指定します。

例

次の例では、usbMacro1 というマクロを作成しています。

```
create_macro usbMacro1
```

関連項目

- [delete_macros](#)
- [get_macros](#)
- [place_design](#)
- [update_macro](#)

create_net

現在のデザインにネットを作成します。

構文

```
create_net [-from <arg>] [-to <arg>] [-quiet] [-verbose] <nets>...
```

使用法

名前	説明
[-from]	開始バス インデックスを指定します。
[-to]	終了バス インデックスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<nets>	作成するネットの名前を指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

開いている合成済みデザインまたはインプリメント済みデザインのネットリストにネットを追加します。

注記: ネットをライブラリ マクロまたはマクロ プリミティブに追加することはできません。

ネットはデザインの最上位に作成するか、階層ネット名を指定して任意の階層レベルに作成できます。

バス ネットは、バス インデックス値に正または負の値を使用することにより、増加するバス インデックスまたは減少するバス インデックスで作成できます。

新しいネットは、作成したときにはネットリスト内では接続されていません。connect_net コマンドを使用して接続する必要があります。接続されているネットの接続を解除するには disconnect_net コマンドを使用し、ネットリストから削除するには remove_net コマンドを使用します。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、write_checkpoint コマンドを使用してデザイン チェックポイントとして保存するか、write_* コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

-from <arg> (オプション): 新しいバスの開始インデックスを指定します。

-to <arg> (オプション): 新しいバスの終了インデックスを指定します。

注記: `-from` または `-to` を片方だけ指定すると、`-from` または `-to` で指定したインデックス値の 1 ビット バスが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<nets> (必須): 作成するネットの名前を指定します。ネット名は最上位から指定するか、名前のみ (net1) で指定するか、階層ネット名 (cell1/cellA/net1) を指定してデザイン階層内で指定できます。

例

次の例では、現在の合成済みデザインまたはインプリメント済みデザインに 24 ビット バスを作成しています。

```
create_net tempBus -from 23 -to 0
```

関連項目

- [connect_net](#)
- [create_pin](#)
- [create_port](#)
- [disconnect_net](#)
- [get_nets](#)
- [remove_net](#)
- [resize_net_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

create_partition_def

新しいパーティション定義を作成します。

構文

```
create_partition_def -name <arg> -module <arg> [-library <arg>] [-quiet]
                    [-verbose]
```

使用法

名前	説明
-name	パーティション定義の名前を指定します。
-module	パーティション定義のモジュール名を指定します。
[-library]	パーティション定義のモジュールのライブラリ名を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)、Partition (パーティション)

説明



重要: まず、プロジェクトの PR_FLOW プロパティを TRUE に設定するか、[Tools]→[Enable Partial Reconfiguration] コマンドを使用して、プロジェクトをパシアル リコンフィギュレーション (PR) プロジェクトとして定義する必要があります。

パシアル リコンフィギュレーション フローでは、デザインの階層セルからパーティション定義 (partitionDef) を作成でき、これらの partitionDef にリコンフィギュラブル モジュール (RM) を割り当てることにより、コア デザインと RM の組み合わせに基づいて独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパシアル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』(UG909) を参照してください。

create_partition_def コマンドは、指定の階層セルから PR プロジェクトに partitionDef オブジェクトを定義します。partitionDef は、特定の PR コンフィギュレーション用に RM を割り当てること可能なパーティション階層を定義します。

このコマンドを実行すると、作成された partitionDef の名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-name <arg> (必須): 新しい partitionDef の名前を指定します。

`-module <arg>` (必須): `partitionDef` を定義する現在のプロジェクトの階層セルまたはモジュールを指定します。階層セルは名前で指定できます。モジュールは、パーティション定義の階層境界を指定します。リコンフィギャラブル モジュール (RM) は、`partitionDef` の階層境界内に収まるように定義されます。

`-library <arg>` (オプション): `partitionDef` に割り当てるライブラリ名を指定します。ライブラリ名を指定しない場合、`xil_defaultlib` が使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のプロジェクトに `PR_FLOW` プロパティを設定し、新しい `partitionDef` を定義して指定のモジュールに割り当てた後、指定の階層からリコンフィギャラブル モジュールを作成しています。

```
set_property PR_FLOW 1 [current_project]
create_partition_def -name partDef1 -module fftTop \
-library xil_defaultlib
create_reconfig_module -name fftTop -define_from fftTop \
-partition_def [get_partition_defs partDef1]
```

関連項目

- [create_pr_configuration](#)
- [create_reconfig_module](#)
- [delete_partition_defs](#)
- [get_partition_defs](#)
- [set_property](#)

create_pblock

新しい Pblock を作成します。

構文

```
create_pblock [-quiet] [-verbose] <name>
```

戻り値

新しい Pblock オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	新しい Pblock の名前を指定します。

カテゴリ

[XDC](#)、[Floorplan \(フロアプラン\)](#)

説明

フロアプラン用にロジック インスタンスを追加する Pblock を定義します。

Pblock にロジック エレメントを追加するには `add_cells_to_pblock` コマンド、Pblock を FPGA ファブリック上に配置するには `resize_pblocks` コマンドを使用します。`resize_pblock` コマンドを使用すると、Pblock を手動で移動およびサイズ変更できます。

2 番目の例に示すように、`set_property` コマンドを使用して PARENT プロパティを設定すると、階層フロアプランのため 1 つの Pblock を別の Pblock に含めることができます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成する Pblock の名前を指定します。

例

次の例では、pb_cpuEngine という Pblock を作成し、cpuEngine モジュールに含まれる最下位セルのみを追加し、配置済みのインスタンスの配置制約を消去します。

```
create_pblock pb_cpuEngine
add_cells_to_pblock pb_cpuEngine [get_cells cpuEngine/*] \
    -add_primitives -clear_locs
```

次の例では、pb_usbEngine0、pb_usbEngine1、pb_usbTop という 3 つの Pblock を作成し、set_property コマンドを使用して pb_usbEngine0 および pb_usbEngine1 を pb_usbTop の下に挿入しています。

```
create_pblock pb_usbEngine0
create_pblock pb_usbEngine1
create_pblock pb_usbTop
set_property PARENT pb_usbTop [get_pblocks {pb_usbEngine?}]
```

関連項目

- [add_cells_to_pblock](#)
- [get_pblocks](#)
- [place_pblocks](#)
- [resize_pblock](#)
- [set_property](#)

create_peripheral

指定の VLNV でペリフェラルを作成します。

構文

```
create_peripheral [-dir <arg>] [-quiet] [-verbose] <vendor> <library>
                  <name> <version>
```

使用法

名前	説明
[-dir]	プロジェクト外で作成し、管理するリモート ペリフェラルへのディレクトリ パスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<vendor>	ベンダー (xilinx.com など) を指定します。
<library>	ライブラリ (ip など) を指定します。
<name>	名前 (myip など) を指定します。
<version>	バージョン (1.4 など) を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[CreatePeripheral \(ペリフェラルの作成\)](#)

説明

IP リポジトリに追加する AXI ペリフェラルを指定の VLNV 属性で作成します。

作成した AXI ペリフェラルは、`add_peripheral_interface` コマンドを使用してインターフェイスを追加し、`generate_peripheral` コマンドを使用して生成するまでは、単なる枠組みです。

引数

`-dir <arg>` (オプション): AXI ペリフェラル データ ファイルを保存する出力ディレクトリを指定します。デフォルトでは、作成されたペリフェラルは現在のプロジェクトのソース ディレクトリ構造 `./project_name.srsc/sources_1/ip` に追加されます。

注記: AXI ペリフェラルが現在のプロジェクト外に保存する場合は、指定のディレクトリを `set_property` コマンドを使用して現在のファイルセットの `IP_REPO_PATH` プロパティに追加し、ペリフェラルを IP カタログから使用できるようにする必要があります。

```
set_property IP_REPO_PATHS {C:/Data/axi_peripheral/ourIP_2.1}
[current_fileset]
update_ip_catalog -rebuild
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<vendor>`: VLVN 属性のベンダー部分を指定します。VLNV 属性は、AXI ペリフェラルの IP カタログでの位置を定義します。VLNV は `<Vendor>:<Library>:<Name>:<Version>` 形式の文字列で、カタログ内の IP を識別します。

`<library>`: VLVN 属性のライブラリ部分を指定します。

`<name>`: VLVN 属性の名前部分を指定します。

`<version>`: VLVN 属性のバージョン部分を指定します。

例

次の例では、AXI ペリフェラルを指定の VLVN 属性で作成しています。

```
create_peripheral {myCompany.com} {user} {testAXI1} {1.3}
  -dir {C:/Data/new_periph}
```

次の例では、VLNV 属性を指定して新しい AXI ペリフェラルを作成し、そのペリフェラル オブジェクトを後の処理用に Tcl 変数に格納して、そのペリフェラルに AXI スレーブ インターフェイスを追加しています。

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
  {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
  -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
  -axi_type {lite} $perifObj
add_peripheral_interface {S2_AXI} -interface_mode {slave} \
  -axi_type {lite} $perifObj
```

関連項目

- [add_peripheral_interface](#)
- [generate_peripheral](#)
- [write_peripheral](#)

create_pin

現在のデザインにピンを作成します。

構文

```
create_pin [-from <arg>] [-to <arg>] -direction <arg> [-quiet] [-verbose]  
<pins>...
```

使用法

名前	説明
[-from]	開始バス インデックスを指定します。
[-to]	終了バス インデックスを指定します。
-direction	ピンの方向を指定します。有効な値は、IN、OUT、INOUT です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pins>	作成するピンの名前を指定します。

カテゴリ

Netlist (ネットリスト)

説明

開いている合成済みデザインまたはインプリメント済みデザインのネットリストにシングル ピンまたはバス ピンを追加します。ピン名だけでなく、方向やバス幅などの属性も指定できます。

バス ピンは、バス インデックス値に正または負の値を使用することにより、増加するバス インデックスまたは減少するバス インデックスで作成できます。

ピンは、既存のセル インスタンスに作成するか、`create_port` コマンドを使用して最上位ピンとして作成する必要があります。ピン名の一部としてセルのインスタンス名を指定しない場合、エラーが返されます。

注記: ピンをライブラリ マクロまたはマクロ プリミティブに追加することはできません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイル セットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

-from <arg> (オプション): バス ピンの開始インデックスを指定します。

-to <arg> (オプション): バス ピンの終了インデックスを指定します。

`-direction` (必須): ピンの方向を指定します。有効な値は IN、OUT、INOUT です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pins>` (必須): 作成するピンの名前を指定します。ピン名は、ピンを割り当てるセル インスタンスを基準に階層で指定する必要があります。デザインの最上位に作成されたピンはポートであり、`create_port` コマンドで作成する必要があります。

例

次の例では、cpuEngine モジュールに指定した名前を入力ピンを作成しています。

```
create_pin -direction IN cpuEngine/inPin
```

次の例では、まず階層区切り文字を設定し、参照セルのブラック ボックス インスタンスを作成して、そのインスタンスに 24 ビットの双方向バスを作成しています。

```
set_hierarchy_separator |
create_cell -reference dmaBlock -black_box usbEngine0|myDMA
create_pin -direction INOUT -from 0 -to 23 usbEngine0|myDMA|dataBus
```

関連項目

- [create_cell](#)
- [create_net](#)
- [create_port](#)
- [connect_net](#)
- [disconnect_net](#)
- [remove_cell](#)
- [remove_pin](#)
- [resize_pin_bus](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

create_port

スカラー ポートまたはバス ポートを作成します。

構文

```
create_port -direction <arg> [-from <arg>] [-to <arg>] [-diff_pair]
          [-interface <arg>] [-quiet] [-verbose] <name> [<negative_name>]
```

戻り値

作成されたポート オブジェクトのリスト

使用法

名前	説明
-direction	ポートの方向を指定します。有効な値は IN、OUT、INOUT です。
[-from]	新しいバスの開始インデックスを指定します。
[-to]	新しいバスの終了インデックスを指定します。
[-diff_pair]	ポートの差動ペアを作成します。
[-interface]	ポートを割り当てるインターフェイスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	ポート名を指定します。
[<negative_name>]	差動ペアの N 側の名前を指定します (オプション)。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

方向、幅、シングルエンドか差動かなどのパラメーターを指定してポートを作成し、オプションで既存のインターフェイスに割り当てます。新しいポートは、デザイン階層の最上位に追加されます。

バス ポートは、バス インデックス値に正または負の値を使用することにより、増加するバス インデックスまたは減少するバス インデックスで作成できます。

create_port コマンドを使用すると、I/O プランニング プロジェクトで新しいポートを作成したり、開いている合成済みデザインまたはインプリメント済みデザインのネットリストを編集中に新しいポートを作成できます。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、write_checkpoint コマンドを使用してデザイン チェックポイントとして保存するか、write_* コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-direction` (必須): ポートの方向を指定します。有効な値は IN、OUT、INOUT です。

`-from <arg>` (オプション): 新しいバスの開始インデックスを指定します。負のインデックス値から開始することも可能です。

`-to <arg>` (オプション): 新しいバスの終了インデックスを指定します。負のインデックス値で終了することも可能です。

`-diff_pair` (オプション): ポートを差動ペアとして作成します。P 側と N 側の両方のポートが作成されます。
`<name>` のみを指定した場合、P 側のポートの名前が `<name>` になり、N 側のポートは `<name>_N` という名前になります。
`<name>` と `<negative_name>` の両方を指定した場合、P 側のポートの名前が `<name>` になり、N 側のポートの名前が `<negative_name>` になります。

`-interface <arg>` (オプション): ポートを割り当てるインターフェイスを指定します。

注記: `create_interface` コマンドを使用してインターフェイスを定義しておく必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成するポートの名前を指定します。`-diff_pair` を指定した場合、`<name>` は P 側のポートの名前となり、N 側のポートの名前は `<name>_N` となります。

`<negative_name>` (オプション): `-diff_pair` を使用する場合に N 側のポートの名前を指定します。この場合、`<name>` が P 側のポートの名前になり、`<negative_name>` が N 側のポートの名前になります。

例

次の例では、PORT0 という名前の入力ポートを作成しています。

```
create_port -direction IN PORT0
```

次の例では、まず Group1 という新しいインターフェイスを作成し、その後そのインターフェイスを使用して 4 ビットの差動ペア出力バスを作成しています。バス ポートが差動ペアとして定義されており、`<name>` のみが指定されているので、N 側のポートの名前は D_BUS_N になります。

```
create_interface Group1
create_port -direction OUT -from 0 -to 3 -diff_pair -interface Group1 D_BUS
```

注記: このコマンドにより、D_BUS[0]、D_BUS_N[0]、D_BUS[1]、D_BUS_N[1]、D_BUS[2]、D_BUS_N[2]、D_BUS[3]、D_BUS_N[3] という 8 本のポートが作成されます。

次のコマンドでは、`<name>` のみを指定し、data および data_N という差動ペア出力ポートを作成しています。

```
create_port -direction OUT -diff_pair data
```

次のコマンドでは、<name> と <negative_name> の両方を指定し、data_P および data_N という差動ペア出力ポートを作成しています。

```
create_port -direction OUT -diff_pair data_P data_N
```

関連項目

- [create_interface](#)
- [make_diff_pair_ports](#)
- [place_ports](#)
- [remove_port](#)
- [resize_port_bus](#)
- [split_diff_pair_ports](#)

create_port_on_reconfigurable_module

指定したフィギャラブル セルにポートを生成します。

構文

```
create_port_on_reconfigurable_module [-cell <arg>] [-port <arg>]
    [-direction <arg>] [-from <arg>] [-to <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-cell]	(必須) ポート パンチング用に HD.RECONFIGURABLE セル名を指定します。
[-port]	(必須) HD.RECONFIGURABLE セルに新しく追加したポートの名前を指定します。
[-direction]	(必須) ポートの方向を指定します。有効な値は INPUT、OUTPUT、または INOUT です。
[-from]	(オプション) ポート バスの開始インデックスを指定します。デフォルトは -1 です。
[-to]	(オプション) ポート バスの終了インデックス指定します。デフォルトは -1 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

create_pr_configuration

新しいコンフィギュレーションを作成します。

構文

```
create_pr_configuration -name <arg> [-partitions <args>]
                        [-greyboxes <args>] [-use_netlist] [-quiet] [-verbose]
```

使用法

名前	説明
-name	パーティションの名前を指定します。
[-partitions]	パーティション インスタンスとリコンフィギュラブル モジュールのペアを指定します。
[-greyboxes]	すべてのポートにバッファを挿入する必要があるインスタンスをリストします。
[-use_netlist]	ネットリストを使用してパーティション定義のインスタンスを取得し、コンフィギュレーションを作成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)、Partition (パーティション)

説明



重要: まず、プロジェクトの PR_FLOW プロパティを TRUE に設定するか、[Tools]→[Enable Partial Reconfiguration] コマンドを使用して、プロジェクトをパシカル リコンフィギュレーション (PR) プロジェクトとして定義する必要があります。

パシカル リコンフィギュレーション フローでは、デザインの階層セルからパーティション定義 (partitionDef) を作成でき、これらの partitionDef にリコンフィギュラブル モジュール (RM) を割り当てることにより、スタティック デザインと 1 つ以上の RM の組み合わせに基づく独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパシカル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

create_pr_configuration コマンドは、スタティック ロジックと RM の組み合わせを定義し、デザインの固有のコンフィギュレーションを作成します。PR コンフィギュレーションは、インプリメントしてビットストリームするデザインです。

create_run -pr_config コマンドを使用して、PR コンフィギュレーション用のインプリメント run を作成する必要があります。

このコマンドを実行すると、作成された PR コンフィギュレーションの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-name <arg> (必須): 新しい PR コンフィギュレーションの名前を指定します。

-partitions arg (オプション): PR コンフィギュレーションのインスタンスに適用するパーティション インスタンスとリコンフィギュラブル モジュールを指定します。引数は <partitionInstance>:<RM> の形式で指定し、複数のパーティション/モジュールのペアを定義する場合はリストとして指定します。この形式により、デザインに含まれる 1 つの partitionDef の複数のインスタンスに異なる RM を割り当てることができます。

-greyboxes <arg> (オプション): 指定のパーティション インスタンスが LUT1 およびシングル バッファァーが挿入されるグレー ボックスとして定義されることを示します。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、新しい PR コンフィギュレーションを定義し、パーティションに割り当てています。

```
create_pr_configuration -name prConfig1 -partitions fftEngine:fftTop
```

次の例では、3 つの PR コンフィギュレーションを定義しています。2 つは partitionDef と異なる RM をパーティションに割り当てて定義し、1 つはグレー ボックスを含むからのコンフィギュレーションとして定義します。

```
create_pr_configuration -name cfg1 -partitions \
[list fftEngine:fftTop mgtEngine:mgtTop]
create_pr_configuration -name cfg2 -partitions \
[list fftEngine:fftTop mgtEngine:mgtBottom]
create_pr_configuration -name cfg3 -partitions { } \
-greyboxes [list fftEngine mgtEngine]
```

関連項目

- [create_partition_def](#)
- [create_reconfig_module](#)
- [current_pr_configuration](#)
- [delete_pr_configurations](#)
- [get_pr_configurations](#)
- [report_pr_configuration_analysis](#)
- [setup_pr_configurations](#)

create_project

新しいプロジェクトを作成します。

構文

```
create_project [-part <arg>] [-force] [-in_memory] [-ip] [-rtl_kernel]
               [-quiet] [-verbose] [<name>] [<dir>]
```

戻り値

新しいプロジェクト オブジェクト

使用法

名前	説明
[-part]	ターゲット パーツを指定します。
[-force]	既存のプロジェクト ディレクトリを上書きします。
[-in_memory]	インメモリ プロジェクトを作成します。
[-ip]	IP 管理プロジェクトでの GUI のデフォルト動作を指定します。
[-rtl_kernel]	RTL カーネル プロジェクトでの GUI のデフォルト動作を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	プロジェクト名を指定します。
[<dir>]	プロジェクト ファイルを保存するディレクトリを指定します。デフォルトは . です。

カテゴリ

[Project \(プロジェクト\)](#)

説明

Vivado Design Suite プロジェクト ファイル (.xpr) または Vivado Lab Edition 用のプロジェクト ファイル (.lpr) を指定したディレクトリに作成します。

Vivado Lab Edition の場合: create_project コマンドの構文は異なり、オプション数は少なくなっています。Vivado Lab Edition でサポートされていないオプションは、次のとおりです。

- -part: Vivado Lab Edition プロジェクト (.lpr) ではターゲット パーツは指定されません。ターゲット パーツは、current_hw_target および current_hw_device で判断されます。
- -ip: Vivado Lab Edition では、プロジェクトは Manage IP フロー用には定義されません。

Vivado Design Suite の場合: Vivado Design Suite で作成されるデフォルト プロジェクトは RTL プロジェクトです。プロジェクトはコンテナーとして定義され、ソース ファイルセットの RTL ソース ファイルを管理します。プロジェクトのタイプは、プロジェクトを作成したときにソース ファイルセットの `DESIGN_MODE` プロパティで決定されます。プロジェクト タイプを変更するには、次のように `set_property` コマンドを使用して `current_fileset` の `DESIGN_MODE` プロパティを設定します。

- RTL プロジェクト: `set_property DESIGN_MODE RTL [current_fileset]`
- ネットリスト プロジェクト: `set_property DESIGN_MODE GateLvl [current_fileset]`
- I/O プランニング プロジェクト: `set_property DESIGN_MODE PinPlanning [current_fileset]`

異なるプロジェクト タイプの詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスと作成されたプロジェクトの名前が返され、正常に実行されなかった場合はエラーが返されます。

索引

`-part <arg>` (オプション): プロジェクトで使用するザイリンクス パーツを指定します。これは、プロジェクトの作成後に変更できます。`-part` オプションを指定しない場合、デフォルトのパーツが使用されます。このオプションは Vivado Lab Edition ではサポートされていません。

`-force` (オプション): 既存のプロジェクトを上書きします。指定した `<dir>` に指定のプロジェクト名が既に存在する場合、`-force` オプションを使用して既存のプロジェクトを上書きする必要があります。

注記: 既存プロジェクトを現在ツールで開いている場合、新しいプロジェクトでディスクの既存プロジェクトが上書きされますが、両方のプロジェクトがツールで開いたままになります。この場合、`create_project` を実行する前に `close_project` コマンドを実行しておくことをお勧めします。

`-in_memory` (オプション): 非プロジェクト デザイン フローをサポートするため、プロジェクトを Vivado Design Suite のインメモリ データベース内に作成するよう指定します。プロジェクト ファイルおよびディレクトリ構造はディスクに保存されません。インメモリ プロジェクトの目的は、通常プロジェクト ベース デザインに関連付けられているプロパティを非プロジェクト デザイン フローのインメモリ デザインに関連付けることができるようにすることです。



ヒント: `-in_memory` オプションの使用は標準の非プロジェクト デザイン フローの一部ではなく、Vivado ツールで予期しない動作が発生することがあります。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

`-ip` (オプション): Manage IP フローで IP カタログの IP を表示およびカスタマイズし、設定済みの IP のリポジトリを管理するためのプロジェクトを作成します。Manage IP フローの詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896) を参照してください。このオプションは Vivado Lab Edition ではサポートされていません。

`-rtl_kernel` (オプション): プロジェクトを Vitis コア開発キット用の RTL カーネルを定義するために作成することを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): この引数はパラメーター名を必要としませんが、**<dir>** よりも前に指定する必要があります。このコマンドにはパラメーターがないので、最初の引数は **<name>**、2 つ目の引数は **<dir>** として認識されます。プロジェクトファイルは **<name>.xpr** (または **<name>.lpr**) という名前で作成され、**<name>.data** という名前のプロジェクトデータ フォルダも作成されて、指定のディレクトリ **<dir>** に配置されます。

注記: 作成されるプロジェクト ファイルは、デフォルトでは RTL ソース ファイルです。`set_property` コマンドを使用して `DESIGN_MODE` プロパティを設定し、RTL ソース プロジェクトからネットリストまたは I/O ピン プランニングに変更する必要があります。

<dir> (オプション): 新しいプロジェクト ファイルを保存するディレクトリ名を指定します。指定したディレクトリが存在しない場合は、その名前の新しいディレクトリが作成されます。ディレクトリを完全なパスで指定すると、その指定したパス名が使用されます。**<dir>** をパスなしで指定すると、現在の作業ディレクトリまたはツールの起動ディレクトリでそのディレクトリが検索されるか、作成されます。

注記: プロジェクトを GUI モードで作成すると、**<dir>** に **<name>** を付けた **<dir>/<name>** という名前のプロジェクトディレクトリが作成され、新しいプロジェクト ファイルとプロジェクト データ フォルダが保存されます。

例

Vivado Design Suite から実行した場合、次の例では `myDesigns` というディレクトリに `project1.xpr` というプロジェクトが作成されます。

```
create_project project1 myDesigns
```

注記: **<dir>** にはフォルダ名のみが指定されているので、プロジェクトは現在の作業ディレクトリかツールの起動ディレクトリに作成されます。

Vivado Lab Edition から実行した場合、次の例では `myDesigns` というディレクトリに `project1.lpr` というプロジェクトが作成されます。

```
create_project project1 myDesigns
```

次の例では、`C:/Designs` の `FPGA` というディレクトリに `Proj1` というプロジェクトを作成しています。指定したディレクトリに同名のプロジェクトが既に存在する場合は、それが上書きされます。2 行目と 3 行目では、`-force` の位置を変更しても問題ないことを示しています。

```
create_project Proj1 C:/Designs/FPGA -force
-or-
create_project Proj1 -force C:/Designs/FPGA
-or-
create_project -force Proj1 C:/Designs/FPGA
```

注記: どの例でも、キーワードのない最初の引数は **<name>** 変数として、2 つ目の引数は **<dir>** 変数として認識されます。

次の例では、Manage IP フロー用のプロジェクトを指定したディレクトリに作成しています。

```
create_project -ip manageIP C:/Data
```

次の例では、`pin_project` という新規プロジェクトを作成し、`DESIGN_MODE` プロパティを I/O ピン プランニング プロジェクトに設定し、最後に I/O ピン プランニング デザインを開いています。

```
create_project pin_project C:/Designs/PinPlanning
set_property design_mode PinPlanning [current_filesset]
open_io_design -name io_1
```

関連項目

- [close_project](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [current_project](#)
- [open_io_design](#)
- [open_project](#)
- [save_project_as](#)
- [set_property](#)

create_property

オブジェクトのクラスのプロパティを作成します。

構文

```
create_property [-description <arg>] [-type <arg>] [-enum_values <args>]
               [-default_value <arg>] [-file_types <args>] [-display_text <arg>]
               [-quiet] [-verbose] <name> <class>
```

戻り値

作成されたプロパティ、エラーが発生した場合は ""

使用法

名前	説明
[-description]	プロパティの説明を指定します。
[-type]	作成するプロパティのタイプを指定します。有効な値は string、int、long、double、bool、enum、file で、デフォルトは string です。
[-enum_values]	列挙値を指定します。
[-default_value]	タイプ文字列のデフォルト値を指定します。
[-file_types]	ファイル タイプの拡張子 (ピリオドなし) を指定します。
[-display_text]	ファイル ブラウザーでファイルを選択したときに表示するテキストを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	作成するプロパティの名前を指定します。
<class>	プロパティを作成するオブジェクト タイプを指定します。有効な値は、design、net、cell、pin、port、pblock、interface、fileset です。

カテゴリ

[PropertyAndParameter \(プロパティおよびパラメーター\)](#)、XDC

説明

指定したオブジェクトのクラス (<class>) に対し、指定したタイプ (<type>) のプロパティを指定した名前 (<name>) で作成します。作成したプロパティは、`set_property` コマンドで指定したクラスのオブジェクトに割り当てることができますが、そのクラスのすべてのオブジェクトに自動的に関連付けられません。

プロパティをそのオブジェクトに割り当てないと、`report_property -all` コマンドを使用しても、指定したクラスのオブジェクトに対して新しく作成したプロパティはレポートされません。

引数

`-description <arg>` (オプション): 作成するプロパティの説明を指定します。この説明は、Vivado Design Suite のヘルプシステムでこのプロパティをクエリしたときに表示されます。

`-type <arg>` (オプション): 作成するプロパティのタイプを指定します。有効なプロパティ タイプは、次のとおりです。

- `string`: 新しいプロパティを文字列値で定義します。`-type` を指定しない場合、これがデフォルトです。
- `int`: 新しいプロパティを -2,147,483,648 ~ 2,147,483,647 の 4 バイト符号付短整数値で定義します。`int` プロパティ タイプに浮動小数点値を指定すると、エラー メッセージが返されます。
- `long`: 新しいプロパティを -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 の符号付 64 ビット整数で定義します。`long` プロパティ タイプに浮動小数点値を指定すると、エラー メッセージが返されます。
- `double`: 新しいプロパティを浮動小数点値で定義します。
- `bool`: 新しいプロパティを `true` (1 または `yes`) または `false` (0 または `no`) のブール値で定義します。
- `enum`: 新しいプロパティを列挙データ型 (`-enum_values` オプションで指定された有効な列挙値) で定義します。
- `string_list`: 新しいプロパティを文字列値の Tcl リストで指定します。
- `int_list`: 新しいプロパティを整数値の Tcl リストで指定します。
- `double_list`: 新しいプロパティを浮動小数点値の Tcl リストで指定します。

`-enum_values <args>` (オプション): プロパティに指定可能な列挙値を指定します。複数の値を `{ }` で囲み、スペースで区切って指定します。このオプションは、`-type enum` オプションを使用している場合にのみ使用可能です。

`-default_value <args>` (オプション): プロパティのデフォルト値を指定します。このオプションは、`string`、`int`、`bool`、および `enum` タイプ プロパティに使用できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 定義するプロパティの名前を指定します。大文字/小文字が区別されます。

`<class>` (必須): 新しいプロパティを割り当てるオブジェクトのクラスを指定します。指定したクラスのオブジェクトはすべて、新しく定義したプロパティに割り当てられます。有効な値は、`design`、`net`、`cell`、`pin`、`port`、`pblock`、`Pblock`、`interface`、`fileset` です。

例

次の例では、セル オブジェクトに対して `PURPOSE` というプロパティを定義します。

```
create_property PURPOSE cell
```

注記: `-type` は指定されていないので、値は文字列になります。

次の例では、整数値を指定する COUNT というピン プロパティを作成します。

```
create_property -type int COUNT pin
```

関連項目

- [get_property](#)
- [list_property](#)
- [list_property_value](#)
- [report_property](#)
- [reset_property](#)
- [set_property](#)

create_reconfig_module

新しいリコンフィギャラブル モジュールを作成します。

構文

```
create_reconfig_module -name <arg> [-top <arg>] [-gate_level]
                        -partition_def <arg> [-define_from <arg>] [-define_from_file <arg>]
                        [-quiet] [-verbose]
```

使用法

名前	説明
-name	リコンフィギャラブル モジュールの名前を指定します。
[-top]	最上位モジュールの名前を指定します。
[-gate_level]	RM が EDIF/DCP ファイルで定義されていることを示します。
-partition_def	リコンフィギャラブル モジュールを作成するパーティション定義を指定します。
[-define_from]	ブロックセットの最上位となるソース ファイルセットのモジュール名を指定します。
[-define_from_file]	リコンフィギャラブル モジュールを作成するソース ファイルセットの最上位ソース ファイルの完全パスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[Partition \(パーティション\)](#)

説明



重要: まず、プロジェクトの PR_FLOW プロパティを TRUE に設定するか、[Tools]→[Enable Partial Reconfiguration] コマンドを使用して、プロジェクトをパースシャル リコンフィギュレーション (PR) プロジェクトとして定義する必要があります。

create_reconfig_module コマンドは、指定した階層セルまたはデザイン ファイルからリコンフィギャラブル モジュール (RM) を定義し、現在のプロジェクトの指定したパーティション定義 (partitionDef) に割り当てます。

パースシャル リコンフィギュレーション フローでは、partitionDef の RM を入れ替えて、スタティック デザインと RM の組み合わせに基づく独自のデザイン コンフィギュレーションを作成できます。1 つの partitionDef には、異なるネットリスト、制約、またはインプリメンテーションを含む複数の RM を含めることができます。デザインに含まれる partitionDef の各インスタンスを異なる RM に割り当てて、さまざまなコンフィギュレーションをサポートできます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパースシャル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

このコマンドを実行すると、作成された RM の階層名が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-name <arg> (必須): 作成するリコンフィギャラブル モジュール (RM) の名前を指定します。

-top <arg> (オプション): RM の階層を定義する最上位セルを指定します。セルは名前で指定できます。

-gate_level (オプション): RM が RTL ソース ファイルではなくネットリスト ファイル (EDIF または DCP) で定義されていることを示します。

-partition_def <arg> (必須): RM を割り当てる partitionDef オブジェクトを指定します。partitionDef は、名前で指定するか、または get_partition_defs コマンドでオブジェクトとして返すことにより指定します。

-define_from <arg> (オプション): RM の定義元となる現在のプロジェクトに含まれる階層セルを指定します。指定したセルのソース ファイルにより RM のソース ファイルの内容が定義されます。セルは名前で指定できます。

-define_from_file <arg> (オプション): ゲート レベル RM のソースを定義するネットリストまたは DCP ファイルを指定します。パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリからファイルが検索されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、指定した名前のリコンフィギャラブル モジュールを作成しています。

```
create_reconfig_module -name fftBottom -partition_def \
[get_partition_defs partDef1 ] -top fftTop
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)
- [delete_reconfig_modules](#)
- [get_partition_defs](#)
- [get_reconfig_modules](#)
- [report_pr_configuration_analysis](#)
- [set_property](#)

create_report_config

設定可能なレポート オブジェクトを作成します。

構文

```
create_report_config [-report_name <arg>] [-report_type <arg>]
  -steps <args> -runs <args> [-options <arg>] [-copy_of <args>] [-quiet]
  [-verbose]
```

戻り値

設定可能なレポート オブジェクトのリスト

使用法

名前	説明
[-report_name]	設定可能なレポート オブジェクトの名前を指定します。複数のオブジェクトを作成する場合は使用できません。
[-report_type]	設定可能なレポート オブジェクトのタイプを指定します。-copy_of オプションを使用している場合は不要です。
-steps	作成するオブジェクトの run 段階を指定します。
-runs	作成するオブジェクトの run を指定します。
[-options]	設定可能なレポート オブジェクトの作成時に設定するレポート コマンドのオプションを指定します。-copy_of オプションを使用している場合は使用できません。
[-copy_of]	既存の設定可能なレポート オブジェクトを指定してコピーします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)、Report (レポート)

説明

合成 run、インプリメンテーション run、またはレポート ストラテジに追加する設定可能なレポート オブジェクトを作成します。レポート オブジェクトは、合成またはインプリメンテーション デザイン run の指定した段階が完了したときに実行するレポートのタイプとオプションを定義します。レポート ストラテジには複数のレポート オブジェクトを定義でき、合成 run およびインプリメンテーション run に関連付け、デザイン フローの異なる段階で再利用できます。詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。

レポート オブジェクトには OPTIONS.MORE_OPTIONS プロパティがあり、レポート オブジェクトに関連付ける report_* Tcl コマンドのコマンド ライン オプションを指定できます。指定したコマンド ライン オプションは、合成 run またはインプリメンテーション run でのレポート生成時に使用されます。コマンド ライン オプションは、下に説明するように -options オプションを使用して指定するか、または set_property コマンドを使用して既存のレポート オブジェクトの OPTIONS.MORE_OPTIONS プロパティを設定することにより指定できます。使用可能なコマンド ライン オプションは、該当する report_* コマンドを参照してください。

引数

`-report_name <arg>` (オプション): 生成されるレポートの名前を指定します。名前を指定しない場合、`-runs`、`-steps`、および `-report_type` オプションを組み合わせた名前が使用されます。



ヒント: `-report_name` オプションは、複数のレポート オブジェクトを作成する場合は指定できません。複数のレポート オブジェクトを作成する場合は、レポート名は自動的に付けられます。

`-report_type <arg>` (オプション): レポート オブジェクトで実行するレポート コマンドを指定します。ほとんどの `report_*` Tcl コマンドを指定できます。



ヒント: `-copy_of` オプションを使用して既存のレポート オブジェクトのコピーを作成する場合は、`-report_type` オプションは使用できません。

`-steps <args>` (必須): レポート オブジェクトに関連付ける合成またはインプリメンテーションのプロセス段階を指定します。レポートが各段階で再実行されるようにオブジェクトを複数の段階で使用するように指定できます。その場合、レポート名は自動的に付けられます。有効な値は、合成またはインプリメンテーションのすべてのプロセス 段階 (`synth_design`、`opt_design`、`place_design`、`route_design` など) です。

`-runs <args>` (必須): レポート オブジェクトに関連付ける合成またはインプリメンテーションのデザイン run を指定します。オブジェクトを複数のデザイン run で使用するよう指定できます。その場合、レポート名は自動的に付けられます。

`-options <arg>` (オプション): 指定した `report_*` コマンドを実行するときに使用するコマンド ライン オプションを指定します。使用可能なオプションは、該当するレポート コマンドを参照してください。`-copy_of` オプションを使用して既存のレポート オブジェクトのコピーを作成する場合は、このオプションは使用できません。



重要: `-options` オプションで指定したコマンド ライン オプションが指定したレポート タイプに正しいものであることはチェックされません。無効なオプションを指定すると、レポートが実行されたときにエラーが返されます。

`-copy_of <arg>` (オプション): 新しいレポート オブジェクトのテンプレートとして使用するレポート オブジェクトを指定します。新しいレポート オブジェクトは、新しい `-steps` および `-runs` オプションに関連付けることができます。



ヒント: `-copy_of` オプションを使用する場合、`-report_type` および `-options` オプションは使用できません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定の名前およびタイプの新しいレポート オブジェクトを作成し、impl_1 run の route_design 段階に関連付けています。

```
create_report_config -report_name post_route_datasheet -report_type
report_datasheet \
-steps route_design -runs impl_1
```

次の例は先ほどの例と同じですが、名前は指定していないので、レポート オブジェクトの名前は自動的に付けられ、最後の行に示す名前になります。

```
create_report_config -report_type report_datasheet -steps route_design -
runs impl_1
impl_1_route_report_datasheet_0
```

次の例では、指定したコマンド ライン オプションを使用するタイミング サマリ レポートの新しいレポート オブジェクトを作成し、インプリメンテーション run の複数の段階に関連付けています。

```
create_report_config -report_type report_timing_summary \
-steps {opt_design place_design route_design} -runs {impl_2} \
-options {-no_detailed_paths -report_unconstrained}
```

関連項目

- [delete_report_configs](#)
- [generate_reports](#)
- [get_report_configs](#)
- [set_property](#)

create_rqs_run

report_qor_suggestions コマンドで生成された QoR 推奨項目に基づき、新しいインプリメンテーション run を作成して実行します。このプロシージャは、ユーザーが指定したディレクトリの 5 つのファイル (RQSPreSynth_<newProjName>.xdc、RQSImpCommon_<newProjName>.xdc、RQSPreImpl_<newProjName>.xdc、RQSPreImpl_<newProjName>.tcl、RQSImpCommon_<newProjName>.tcl) を検索します。2 つのフローがあります。1 つ目のフローでは合成 run とインプリメンテーション run の両方が作成され、2 つ目のフローではインプリメンテーション run のみが作成され、ユーザーが合成 run を指定できます。1 つ目のフローでは、現在のインプリメンテーション run の親 run (現在の合成 run) に基づいて新しい合成 run が作成されます。新しい制約ファイルセットが作成され、それに現在の合成 run の制約ファイルセットが追加されます。そして、RQSPreSynth_<>.xdc ファイルが新しく作成された制約セットに追加されます。インプリメンテーション run は、現在のインプリメンテーション run に基づいて作成されます。インプリメンテーション run 制約ファイルセットは、既に作成されたものと異なる場合に作成され、それに現在のインプリメンテーション run の制約ファイルセットのファイルが追加されます。この新しく作成されたインプリメンテーション制約ファイルセットには、RQSImpCommon_<>.xdc も追加されます。新しく作成されたインプリメンテーション run の STEPS.OPT_DESIGN.TCL.PRE プロパティは、RQSImpCommon_<>.tcl ファイルに設定されます。このフローでは、RQSPreImpl_<>.xdc/tcl ファイルは無視されます。2 つ目のフローでは、新しく作成されたインプリメンテーション run の親合成 run はユーザーが指定するので、合成 run は作成されません。インプリメンテーション run は、現在のインプリメンテーション run とユーザーが指定した合成 run に基づいて作成されます。新しいインプリメンテーション制約ファイルセットが作成され、それに現在のインプリメンテーション run の制約ファイルセットのファイルが追加されます。RQSImpCommon_<>.xdc および RQSPreImpl_<>.xdc ファイルも追加されます。RQSPreImpl_<>.tcl ファイルがある場合はこれが新しいインプリメンテーション run の STEPS.OPT_DESIGN.TCL.PRE プロパティとして設定され、RQSPreImpl_<>.tcl ファイルがない場合は RQSImpCommon_<>.tcl が設定されます。どちらのフローでも、ファイルの追加および設定は、それらのファイルが出力ディレクトリにあるかどうかによります。

構文

```
create_rqs_run -dir <arg> -new_name <arg> [-synth_name <arg>]
               [-opt_more_options <arg>] [-place_more_options <arg>] [-quiet]
               [-verbose]
```

戻り値

なし

使用法

名前	説明
-dir	XDC ファイルおよび Tcl ファイルを含むディレクトリを指定します。
-new_name	新しい run の名前を入力します。
[-synth_name]	既存の合成 run の名前を指定します。これが新しく作成されるインプリメンテーション run の親 run になります。デフォルトはありません。
[-opt_more_options]	新しく作成する run に設定する opt_design の追加のオプションと値を指定します。デフォルトはありません。
[-place_more_options]	新しく作成する run に設定する place_design の追加のオプションと値を指定します。デフォルトはありません。
[-quiet]	コマンド エラーを表示しません。

名前	説明
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[xilinxclstore](#) (ザイリンクス Tcl Store)、[projutils](#) (プロジェクト ユーティリティ)

説明

`report_qor_suggestions` による QoR 推奨項目に基づいて新しいインプリメンテーション run を作成して実行します。

引数

`-dir <arg>` (必須): 必須のファイルが含まれるディレクトリを指定します。

`-new_name <arg>` (必須): 新しい run の名前を指定します。

`-synth_name <arg>` (オプション): 既存の合成 run を指定します。この run が新しく作成されるインプリメンテーション run の親 run として設定されます。



ヒント: 既存の合成 run が複数存在する場合は、このオプションを指定する必要があります。

`-opt_more_options <arg>` (オプション): `opt_design` コマンドの追加のコマンドライン オプションを指定します。指定したオプションは作成するインプリメンテーション run の `MORE_OPTIONS` プロパティとして設定され、`opt_design` の実行時にオプションとして追加されます。

`-place_more_options <arg>` (オプション): `place_design` コマンドの追加のコマンドライン オプションを指定します。指定したオプションは作成するインプリメンテーション run の `MORE_OPTIONS` プロパティとして設定され、`place_design` の実行時にオプションとして追加されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したディレクトリからの制約ファイルを使用して `exp_1` という新しい run を作成して実行しています。

```
create_rqs_run -dir path_to_dir -new_name exp_1 -synth_name synth_1 \
-opt_more_options optVal -place_more_options placeVal
```



ヒント: 制約は、`report_qor_suggestions` の `-output_dir` オプションを使用して作成されたものです。

関連項目

- [current_run](#)
- [get_runs](#)
- [report_qor_suggestions](#)

create_run

現在のプロジェクトの合成またはインプリメンテーション run を定義します。

構文

```
create_run [-constrset <arg>] [-parent_run <arg>] [-part <arg>] -flow <arg>
          [-strategy <arg>] [-report_strategy <arg>] [-pr_config <arg>] [-quiet]
          [-verbose] <name>
```

戻り値

run オブジェクト

使用法

名前	説明
[-constrset]	使用する制約ファイルセットを指定します。
[-parent_run]	新しいインプリメンテーション run に関連付ける合成 run を指定します。
[-part]	ターゲット パーツを指定します。
-flow	フロー名を指定します。
[-strategy]	run に適用するストラテジを指定します。
[-report_strategy]	run に適用するレポート ストラテジを指定します。
[-pr_config]	run に適用するパーティション コンフィギュレーションを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	新規 run の名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

合成またはインプリメンテーション run を定義します。run の属性は、set_property コマンドを使用して設定できます。

引数

-constrset <arg> (オプション): 合成またはインプリメンテーション run で使用する制約セットを指定します。

-parent_run <arg>: 現在の run のネットリストを定義する run を指定します。ネットリスト ベースのプロジェクトの場合は、-parent_run オプションは必要ありません。RTL ソース プロジェクトの場合、インプリメンテーション run に -parent_run を指定する必要があります。合成 run には必要ありません。パースナル リコンフィギュレーション フローでは、-parent_run オプションで合成 run、インプリメンテーション run、または -pr_config オプションを指定している場合は PR コンフィギュレーションを指定できます。

`-part <partName>` (オプション): `run` に使用するサイリンクス パーツを指定します。`-part` オプションを指定しない場合は、プロジェクトに定義されているデフォルト パーツが使用されます。

`-flow <arg>` (必須): 合成ツール (例: {Vivado Synthesis 2017}) またはインプリメンテーション ツール (例: {Vivado Implementation 2017}) のツール フローおよびリリース バージョンを指定します。

`-strategy <arg>` (オプション): 合成またはインプリメンテーション `run` で使用するストラテジを指定します。ツールには、ユーザー定義のカスタム ストラテジも含め、多くのストラテジが含まれています。使用可能な合成およびインプリメンテーション ストラテジについては、該当するユーザー ガイドを参照してください。ストラテジを指定しない場合、[Synthesis Defaults] または [Implementation Defaults] が使用されます。

`-report_strategy <arg>` (オプション): デザイン フローの異なる段階で実行するレポート オブジェクトのコレクションを定義するレポート ストラテジを指定します。レポート オブジェクトは、`create_report_config` コマンドで作成します。レポート ストラテジは、Vivado IDE の [Settings] ダイアログ ボックスで定義します。

`-pr_config <arg>` (オプション): 新しい `run` に適用するパーティション コンフィギュレーションを指定します。パーティション コンフィギュレーションは、`create_pr_configuration` コマンドで指定します。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 設定する合成 `run` またはインプリメンテーション `run` の名前を指定します。

例

次の例では、Vivado 合成ツール フローを使用する `synth_1` という名前の `run` を作成しています。

```
create_run -flow {Vivado Synthesis 2013} synth_1
```

注記: この場合、合成 `run` でデフォルトの `sources_1`、`constrs_1`、プロジェクトのデフォルト パーツが使用されます。また、これは合成 `run` なので、`-parent_run` オプションは必要ありません。

次の例では、Vivado インプリメンテーション 2013 ツール フローを使用するインプリメンテーション `run` を作成し、先ほどの例で作成した `synth_1` という合成 `run` に関連付けています。

```
create_run impl_2 -parent_run synth_1 -flow {Vivado Implementation 2013}
```

注記: この例では、合成済み RTL ソースをインプリメントしているので、`-parent_run` オプションを使用する必要があります。

関連項目

- [create_pr_configuration](#)
- [create_report_config](#)
- [current_run](#)
- [launch_runs](#)

- [set_property](#)

create_slack_histogram

スラック ヒストグラムを作成します。

構文

```
create_slack_histogram [-to <args>] [-delay_type <arg>] [-num_bins <arg>]
                        [-slack_less_than <arg>] [-slack_greater_than <arg>] [-group <args>]
                        [-report_unconstrained] [-significant_digits <arg>] [-scale <arg>]
                        [-name <arg>] [-cells <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-to]	範囲の終わりのクロックを指定します。
[-delay_type]	パス遅延のタイプを指定します。有効な値は max、min、min_max で、デフォルトは max です。
[-num_bins]	ビンの最大数を指定します。有効な値は 1 ～ 100 で、デフォルト値は 10 です。
[-slack_less_than]	この値よりも小さいスラックのパスを表示します。デフォルトは 1e+30 です。
[-slack_greater_than]	この値よりも大きいスラックのパスを表示します。デフォルトは -1e+30 です。
[-group]	指定のグループのパスのみをレポートします。
[-report_unconstrained]	制約の設定されていない終点をレポートします。
[-significant_digits]	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 3 です。
[-scale]	ヒストグラムを描画する尺度のタイプを指定します。有効な値は linear または logarithmic で、デフォルトは linear です。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-cells]	create_slack_histogram コマンドを指定のセルに対して実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

パスをスラック値ごとにグループ化し、グラフィカルに表示するスラック ヒストグラムを作成します。



ヒント: グラフィカル スラック ヒストグラムを表示するには、ツールを GUI モードで実行する必要があります。

引数

`-to <arg>` (オプション): クロック名を指定し、そのクロック ドメインで終了するパスを解析します。

`-delay_type <arg>` (オプション): スラック レポートを作成する際に解析するパス遅延のタイプを指定します。有効な値は `min`、`max`、および `min_max` です。`-delay_type` オプションのデフォルト値は `max` です。

`-num_bins <arg>` (オプション): 結果を分割するスラック ビンの数を指定します。ビンの数により、ヒストグラムの粒度が決まります。算出されたスラック値の範囲が指定したビン数に均等に分割され、スラック値に応じてパスがグループ化されます。1 ~ 100 の値を指定でき、デフォルト値は 10 です。

`-slack_less_than <arg>` (オプション): 算出されたスラック値が指定した値より小さいパスのスラックをレポートします。`-slack_greater_than` と共に使用すると、スラック値の範囲を指定できます。

`-slack_greater_than <arg>` (オプション): 算出されたスラック値が指定した値より大きいパスのスラックをレポートします。`-slack_less_than` と共に使用すると、スラック値の範囲を指定できます。

`-group <args>` (オプション): 指定したパス グループのパスのスラックをレポートします。現在定義されているパスグループを確認するには、`get_path_groups` コマンドを使用します。

`-report_unconstrained` (オプション): 制約が適用されていないパスの遅延スラックをレポートします。デフォルトでは、制約が適用されていないパスは解析されません。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 3 です。

`-scale [linear | logarithmic]` (オプション): スラック ヒストグラムで使用する Y 軸の尺度を指定します。`logarithmic` に設定すると、大きく異なる値の表示がスムーズになります。デフォルトは `linear` です。

`-name <arg>` (オプション): GUI で表示する場合の結果の名前を指定します。指定した名前の結果が既にある場合は、`create_slack_histogram` コマンドによりその結果が上書きされます。

`-cells <arg>` (オプション): 指定した階層セルのレポートを生成します。レポートの詳細は、デザイン全体 (`current_instance`) ではなく、指定したセルに基づきます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインのスラック ヒストグラムをデフォルト設定で作成し、結果を指定した名前で GUI に表示しています。

```
create_slack_histogram -name slack1
```

関連項目

- [delete_timing_results](#)

- [get_path_groups](#)
- [report_timing](#)

create_sysgen

ザイリンクス System Generator 用の DSP ソースを作成し、ソース ファイルセットに追加します。

構文

```
create_sysgen [-quiet] [-verbose] <name>
```

戻り値

新しいサブモジュールの名前

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	サブモジュールの名前を指定します。

カテゴリ

[SysGen \(System Generator\)](#)

説明

現在のプロジェクトで使用する DSP サブモジュールを作成し、ソース ファイルとして追加します。

このコマンドを実行すると、System Generator for DSP が起動し、システム デザインのハードウェア部分を設計できます。System Generator はザイリンクスの DSP デザイン ツールで、RTL ソース ファイル、Simulink® および MATLAB® ソフトウェア モデル、および DSP システムの C/C++ コンポーネントを 1 つのシミュレーションおよびインプリメンテーション環境にまとめることができます。

ツールの機能の詳細は、『System Generator for DSP 入門ガイド』 (UG639) を参照してください。

System Generator からの既存の DSP モジュール ファイル (.mdl) を `add_files` コマンドを使用して現在のプロジェクトに追加することもできます。

このコマンドを実行すると、作成されプロジェクトに追加された DSP モジュールの名前が返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 作成し、プロジェクトに追加する DSP モジュールの名前を指定します。

例

次の例では、System Generator を起動し、指定の DSP モジュールを定義および設定しています。

```
create_sysgen DSP_mod1
```

関連項目

- [add_files](#)
- [generate_target](#)
- [list_targets](#)
- [make_wrapper](#)

create_waiver

DRC、METHODOLOGY、CDC メッセージ除外を作成します。

構文

```
create_waiver [-type <arg>] [-id <arg>] [-objects <args>] [-from <args>]
               [-to <args>] [-strings <args>] [-of_objects <args>] [-user <arg>]
               -description <arg> [-tags <arg>] [-timestamp <arg>] [-scoped] [-quiet]
               [-verbose]
```

戻り値

除外オブジェクト

使用法

名前	説明
[-type]	除外のタイプを指定します。有効な値は、DRC、METHODOLOGY、CDC です。
[-id]	除外する DRC、METHODOLOGY、CDC メッセージの ID を指定します。-of_objects オプションを使用している場合は不要です。
[-objects]	DRC、METHODOLOGY 除外を設定するメッセージ オブジェクトを 1 つまたは複数指定します。オブジェクトは、セル、ネット、サイトなどの違反定義 (%ELG、%SIG など)、またはワイルドカード (*CELL、*NET、*SITE) として指定できます。
[-from]	CDC 除外を設定するソース (ドライバー) ピンまたはポート (あるいはワイルドカードとして *PORT、*PIN) を指定します。
[-to]	CDC 除外を設定するターゲット (ロード) ピンまたはポート (あるいはワイルドカードとして *PORT、*PIN) を指定します。
[-strings]	DRC、METHODOLOGY、CDC 除外を設定するメッセージ文字列値 (文字列の %STR またはワイルドカード (*)) を指定します。
[-of_objects]	除外を設定する DRC、METHODOLOGY、CDC オブジェクトを指定します。
[-user]	除外を作成するユーザーの名前を指定します。指定しない場合は、環境からの USER 名が使用されます。
-description	除外の理由を説明する記述を指定します。
[-tags]	除外の特定または分類に役立つ 1 つまたは複数のタグを指定します。
[-timestamp]	除外が作成された時間を保持するため、除外のタイムスタンプを指定します。
[-scoped]	オブジェクトのワイルドカードを設定されている現在のインスタンス (current_instance) に限定して解釈するように指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Waiver (除外)、XDC、DRC、Methodology (設計手法)、Object (オブジェクト)

説明

`report_drc`、`report_methodology`、または `report_cdc` コマンドを実行すると、デザインで検出された違反または状況に関するメッセージが返されます。これらの違反は、解決しないと先に進めないことがあります。
`create_waiver` コマンドを使用すると、デザインで無視しても問題ない違反または状況を除外オブジェクトとして指定し、デザイン フローを先に進めることができます。



重要: 違反を除外設定する場合には、注意が必要です。除外設定をすると、デザイン フローを先に進めることはできませんが、構造上欠陥のあるデザインが作成されてしまいます。

除外の履歴を記録するため、`create_waiver` コマンドを実行するときにユーザー ID と説明を指定する必要があります。

除外は、個々の DRC または設計手法違反、特定の DRC または設計手法チェック、または CDC パスに指定する必要があります。特定のオブジェクト、特定の違反 ID、または `-from/-to` オプションを使用してパスを指定します。
`create_waiver` コマンドの形式は、下の例に示すように、除外するチェック、違反、またはオブジェクトによって異なります。



ヒント: 多くのコマンド オプションは必須ではありませんが、除外に関連付けるために識別するものがが必要です。

除外を次のデザイン セッションでも使用できるように保存するには、`write_waivers` コマンドを使用して除外コマンドの XDC ファイルを作成し、デザインを開き直したときに `read_xdc` コマンドを使用して読み込みます。

除外を作成したら、DRC、設計手法、または CDC レポートを再実行し、解析で除外が考慮されるようにする必要があります。現在のデザインに設定されている除外を確認するには、`report_waivers` コマンドを使用します。また、`report_drc`、`report_methodology`、および `report_cdc` コマンドには、除外された違反またはチェックに対してレポートを実行するオプションがあります。デザインで指定した除外オブジェクトを削除するには、`delete_waivers` コマンドを使用します。

引数

`-type <arg>` (オプション): 作成する除外をタイプを指定します。有効な値は DRC、METHODOLOGY、および CDC です。

`-id <arg>` (オプション): 除外されているチェックまたは違反の ID を指定します。`-of_objects` オプションを使用している場合は不要です。ID に関連付けられている除外は、`-objects`、`-from/-to`、または `-strings` オプションを使用してさらに絞り込むことができます。

`-objects <arg>` (オプション): DRC または設計手法のチェックおよび違反に対して、除外を適用するオブジェクトを 1 つまたは複数指定します。オブジェクトは、セル、ネット、サイトなどの違反定義 (%ELG、%SIG など)、またはワイルドカード (*CELL、*NET、*SITE) として指定できます。詳細は、`create_drc_check` または `create_drc_violation` コマンドを参照してください。

`-from <arg>` (オプション): CDC チェックまたは違反に対して、除外を適用するソース (ドライバー) ピンまたはポート (あるいはワイルドカードとして *PORT、*PIN) を指定します。

`-to <arg>` (オプション): CDC チェックまたは違反に対して、除外を適用するターゲット (ロード) ピンまたはポート (あるいはワイルドカードとして *PORT、*PIN) を指定します。

`-strings` (オプション): DRC または設計手法のチェックおよび違反に対して、除外を適用するメッセージ文字列値 (文字列の %STR またはワイルドカード (*)) を指定します。

`-of_objects <arg>` (オプション): DRC 違反を `get_drc_violations` コマンドを使用して、設計手法違反を `get_methodology_violations` コマンドを使用して、または CDC 違反を `get_cdc_violations` コマンドを使用してオブジェクトとして指定します。除外は、指定した違反オブジェクトに適用されます。

`-user <arg>` (オプション): 除外を作成するユーザーの ID を指定します。指定しない場合、ユーザー ID は OS から取得されます。

`-description <arg>` (必須): 除外の簡単な説明、適用する理由を記述します。

`-tags <arg>` (オプション): 除外に関連付けるユーザー定義文字列のリストを指定します。これらのタグは、`get_waivers` コマンドを使用した場合に除外をフィルターするのに使用できます。

`-timestamp <arg>` (オプション): 除外を作成した時間を指定します。「SUN Aug 6 17:02:21 GMT 2017」という形式で指定する必要があります。`-timestamp` オプションを指定しない場合、現在のタイマーが使用されます。



ヒント: タイムスタンプを現地時間で指定するには、`clock format [clock seconds] -gmt 0` を使用します。

`-scoped` (オプション): オブジェクトのワイルドカードを設定されている現在のインスタンス (`current_instance`) に限定して解釈するよう指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ID で指定した設計手法チェックに除外を作成しています。

```
create_waiver -id TIMING-18 -user samwise -description {waive rule check}
```

次の例では、指定したポート オブジェクトの指定した ID に DRC チェック除外を作成し、タイムスタンプを現時時間で指定しています。

```
create_waiver -type DRC -id UCIO-1 -user samwise -desc {waiving DRC
violation} \
-objects [get_ports {src_in* dest_out*}] \
-timestamp [clock format [clock seconds] -gmt 0]
```

次の例では、`-from/-to` オプションで指定した CDC パスに除外を作成しています。

```
create_waiver -type CDC -id CDC-6 -user samwise \
-description {Paths to be re-tested later}\
-from [list [get_pins {inst_xpm_grey/src_gray_ff_reg[3]/C} \
inst_xpm_grey/src_gray_ff_reg[16]/C \
inst_xpm_grey/src_gray_ff_reg[22]/C] \
```

```
inst_xpm_grey/src_gray_ff_reg[25]/C}} ] \
-to [list [get_pins {inst_xpm_grey/dest_graysync_ff_reg[0][1]/D } \
inst_xpm_grey/dest_graysync_ff_reg[0][6]/D } \
inst_xpm_grey/dest_graysync_ff_reg[0][9]/D } \
inst_xpm_grey/dest_graysync_ff_reg[0][24]/D}} ]
```

関連項目

- [current_instance](#)
- [delete_waivers](#)
- [get_cdc_violations](#)
- [get_drc_violations](#)
- [get_methodology_violations](#)
- [get_waivers](#)
- [report_cdc](#)
- [report_drc](#)
- [report_methodology](#)
- [report_waivers](#)
- [write_waivers](#)

create_wave_config

新しい波形設定を作成します。

構文

```
create_wave_config [-quiet] [-verbose] [<name>]
```

戻り値

新しい波形設定

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	作成する波形設定の名前を指定します。指定しない場合は、デフォルト名で作成されます。波形設定を表示する新しい波形ウィンドウが作成され、現在の波形ウィンドウとなります。

カテゴリ

[Waveform \(波形\)](#)

説明

現在のシミュレーションで新規波形設定オブジェクトを作成し、Vivado IDE で開きます。これにより、新しい波形設定オブジェクトが現在の波形設定になります。

Vivado® シミュレータ GUI では、波形ウィンドウを使用してデザインを解析し、コードをデバッグできます。波形設定ファイルには、表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。波形設定には最上位 HDL オブジェクトが表示されますが、`add_wave` および `add_wave_divider` などのコマンドを使用してオブジェクトを追加できます。波形設定に加えた変更は、`save_wave_config` コマンドを使用して保存できます。

このコマンドを実行すると、作成された波形設定の名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (オプション): 波形設定オブジェクトの名前を指定します。<name> を指定しない場合、「Untitled #」(# は 1 から開始する番号) という名前の波形設定が作成されます。

例

次の例では、指定の名前の波形設定オブジェクトを作成しています。

```
create_wave_config testbench1
```

関連項目

- [close_wave_config](#)
- [current_wave_config](#)
- [get_wave_configs](#)
- [open_wave_database](#)
- [open_wave_config](#)
- [save_wave_config](#)

create_xps

XPS 用のエンベデッド ソースを作成し、ソース ファイルセットに追加します。このフローはサポートされません。Vivado IP インテグレーターを使用してください。

構文

```
create_xps [-quiet] [-verbose] <name>
```

戻り値

作成されたソース ファイルの名前

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	ソース ファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のプロジェクトで使用するエンベデッド プロセッサ ソースを作成し、ソース ファイルとして追加します。

このコマンドを実行すると、Xilinx Platform Studio (XPS) が起動し、エンベデッド プロセッサ システムのハードウェア部分を設計できます。XPS では、マイクロプロセッサ、ペリフェラル、およびそれらのコンポーネントのインターコネクトを定義および設定できます。XPS を閉じると、作成されたエンベデッド プロセッサ サブデザインのファイルがローカルのプロジェクト ディレクトリ (`<project_name>.srcs/sources_1/edk/<name>`) に保存され、ソース ファイルとして追加されます。

XPS の機能の詳細は、『EDK コンセプト、ツール、テクニック』 (UG683) を参照してください。

XPS からの既存のザイリンクス マイクロプロセッサ プロジェクト (.xmp) ファイルを `add_files` コマンドを使用して現在のプロジェクトに追加することもできます。

このコマンドを実行すると、作成されたエンベデッド プロセッサ サブデザインの名前が返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 作成し、プロジェクトに追加するエンベデッド プロセッサ サブデザインの名前を指定します。

例

次の例では、XPS を起動し、指定のエンベデッド プロセッサ サブデザインを定義および設定しています。

```
create_xps xpsTest1
```

関連項目

- [add_files](#)
- [generate_target](#)
- [list_targets](#)
- [make_wrapper](#)

current_bd_design

現在のデザインを設定または取得します。

構文

```
current_bd_design [-quiet] [-verbose] [<design>]
```

戻り値

現在のデザイン オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<design>]	設定する現在のデザインの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado Design Suite の IP インテグレーターで使用する現在の IP サブシステム デザインを定義するか、アクティブ プロジェクトの現在のデザインを返します。

IP インテグレーターで実行するデほとんどの Tcl コマンドおよびデザインの変更では、現在の IP サブシステム デザインおよび現在の IP サブシステム インスタンスがターゲットとなります。現在の IP サブシステム インスタンスを指定するには、`current_bd_instance` コマンドを使用します。

`get_bd_designs` コマンドを使用すると、アクティブ プロジェクトで開いている IP サブシステム デザインのリストが返されます。

IP インテグレーターのすべての Tcl コマンドをリストするには、Vivado Design Suite Tcl シェルで次のコマンドを実行します。

```
load_features IPIntegrator
help -category IPIntegrator
```

注記: `load_features` コマンドは、IP インテグレーターが現在 Vivado Design Suite に読み込まれていない場合にのみ必要です。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<design>` (オプション): IP インテグレーターで現在のデザインとして設定する IP サブシステム デザインの名前を指定します。`<design>` を指定しない場合、アクティブ プロジェクトの現在の IP サブシステム デザインが返されます。

例

次の例では、IP サブシステム デザインを現在のデザインに設定しています。

```
current_bd_design design_1
```

関連項目

- [current_bd_instance](#)
- [get_bd_designs](#)
- [open_bd_design](#)

current_bd_instance

現在のセル インスタンスを設定または取得します。

構文

```
current_bd_instance [-quiet] [-verbose] [<instance>]
```

戻り値

現在のセル インスタンス オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<instance>]</code>	設定する現在のセル インスタンスの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

`current_bd_design` コマンドで定義された IP インテグレーター サブシステム デザインの、現在の階層セル インスタンスを設定または取得します。現在のインスタンスは、サブシステム デザイン階層の最上位を基準としています。

このコマンドを実行すると、現在の IP インテグレーター セル インスタンス オブジェクトが返されるか、正常に実行されなかった場合は何も返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<instance>` (オプション): サブシステム デザインの現在のインスタンスとして設定する IP インテグレーター階層セルの名前を指定します。

- `<instance>` は、現在のインスタンスとして定義されているインスタンスを基準に、階層区切り文字を使用して指定します。

- 現在のインスタンスに対して 1 つ上の階層に移動するには、「..」を使用します。
- <instance> を指定しない場合、現在のインスタンスがサブシステム デザイン階層の最上位モジュールにリセットされます。
- <instance> に「.」を指定した場合、現在のインスタンスの名前が返され、インスタンスは変更されません。

例

次の例では、サブシステム デザインの現在のインスタンスを指定のモジュールに設定しています。

```
current_bd_instance module2
```

次の例では、現在のインスタンスが返されます。

```
current_bd_instance .
```

次の例では、サブシステム デザインの現在のインスタンスを階層の最上位にリセットしています。

```
current_instance /
```

関連項目

- [current_bd_design](#)

current_board

現在のボード オブジェクトを取得します。

構文

```
current_board [-quiet] [-verbose]
```

戻り値

現在のボード オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のプロジェクトで使用されているボードを返します。

Vivado Design Suite インストール エリアの `data/boards` フォルダにあるボード ファイル `board.xml` には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボードは、次の方法で指定できます。

- プロジェクトを作成するときに New Project ウィザードの [Default Part] ページでボードを選択。
- 例に示すように現在のプロジェクトの BOARD_PART プロパティを設定。
- Vivado IDE の [Settings] ダイアログ ボックスの [General] ページで [Project device] にボードを選択。

プロジェクトの作成方法およびプロジェクト設定の詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。



重要: `set_property` コマンドでボードを指定すると、ターゲット デバイスも BOARD プロパティに合わせて変更されます。

`current_board` コマンドを実行すると、現在のボードの BOARD_PART プロパティとして設定されている `Vendor:Board_Name:File_Version` 属性が返されます。プロジェクトで TRD とボードではなくザイリンクス FPGA がターゲットに指定されている場合、BOARD_PART プロパティが定義されていない場合は、何も返されません。正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のプロジェクトの `BOARD` プロパティを設定し、プロジェクトで使用されるボードをレポートしています。

```
set_property BOARD_PART xilinx.com:kc705:1.0 [current_project]
current_board
xilinx.com:kintex7:kc705:1.0
```

次の例では、`BOARD_PART` プロパティを設定した結果、ターゲット デバイスが変更されたことを示しています。ターゲット デバイス (`PART` プロパティ) は自動的に変更され、警告メッセージが表示されます。

```
set_property BOARD_PART xilinx.com:ac701:1.0 [current_project]
WARNING: [Project 1-153] The current project part 'xc7k325tffg900-2'
does
not match with the 'XILINX.COM:AC701:1.0' board part settings. The
project
part will be reset to 'XILINX.COM:AC701:1.0' board part.
```

注記: `report_property` コマンドを使用すると、`current_project` の `BOARD_PART` および `PART` プロパティを確認できます。

関連項目

- [current_board_part](#)
- [current_project](#)
- [get_board_components](#)
- [get_board_parts](#)
- [get_boards](#)
- [report_property](#)
- [set_property](#)

current_board_part

現在のボード パーツ (board_part) オブジェクトを取得します。

構文

```
current_board_part [-quiet] [-verbose]
```

戻り値

現在の board_part オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のプロジェクトまたはデザインで使用されるサイリンクス デバイスを返します。

Vivado Design Suite インストール エリアの `data/boards` フォルダにあるボード ファイル `board.xml` には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード パーツは、ボード レベル システム内でのサイリンクス デバイスを表しており、ボード インターフェイス ファイルの `part0` コンポーネントで表されます。

ボードは、次の方法で指定できます。

- プロジェクトを作成するときに New Project ウィザードの [Default Part] ページでボードを選択。
- 例に示すように現在のプロジェクトの BOARD_PART プロパティを設定。
- Vivado IDE の [Settings] ダイアログ ボックスの [General] ページで [Project device] にボードを選択。

プロジェクトの作成方法およびプロジェクト設定の詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。



重要: `set_property` コマンドでボード パーツを指定すると、ターゲット パーツも BOARD_PART プロパティに合わせて変更されます。

`current_board_part` コマンドを実行すると、現在のボード パーツの Name プロパティが返されます。プロジェクトでボードではなくサイリンクス FPGA がターゲットに指定されている場合、`BOARD_PART` プロパティが定義されていない場合は、何も返されません。正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のプロジェクトの `BOARD_PART` プロパティを設定し、プロジェクトで使用されるボード パーツをレポートしています。

```
set_property BOARD_PART xilinx.com:kc705:part0:1.0 [current_project]
current_board_part
xilinx.com:kc705:part0:1.0
```

次の例では、`BOARD_PART` プロパティを設定した結果、ターゲット パーツが変更されたことを示しています。ターゲット デバイスは自動的に変更され、警告メッセージが表示されます。

```
set_property BOARD_PART xilinx.com:ac701:part0:1.0 [current_project]
WARNING: [Project 1-153] The current project part 'xc7k325tffg900-2'
does not match with the 'XILINX.COM:AC701:PART0:1.0' board part
settings. The project part will be reset to
'XILINX.COM:AC701:PART0:1.0'
board part.
INFO: [Project 1-152] Project part set to artix7 (xc7a200tfbg676-2)
```

注記: `report_property` コマンドを使用すると、`current_project` の `BOARD_PART` および `PART` プロパティを確認できます。

次の例では、現在のボード パーツの `PART_NAME` プロパティで定義されたパーツの `DEVICE`、`PACKAGE`、`SPEED`、および `FAMILY` プロパティを取得しています。

```
get_property DEVICE [get_parts [get_property PART_NAME \
[current_board_part]]]
get_property PACKAGE [get_parts [get_property PART_NAME \
[current_board_part]]]
get_property SPEED [get_parts [get_property PART_NAME \
[current_board_part]]]
get_property FAMILY [get_parts [get_property PART_NAME \
[current_board_part]]]
```

関連項目

- [current_project](#)
- [get_board_components](#)

- [get_boards](#)
- [get_board_part_interfaces](#)
- [get_board_part_pins](#)
- [get_board_parts](#)
- [report_property](#)
- [set_property](#)

current_design

現在のデザインを設定または取得します。

構文

```
current_design [-quiet] [-verbose] [<design>]
```

戻り値

デザイン オブジェクト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<design>]	設定する現在のデザインの名前を指定します。

カテゴリ

[SDC](#)、[XDC](#)

説明

現在のデザインを定義するか、アクティブ プロジェクトの現在のデザインの名前を返します。

ほとんどの Tcl コマンド、ツールで実行したデザインのおよび制約の変更では、現在のデザインおよび現在のインスタンスがターゲットとなります。現在のインスタンスを指定するには、`current_instance` コマンドを使用します。

`get_designs` コマンドを使用するとアクティブ プロジェクトで開いているデザインのリストを取得でき、`get_projects` コマンドを使用すると開いているプロジェクトのリストを取得できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<design>` (オプション): 現在のデザインとして設定するデザインの名前を指定します。`<design>` を指定しない場合、アクティブ プロジェクトの現在のデザインが返されます。

例

次の例では、デザイン rtl_1 を現在のデザインに設定しています。

```
current_design rtl_1
```

関連項目

- [current_instance](#)
- [get_designs](#)
- [get_projects](#)

current_fileset

現在のファイルセットを取得するか(任意のタイプ)、現在のファイルセットを設定します (シミュレーション ファイルセットのみ)。

構文

```
current_fileset [-constrset] [-simset] [-quiet] [-verbose] [<fileset>...]
```

戻り値

現在のファイルセット (デフォルトでは現在のソース ファイルセット)

使用法

名前	説明
[-constrset]	現在の制約ファイルセットを取得します。
[-simset]	現在アクティブなシミュレーション ファイルセットを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<fileset>]	現在の (アクティブな) シミュレーション ファイルセットを指定します (オプション)。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のプロジェクトでアクティブなソース、制約、またはシミュレーション ファイルセットの名前を取得します。

オプションを指定せずに使用すると、sources_1 がアクティブ ファイルセットとして指定され、返されます。

このコマンドは、現在のシミュレーション ファイルセットを設定するのに也可以使用できます。

注記: 合成 run またはインプリメンテーション run でのアクティブ制約セットを定義するには、set_property CONSTRSET を使用します。

引数

-constrset (オプション): 現在アクティブな制約セットを返します。

-simset (オプション): 現在アクティブなシミュレーション ファイルセットを設定または返します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<fileset>` (オプション): アクティブにするシミュレーション ファイルセットの名前を指定します。この引数は、複数のファイルセットを含むプロジェクトでアクティブなシミュレーション ファイルセットを設定します。`<fileset>` を指定しない場合、`sources_1` ファイルセットがアクティブなファイルセットとして返されます。

例

次の例では、現在アクティブな制約ファイルセットの名前が返されます。

```
current_fileset -constrset
```

次の例では、`sim_2` をアクティブなシミュレーション セットとして設定しています。

```
current_fileset -simset sim_2
```

関連項目

- [create_fileset](#)
- [delete_fileset](#)
- [get_filesets](#)

current_frame

HDL プロセス スコープ (current_scope) の呼び出しスタックにある選択したサブプログラム フレーム (デフォルトは最も最近のサブプログラム呼び出しである top) のインデックスを取得します。current_scope のサブプログラム呼び出しスタックの現在のスタック フレームを設定します。

構文

```
current_frame [-up] [-down] [-set <arg>] [-quiet] [-verbose]
```

戻り値

current_scope の呼び出しスタックにある選択したサブプログラム フレームのインデックス。

使用法

名前	説明
[-up]	現在のフレームとして呼び出し元サブプログラム/プロセスのスタック フレームを選択します。
[-down]	現在のフレームとして呼び出し先サブプログラムのスタック フレームを選択します。
[-set]	現在の HDL プロセス スコープの呼び出しスタックの現在のフレームとして指定のインデックスのスタック フレームを選択します。デフォルト: 0
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の HDL プロセス スコープ (current_scope) のサブプログラム コールスタックで、現在のフレームとして選択されているフレームのインデックスを返します。デフォルトでは、最後に呼び出されたサブプログラム フレームが現在のフレーム (フレーム インデックス 0) で、(->) でマークされています。

-up、-down、-set などのオプションを使用すると、現在のフレームではないコールスタックのほかのフレームを選択できます。



重要: current_frame は current_scope に厳密に従います。current_scope がサブプログラム内で待機中の HDL プロセス スコープでない場合は、current_frame コマンドで現在のプロセスに関連のサブプログラム スタックがないことがレポートされます。

引数

-up (オプション): 呼び出し元のプロセス/プログラムのスタック フレームで、現在のサブプログラム フレームから 1 スタック フレーム (ポップ) 離れたものを現在のフレームとして設定します。

-down (オプション): 呼び出し先のプロセス/プログラムのスタック フレームで、現在のサブプログラム フレームから 1 スタック フレーム (ホップ) 離れたものを現在のフレームとして設定します。

-set <num> (オプション): 現在選択しているコールスタックのインデックス番号が <num> のフレームを現在のフレームとして設定します。-set 0 を設定すると、フレーム スタックが選択したコールスタックの最も深いレベルに戻ります。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

サンプル デザイン:

```
module top;

    int i;

    function void f(input int in1);
        automatic int a;
        a = in1 + 7;
        $display($time, " in f :: a %d in1 %d ", a, in1);
    endfunction

    task automatic t(input int in2);
        int b;
        b = in2 + 10;
        $display($time, " in t :: in2 %d b %d ", in2, b);
        #5;
        f(b);        // Case C
        $display($time, " Back in t : after wait and f(%d) ", b);
    endtask

    initial begin                                // "/top/Initial18_0"
        $display($time, " in initial 1 ");
        i = 200;
        t(i);        // Case B
        $display($time, " Back in initial 1 after t(%d) ", i);
    end

    initial begin                                // "/top/Initial25_1"
        $display($time, " in initial 2 ");
        #2;
        f(50);        // Case A
        $display($time, " Back in initial 2 after f(50) ");
    end
endmodule
```

シミュレーションが関数 f の Case C の呼び出しで停止した場合、関数 f はタスク t の Case C で呼び出され、タスク t はプロセス /top/Initial18_0 の Case B で呼び出されます。

```
> current_scope
/top/Initial18_0
> report_frames
-> 0 : f
    1 : t
    2 : /top/Initial18_0
1. > current_frame
0
2. > current_frame -up
1
> report_frames
0 : f
-> 1 : t
    2 : /top/Initial18_0
3. > current_frame -down
0
> report_frames
-> 0 : f
    1 : t
    2 : /top/Initial18_0
4. > current_frame -set 1
1
> report_frames
0 : f
-> 1 : t
    2 : /top/Initial18_0
5. > current_frame -verbose
-> 0 : f @top.v:6
    1 : t @top.v:15
    2 : /top/Initial18_0 @top.v:21
```

関連項目

- [current_scope](#)
- [report_frames](#)

current_hw_cfgmem

現在のハードウェア コンフィギュレーション メモリを取得または設定します。

構文

```
current_hw_cfgmem [-hw_device <args>] [-quiet] [-verbose] [<hw_cfgmem>]
```

戻り値

ハードウェア コンフィギュレーション メモリ

使用法

名前	説明
[-hw_device]	ハードウェア デバイスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_cfgmem>]	ハードウェア コンフィギュレーション メモリを指定します。デフォルトは現在のハードウェア コンフィギュレーション メモリです。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア コンフィギュレーション メモリ オブジェクトを取得または設定します。

ザイリンクス® FPGA にデザイン特定のデータを読み込む (プログラムする) プロセスは、コンフィギュレーションと呼ばれます。create_hw_cfgmem を使用して、FPGA デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。

新しいハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成すると、それが現在の hw_cfgmem オブジェクトになります。current_hw_cfgmem コマンドを使用すると、現在の hw_cfgmem オブジェクトを返すか、現在の hw_cfgmem オブジェクトを変更できます (get_hw_cfgmems コマンドを使用して返すことにより指定)。

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成し、ハードウェア デバイス (hw_device) に関連付けたら、write_cfgmem コマンドで作成したメモリ コンフィギュレーション ファイルからのビットストリームおよびその他のデータでコンフィギュレーション メモリをプログラムできます。

hw_cfgmem オブジェクトをプログラムするには、program_hw_cfgmem コマンドを使用します。

このコマンドを実行すると、現在のハードウェア コンフィギュレーション メモリがオブジェクトとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-hw_device <arg>` (必須): 現在の `hw_cfgmem` を設定する `hw_device` オブジェクトを指定します。`hw_device` は、`current_hw_device` または `get_hw_devices` コマンドを使用してオブジェクトとして指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_cfgmem>` (オプション): プログラムおよびデバッグ用の現在のコンフィギュレーション メモリとして使用する `hw_cfgmem` オブジェクトを指定します。`hw_cfgmem` は、`get_hw_cfgmems` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、現在のハードウェア コンフィギュレーション メモリ オブジェクトが返されます。

```
current_hw_cfgmem
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [get_hw_cfgmems](#)
- [get_hw_devices](#)
- [program_hw_cfgmem](#)
- [write_cfgmem](#)

current_hw_device

現在のハードウェア デバイスを取得または設定します。

構文

```
current_hw_device [-quiet] [-verbose] [<hw_device>]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_device>]</code>	現在のハードウェア デバイスとして設定するハードウェア デバイスを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite のハードウェア マネージャーでプログラムおよびデバッグ用にターゲットとする現在のザイリンクス FPGA を設定または取得します。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のザイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、`connect_hw_server` コマンドを使用して開き、`hw_server` アプリケーションで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

各ハードウェア ターゲットには、プログラムまたはデバッグ目的で使用するザイリンクス デバイスを 1 つまたは複数含めることができます。`current_hw_device` コマンドは、現在のデバイスを指定するか、返します。

ハードウェア マネージャーを介してザイリンクス FPGA にアクセスするには、次の Tcl コマンド シーケンスを使用する必要があります。

1. `open_hw`: Vivado Design Suite でハードウェア マネージャーを開きます。
2. `connect_hw_server`: ローカルまたはリモートのハードウェア サーバー アプリケーションに接続します。
3. `current_hw_target`: 接続されたサーバーのハードウェア ターゲットを定義します。
4. `open_hw_target`: ハードウェア ターゲットへの接続を開きます。
5. `current_hw_device`: プログラムおよびデバッグに使用するザイリンクス FPGA を指定します。

適切なハードウェア デバイスに接続したら、`program_hw_device` コマンドを使用してデバイスをビットストリーム ファイルでプログラムし、ハードウェア マネージャー Tcl コマンドを使用してデバイスをデバッグできます。デバイスを対話的にデバッグするには、ハードウェア マネージャーを Vivado Design Suite IDE で開きます。



重要: デザインのプログラムおよびデバッグ用のハードウェアを設定するには、`current_hw_server`、`current_hw_target`、および `current_hw_device` コマンドを使用できます。これらのコマンドを使用する際は、現在のサーバー、ターゲット、およびデバイスが揃っていることを確認してください。現在のデバイスは現在のターゲット上にあり、現在のターゲットは現在のサーバー上にある必要があります。

このコマンドを実行すると、現在のハードウェア デバイスがオブジェクトとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_device>` (オプション): プログラムおよびデバッグ用に現在のデバイスとして設定するハードウェア デバイス (`hw_device`) オブジェクトを指定します。`hw_device` は、`get_hw_devices` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア デバイスを指定しない場合、現在のハードウェア デバイス (`current_hw_device`) が返されます。

例

次の例では、現在ターゲットとして指定されている `hw_device` が返されます。

```
current_hw_device
```

次の例では、現在のハードウェア デバイス (`current_hw_device`) を設定し、そのデバイスの `PROGRAM.FILE` プロパティを設定して、デバイスをプログラムしています。

```
current_hw_device [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)

current_hw_ila

現在のハードウェア ILA を取得または設定します。

構文

```
current_hw_ila [-quiet] [-verbose] [<hw_ila>]
```

戻り値

ハードウェア ILA

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_ila>]</code>	ハードウェア ILA を指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite のハードウェア マネージャーでプログラムおよびデバッグ用にターゲットとする現在のハードウェア ILA デバッグ コアを設定または取得します。

ILA (Integrated Logic Analyzer) デバッグ コアを使用すると、サイリンクス FPGA にプログラムされているインプリメント済みデザイン (デザイン ビットストリーム) のインシステム デバッグを実行できます。ILA コアには、ブルートリガー論理式、エッジ遷移トリガーなど、現代のロジック アナライザーで提供される多数のアドバンス機能が含まれています。ILA コアを使用すると、デザインの特定の信号のプロープ、プログラムされたハードウェア イベントでのトリガー、サイリンクス FPGA からのリアルタイムでのデータ キャプチャを実行できます。ILA コアの詳細は、『LogiCORE IP Integrated Logic Analyzer 製品ガイド』 (PG172) を参照してください。

ILA デバッグ コアをデザインに追加するには、IP カタログから ILA コアを RTL デザインにインスタンスエートするか、`create_debug_core` コマンドを使用して合成済みネットリストに ILA コアを追加します。ILA デバッグ コアのデザインへの追加に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

デザインからビットストリームを生成し、`program_hw_devices` コマンドを使用してデバイスをプログラムすると、`get_hw_ilas` コマンドを使用してハードウェア マネージャーからデザインに含まれる ILA デバッグ コアにアクセスできます。デザインの ILA デバッグ コアに割り当てられているデバッグ プロープは、`get_hw_probes` コマンドを使用して取得できます。

このコマンドを実行すると、現在のハードウェア ILA コアがオブジェクトとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_ila>` (オプション): プログラムおよびデバッグ用に現在のデバッグ コアとして設定するハードウェア ILA (`hw_ila`) オブジェクトを指定します。`hw_ila` は、`get_hw_ilas` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア ILA デバッグ コアを指定しない場合、`current_hw_ila` が返されます。

例

次の例では、現在のハードウェア ILA デバッグ コアが返されます。

```
current_hw_ila
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)

current_hw_ila_data

現在のハードウェア ILA データを取得または設定します。

構文

```
current_hw_ila_data [-quiet] [-verbose] [<hw_ila_data>]
```

戻り値

ハードウェア ILA データ

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_ila_data>]	ハードウェア ILA データ

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

現在の ILA デバッグ コア のデータ オブジェクトを設定または取得します。

ILA データ オブジェクトは、Vivado ロジック解析機能で `upload_hw_ila_data` コマンドまたは `read_hw_ila_data` コマンドを使用して作成します。デフォルトでは、現在のハードウェア ILA データ (`hw_ila_data`) オブジェクトは Vivado ロジック解析機能で最後に作成されたハードウェア ILA データ オブジェクトです。 `current_hw_ila_data` コマンドを使用すると、現在のオブジェクトを変更できます。

ILA デバッグ コアでは、ハードウェア デバイスの実行中に、ハードウェア ILA (`hw_ila`) オブジェクトで定義されているイベント トリガーまたはキャプチャ条件に基づいて、リアルタイムでサンプル データをキャプチャできます。 `hw_ila` 上のハードウェア ILA オブジェクト トリガーは、`run_hw_ila` コマンドにより検出可能な状態になります。

ILA データ オブジェクトは、`display_hw_ila_data` コマンドを使用して Vivado ロジック解析機能の波形ウィンドウに表示できます。ILA データを後で使または解析できるようにするため、`write_hw_ila_data` コマンドを使用してディスクに保存できます。

このコマンドを実行すると、キャプチャされたハードウェア ILA デバッグ コア のデータがオブジェクトとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_ila_data>` (オプション): プログラムおよびデバッグ用に現在のデバッグ コアとして設定する `hw_ila_data` オブジェクトを指定します。`hw_ila_data` は、`get_hw_ila_datas` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア ILA データを指定しない場合、現在のハードウェア ILA データ オブジェクト (`current_hw_ila_data`) が返されます。

例

次の例では、ハードウェア ILA デバッグ コアの現在のデータ オブジェクトが返されます。

```
current_hw_ila_data
```

関連項目

- [current_hw_ila](#)
- [display_hw_ila_data](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [read_hw_ila_data](#)
- [run_hw_ila](#)
- [write_hw_ila_data](#)

current_hw_server

現在のハードウェア サーバーを取得または設定します。

構文

```
current_hw_server [-quiet] [-verbose] [<hw_server>]
```

戻り値

ハードウェア サーバー

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_server>]</code>	ハードウェア サーバー

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite に接続されているハードウェア サーバーの中から現在のハードウェア サーバーを定義するか、現在のハードウェア サーバー オブジェクトを返します。

ハードウェア サーバーは、リモートまたはローカル マシンで実行されているサイリンクス ハードウェア サーバー (`hw_server`) アプリケーションのインスタンスです。ハードウェア サーバーは、FPGA デザインをプログラムおよびデバッグするために使用する 1 つまたは複数のサイリンクス デバイスで構成された JTAG チェーンを含むハードウェア ボードハードウェア ターゲットへの接続を制御します。

ハードウェア サーバーを Vivado Design Suite に接続するには、`connect_hw_server` コマンドを使用します。現在のハードウェア サーバー、現在のハードウェア ターゲットおよびデバイスが、ほとんどのハードウェア マネージャー Tcl コマンドの実行対象となります。現在のターゲットおよびデバイスは、`current_hw_target` および `current_hw_device` コマンドを使用して定義できます。

注記: 現在の `hw_server` は、最後に接続したハードウェア サーバーか、このコマンドを使用して定義したハードウェア サーバーです。現在のハードウェア サーバーの接続を解除した場合は、新しく現在の `hw_server` オブジェクトを定義する必要があります。

接続されているハードウェア サーバーのリストを取得するには、`get_hw_servers` コマンドを使用します。使用可能なハードウェア ターゲットおよびデバイスのリストを取得するには、それぞれ `get_hw_targets` および `get_hw_devices` コマンドを使用します。

このコマンドを実行すると、`hw_server` オブジェクトが返されます。`current_hw_server` コマンドの一部として `<hw_server>` を指定した場合、指定したサーバーが現在のハードウェア サーバーとなり、そのオブジェクトが返されます。サーバーを指定しない場合は、`current_hw_server` コマンドで現在のハードウェア サーバー オブジェクトが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_server>` (オプション): 現在のハードウェア サーバーとして設定する `hw_server` を指定します。ハードウェア サーバーを指定しない場合、現在の `hw_server` が返されます。ハードウェア サーバーは、名前で指定するか、または `get_hw_servers` コマンドで 1 つの `hw_server` オブジェクトを返すことにより指定します。

例

次の例では、現在のハードウェア サーバーを指定の名前のものに設定しています。

```
current_hw_server picasso
```

次の例では、現在のハードウェア サーバーが返されます。

```
current_hw_server
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [disconnect_hw_server](#)
- [get_hw_servers](#)
- [get_hw_targets](#)
- [refresh_hw_server](#)

current_hw_target

現在のハードウェア ターゲットを取得または設定します。

構文

```
current_hw_target [-quiet] [-verbose] [<hw_target>]
```

戻り値

ハードウェア ターゲット

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_target>]</code>	ハードウェア ターゲット

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite でハードウェア マネージャーを開き、`connect_hw_server` コマンドを使用してザイリンクス ハードウェア サーバー (`hw_server`) に接続したら、ハードウェア ターゲットを設定する必要があります。このコマンドは、現在のハードウェア ターゲットを設定または取得します。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のザイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、`hw_server` オブジェクトで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

使用可能なハードウェア ターゲットは、Vivado ハードウェア マネージャーが `hw_server` に接続されたときに定義されます。`get_hw_targets` コマンドを使用して使用可能なハードウェア ターゲットのリストを取得し、`current_hw_target` コマンドを使用して現在のハードウェア ターゲットを定義します。

`current_hw_target` コマンドの一部として `<hw_server>` を指定した場合、指定したサーバーが現在のハードウェア サーバーとなり、そのオブジェクトが返されます。`current_hw_target` でハードウェア ターゲットを指定しない場合は、現在のハードウェア ターゲット オブジェクトが返されます。

各ハードウェア ターゲットには、プログラムまたはデバッグ目的で使用するザイリンクス デバイスを 1 つまたは複数含めることができます。`current_hw_device` コマンドは、現在のデバイスを指定するか、返します。現在のハードウェア ターゲットを指定したら、`open_hw_target` コマンドを使用して、そのハードウェア ターゲットを介してザイリンクス デバイスへの接続を開くことができます。



重要: デザインのプログラムおよびデバッグ用のハードウェアを設定するには、`current_hw_server`、`current_hw_target`、および `current_hw_device` コマンドを使用できます。これらのコマンドを使用する際は、現在のサーバー、ターゲット、およびデバイスが揃っていることを確認してください。現在のデバイスは現在のターゲット上にあり、現在のターゲットは現在のサーバー上にある必要があります。

このコマンドを実行すると、現在のハードウェア ターゲットがオブジェクトとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_target>` (オプション): プログラムおよびデバッグ用に現在のハードウェア ターゲットとして設定するハードウェア ターゲット (`hw_target`) オブジェクトを指定します。`hw_target` は、`get_hw_targets` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア ターゲットを指定しない場合、現在のハードウェア ターゲット (`current_hw_target`) が返されます。

例

次の例では、接続されているハードウェア サーバーで使用可能なハードウェア ターゲットを取得し、指定のターゲットを現在のハードウェア ターゲット (`current_hw_target`) に設定しています。

```
get_hw_targets
  trumpet/xilinx_tcf/Digilent/210203327985A
  picasso/xilinx_tcf/Digilent/210203368518A
current_hw_target [lindex [get_hw_targets] 1]
```

関連項目

- [close_hw_target](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)
- [refresh_hw_target](#)

current_instance

現在のインスタンスを設定または取得します。

構文

```
current_instance [-quiet] [-verbose] [<instance>]
```

戻り値

インスタンス名

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<instance>]	インスタンス名を指定します。

カテゴリ

SDC、XDC

説明

デザイン階層の現在のインスタンスを指定のインスタンス セルまたは現在のデザインの最上位に設定します。デフォルトでは、`current_instance` は `current_design` の最上位モジュールを指定します。これは、インスタンス化されたセル オブジェクトではありません。`current_instance` をデザインの中でインスタンス化されたセル オブジェクトに設定することも可能です。



重要: 最上位モジュールはインスタンス化されたオブジェクトではないので、現在のデザイン (`current_instance`) の最上位に設定すると、このコマンドによりデザイン オブジェクトではなく空の文字列が返されます。

ほとんどのコマンドおよびデザインの変更では、現在のデザインおよび現在のインスタンスがターゲットとなります。現在のデザインを指定するには、`current_design` コマンドを使用します。

`<instance>` は現在定義されている現在のインスタンスを基準に指定し、階層区切り文字を使用してインスタンス パスを定義します。現在の階層区切り文字を確認するには、`get_hierarchy_separator` コマンドを使用します。

インスタンス パスの 1 つ上の階層に移動するには、「..」を使用します。

このコマンドを実行すると、現在のインスタンスのデザイン オブジェクトの名前が返されるか、正常に実行されなかった場合は何も返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<instance>` (オプション): 現在のデザインの現在のインスタンスとして設定するインスタンスの名前を指定します。

- `<instance>` は、現在のインスタンスとして定義されているインスタンスを基準に、階層区切り文字を使用して指定します。
- 現在のインスタンスに対して 1 つ上の階層に移動するには、「..」を使用します。
- `<instance>` を指定しない場合、現在のインスタンスがデザイン階層の最上位モジュールにリセットされます。
- `<instance>` に「..」を指定した場合、現在のインスタンスの名前が返され、インスタンスは変更されません。

例

次の例では、現在のインスタンスを現在のデザインの最上位モジュールに設定します。

```
current_instance
INFO: [Vivado 12-618] Current instance is the top level of design
'netlist_1'.
```

次の例では、Vivado IDE で選択しているオブジェクトを現在のインスタンスとして設定しています。

```
current_instance [lindex [get_selected_objects] 0]
```

注記: `get_selected_objects` コマンドでは、オブジェクトが 1 つであってもリストが返されるので、そのリストからオブジェクトを指定するために `lindex` を使用する必要があります。

次の例では、まず階層区切り文字を設定し、その後現在定義されている現在のインスタンスを基準に現在のインスタンスを設定します。

```
set_hierarchy_separator |
current_instance ..|cpu_iwb_dat_o|buffer_fifo
```

次の例では、現在定義されている現在のインスタンスの名前が返されます。

```
current_instance .
cpuEngine|cpu_iwb_dat_o|buffer_fifo
```

関連項目

- [current_design](#)
- [get_hierarchy_separator](#)
- [get_selected_objects](#)
- [set_hierarchy_separator](#)

current_pr_configuration

PartitionDef のリストを取得します。

構文

```
current_pr_configuration [-quiet] [-verbose] [<config>...]
```

戻り値

PartitionDef オブジェクトのリスト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<config>]	現在 (アクティブ) にする PR コンフィギュレーションを指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[Partition \(パーティション\)](#)

説明

現在の PR コンフィギュレーションを取得または設定します。

パーシャル リコンフィギュレーション (PR) デザイン フローでは、PR コンフィギュレーションでパーティション定義 (partitionDef) の特定のインスタンスに割り当てるリコンフィギュラブル モジュール (RM) を指定できます。このフローでは、スタティック デザインと 1 つ以上の RM の組み合わせに基づいて、独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパーシャル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

current_pr_configuration コマンドは、デザインの現在の (アクティブな) PR コンフィギュレーションを返すか、アクティブにする PR コンフィギュレーションを指定します。

このコマンドを実行すると、現在の PR コンフィギュレーションの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<config>` (オプション): アクティブにする PR コンフィギュレーションを指定します。コンフィギュレーションは、名前で指定するか、または `get_pr_configurations` コマンドを使用してオブジェクトとして指定します。コンフィギュレーションを指定しない場合は、現在の PR コンフィギュレーションが返されます。

例

次の例では、アクティブにする PR コンフィギュレーションを指定しています。

```
current_pr_configuration clockHigh
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)
- [create_reconfig_module](#)
- [delete_pr_configurations](#)
- [get_pr_configurations](#)
- [setup_pr_configurations](#)

current_project

現在のプロジェクトを設定または取得します。

構文

```
current_project [-quiet] [-verbose] [<project>]
```

戻り値

現在のまたは新しく設定されたプロジェクト オブジェクト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<project>]	現在のプロジェクトとして設定するプロジェクトの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のプロジェクトを指定するか、プロジェクトを指定しない場合は現在のプロジェクトの名前を返します。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<project> (オプション): 現在のプロジェクトとして設定するプロジェクトの名前を指定します。このコマンドを close_project コマンドの前に使用し、特定のプロジェクトをアクティブにしてから、プロジェクトを閉じます。

例

次の例では、project_2 を現在のプロジェクトとして設定しています。

```
current_project project_2
```

このコマンドにより、現在のプロジェクトがすべてのツール コマンドの対象となります。GUI モードでは、プロジェクト間の GUI を切り替えると現在のプロジェクトが自動的に定義されます。

次の例では、現在のプロジェクト名が返されます。

```
current_project
```

注記: 戻り値は、プロジェクト ファイルの名前やパスではなく、プロジェクトの名前です。

関連項目

- [close_project](#)
- [create_project](#)
- [current_design](#)

current_run

現在の run を設定または取得します。

構文

```
current_run [-synthesis] [-implementation] [-quiet] [-verbose] [<run>]
```

戻り値

run オブジェクト

使用法

名前	説明
[-synthesis]	現在の合成 run を設定または取得します。
[-implementation]	現在のインプリメンテーション run を設定または取得します。-synthesis を指定しない場合、これがデフォルト設定です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<run>]	現在の run として設定する run を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在の合成またはインプリメンテーション run を定義するか、現在の run の名前を返します。現在の run とは、合成またはインプリメント コマンドを実行したときに自動的に選択される run のことです。

get_runs コマンドを使用すると、現在のデザインで定義されている run のリストを取得できます。

引数

-synthesis (オプション): current_run コマンドで現在の合成 run を設定または返します。

-implementation (オプション): current_run コマンドで現在のインプリメンテーション run を設定または返します。これが、-synthesis または -implementation が指定されていない場合のデフォルトです。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<run> (オプション): 現在の run として設定する合成 run またはインプリメンテーション run の名前を指定します。

例

次の例では、synth_1 という run を現在の run として定義しています。

```
current_run synth_1
```

注記: この場合、run の名前が指定されており、ツールで特定の run 名が認識されるので、-synthesis および -implementation オプションは必要ありません。

次のコマンドでは、現在のインプリメンテーション run の名前が返されます。

```
current_run -implementation -quiet
```

関連項目

- [create_run](#)
- [get_runs](#)
- [launch_runs](#)

current_scope

現在のスコープを取得するか、現在のスコープを設定します。

構文

```
current_scope [-quiet] [-verbose] [<hdl_scope>]
```

戻り値

現在のスコープ

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hdl_scope>]	デフォルトは NULL です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションの現在のスコープを取得するか、現在のスコープを指定の HDL スコープに設定します。

`current_scope` コマンドでは、現在のシミュレーション スコープの名前が返されます。

`<hdl_scope>` を指定すると、現在のスコープが指定のスコープに設定されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hdl_scope>` (オプション): 現在のシミュレーションの現在のスコープとして設定する HDL スコープを指定します。スコープは、スコープの絶対階層パス名 (`/tb/UUT` など) または相対パス名 (`/uut`、`gt`、`./gt`、`uut/fg` など) として指定するか、`get_scopes` コマンドで返される HDL スコープ オブジェクトとして指定します。



ヒント: HDL スコープとして `/` を指定すると、スコープは現在のシミュレーションの最上位スコープにリセットされます。

例

次の例では、現在のスコープを指定の HDL スコープに設定しています。

```
current_scope /testbench/dut
```

次の例では、現在のスコープ名をコンソールに表示しています。

```
current_scope
```

関連項目

- [get_scopes](#)
- [report_scopes](#)

current_sim

現在のシミュレーション オブジェクトを設定するか、現在のシミュレーション オブジェクトを取得します。

構文

```
current_sim [-quiet] [-verbose] [<simulationObject>]
```

戻り値

現在のシミュレーション オブジェクト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<simulationObject>]	現在のシミュレーション オブジェクトに設定するシミュレーション オブジェクトを指定します。デフォルトは NULL です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の Vivado シミュレーション オブジェクトを取得または設定します。

このコマンドは、Vivado シミュレータを起動した後に実行して、シミュレーション オブジェクトを取得するか、現在のシミュレーション オブジェクトを設定します。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

simulationObject (オプション): 現在のシミュレーションとして設定するシミュレーション オブジェクトの名前を指定します。<simulationObject> を指定しない場合、現在のプロジェクトの現在のシミュレーションが返されます。

例

次の例では、現在のシミュレーションを設定しています。

```
current_sim simulation_2
```

関連項目

- [close_sim](#)

current_time

現在のシミュレーション時間をレポートします。

構文

```
current_time [-s] [-quiet] [-verbose]
```

戻り値

現在のシミュレーション時間

使用法

名前	説明
<code>[-s]</code>	数値と単位のためのスペースを削除して短く表示します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーション時間を Tcl コンソールまたは Vivado Design Suite Tcl シェルに返します。

引数

`-s` (オプション): 時間の数値部分と単位部分の間のスペースを削除し、解析しやすくします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のシミュレーションの時間が返されます。

```
current_time
```

関連項目

- [restart](#)
- [run](#)
- [stop](#)

current_vcd

現在の VCD オブジェクトを設定するか、現在の VCD オブジェクトを取得します。

構文

```
current_vcd [-quiet] [-verbose] [<VCDObject>]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<VCDObject>]	VCD オブジェクトを指定します。デフォルトは NULL です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の VCD (Value Change Dump) を定義するか、現在のシミュレーションにおける現在の VCD オブジェクトの名前を返します。

現在の VCD オブジェクトが存在するようにするためには、`open_vcd` コマンドを使用して VCD ファイルを開き、VCD オブジェクトに割り当てる必要があります。

このコマンドを実行すると、現在の VCD オブジェクトが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<VCDObject>` (オプション): 現在のオブジェクトとして設定する VCD オブジェクトの名前を指定します。
`<VCDObject>` を指定しない場合、アクティブ シミュレーションの現在の VCD オブジェクトが返されます。

例

次の例では、指定した VCD オブジェクトを現在の VCD オブジェクトとして設定しています。

```
current_vcd vcd2
```

関連項目

- [open_vcd](#)

current_wave_config

現在の波形設定 (WCFG) オブジェクトを取得するか、WCFG を指定した場合は現在の WCFG を設定します。

構文

```
current_wave_config [-quiet] [-verbose] [<wcfgObj>]
```

戻り値

新しいまたは現在の波形設定オブジェクト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<wcfgObj>]	指定した wcfgObj を現在の WCFG に設定します。デフォルトは、現在の WCFG です。

カテゴリ

[Waveform \(波形\)](#)

説明

現在の波形設定オブジェクトを指定するか、現在の波形設定オブジェクトを取得します。

Vivado® シミュレータ GUI では、波形ウィンドウを使用してデザインを解析し、コードをデバッグできます。波形設定には最上位 HDL オブジェクトが表示されますが、`add_wave` および `add_wave_divider` などのコマンドを使用してオブジェクトを追加できます。

このコマンドを実行すると、現在の波形設定オブジェクトの名前が返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<wcfgObj>` (オプション): 現在の波形設定オブジェクトとして設定する波形設定オブジェクトを指定します。波形設定オブジェクトは、名前で指定するか、または `get_wave_configs` コマンドで 1 つのオブジェクトを返すことにより指定します。

例

次の例では、テストベンチ波形設定オブジェクトを取得し、シミュレーションの現在の波形設定として設定しています。

```
current_wave_config [get_wave_config testbench]
```

関連項目

- [create_wave_config](#)
- [get_wave_configs](#)
- [open_wave_config](#)

decrypt_bitstream

AES-GCM で暗号化されたビットストリームを復号化します。

構文

```
decrypt_bitstream -encrypted_file <arg> -keyfile <arg> [-force] [-quiet]  
                  [-verbose] <file>
```

使用法

名前	説明
-encrypted_file	AES-GCM で暗号化されたビットストリーム (.bit または .rbt) を指定します。
-keyfile	暗号キーを含むファイルを指定します。
[-force]	既存のファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	出力する復号化されたビットストリーム (.bit、.bin、または .rbt) を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

保護された暗号化 UltraScale アーキテクチャ デザインのインプリメンテーションでは、最終ビットストリームでサイリンクスでテスト済みの Security Monitor (SecMon) IP のゴールデン ビットストリームに対するビットストリーム レベル検証を実行する必要があります。

decrypt_bitstream コマンドは、インプリメント済みデザインから SecMon IP を含む AES-GCM で暗号化されたビットストリーム ファイル (.bit または .rbt) と暗号キー ファイル (.nky または .nkz) を読み込み、されたビットストリーム ファイルを返します。この復号化されたビットストリームは、ビットストリーム検証プロセスを完了するために使用できます。

このコマンドを実行すると、正常に実行された場合は復号化ファイルが返され、正常に実行されなかった場合はエラーが返されます。

引数

-encrypted_file <arg> (必須): 復号する AES-GCM で暗号化されたビットストリーム (.bit または .rbt) を指定します。

-keyfile <arg> (必須): 暗号キー ファイル (.nky または .nkz) の名前を指定します。このオプションの指定は、暗号化されたビットストリームを復号化するために必要です。

-force (オプション): 指定した名前のファイルが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 復号化ビットストリームを標準フォーマット (`.bit`) またはヘッダー情報なし (`.bin`) で記述します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定の暗号化されたビットストリームを復号化しています。

```
decrypt_bitstream -encrypted_file C:/Data/myDesign.bit \
-keyfile C:/Data/key.nky -force C:/Data/myDesign_decrypted.bit
```

関連項目

- [write_bitstream](#)

delete_bd_objs

指定のオブジェクト削除します。

構文

```
delete_bd_objs [-quiet] [-verbose] <objects>...
```

戻り値

オブジェクトが正しく削除された場合は Pass

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	削除するオブジェクトを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインから指定のオブジェクトを削除します。

オブジェクトは `delete_bd_objs` コマンドに直接渡す必要があります。オブジェクト名で指定することはできません。たとえばピンを指定する場合、ピン名ではなく `get_bd_pins` を使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>`: 現在の IP インテグレーター サブシステム デザインから削除するオブジェクトを指定します。

例

次の例では、現在のサブシステム デザインからさまざまなオブジェクトを削除しています。

```
delete_bd_objs [get_bd_nets /Net] [get_bd_nets /vidout_1_vtg_ce] \
[get_bd_nets /newMod1/aclk_1] [get_bd_ports /addr] [get_bd_cells /vidOut_1]
```

次の例では同じオブジェクトを削除していますが、複数の `delete_bd_objs` コマンドを使用し、オブジェクトをタイプ別にグループ化して削除するオブジェクトを明確にしています。

```
delete_bd_objs [get_bd_nets /Net] [get_bd_nets /vidout_1_vtg_ce] \
[get_bd_nets /newMod1/aclk_1]
delete_bd_objs [get_bd_ports /addr]
delete_bd_objs [get_bd_cells /vidOut_1]
```

関連項目

- [get_bd_addr_segs](#)
- [get_bd_addr_spaces](#)

delete_clock_networks_results

メモリからクロック ネットワーク結果のセットを削除します。

構文

```
delete_clock_networks_results [-quiet] [-verbose] <name>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	削除する結果の名前を指定します。

カテゴリ

[Report \(レポート\)](#)

説明

指定の結果セットから指定の `report_clock_networks` レポートを削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 削除するクロック ネットワーク結果の名前を指定します。

例

次の例では、メモリから指定した結果を削除しています。

```
delete_clock_network_results ClkNets
```

関連項目

- [report_clock_networks](#)

delete_dashboard_gadgets

プロジェクト サマリ ダッシュボードのガジェットを削除します。

構文

```
delete_dashboard_gadgets [-quiet] [-verbose] <gadgets>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><gadgets></code>	削除するガジェットを指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

プロジェクト サマリ ダッシュボードからガジェットを削除し、プロジェクトからも削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<gadgets>` (必須): 削除するダッシュボード ガジェットを指定します。ガジェットは、名前で指定するか、または `get_dashboard_gadgets` コマンドを使用してオブジェクトとして指定します。

例

次の例では、指定したガジェットを削除しています。

```
delete_dashboard_gadget gadget_power
```

関連項目

- [create_dashboard_gadget](#)
- [get_dashboard_gadgets](#)

- [move_dashboard_gadget](#)

delete_debug_core

デバッグ コアを削除します。

構文

```
delete_debug_core [-quiet] [-verbose] <cores>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><cores></code>	削除するデバッグ コアを指定します。

カテゴリ

Debug (デバッグ)

説明

現在のプロジェクトから `create_debug_core` コマンドで追加された Vivado Lab Edition デバッグ コアを削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cores>` (必須): 現在のプロジェクトから削除するデバッグ コアを 1 つ以上指定します。

例

次の例では、現在のプロジェクトから `myCore` デバッグ コアを削除しています。

```
delete_debug_core myCore
```

次の例では、現在のプロジェクトからすべてのデバッグ コアを削除しています。

```
delete_debug_core [get_debug_cores]
```

注記: `get_debug_cores` コマンドを実行すると、すべてのコアがデフォルトで返されます。

関連項目

- [create_debug_core](#)
- [get_debug_cores](#)

delete_debug_port

デバッグ ポートを削除します。

構文

```
delete_debug_port [-quiet] [-verbose] <ports>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><ports></code>	削除するデバッグ ポートを指定します。

カテゴリ

Debug (デバッグ)

説明

現在のプロジェクトに含まれる Vivado Lab Edition デバッグ コアからポートを削除します。デバッグ ポートからの信号は、`disconnect_debug_port` で削除するか、このコマンドでポートと共に削除できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<ports>` (必須): コアから削除するデバッグ ポートを `<core_name>/<port_name>` の形式で指定します。

例

次の例では、`myCore` デバッグ コアから `DATA` ポートを削除しています。

```
delete_debug_port myCore/DATA
```

注記: ILA ポートには、最低 1 つの `CLK` ポートと 1 つの `TRIG` ポートが必要なので、削除できないポートもあります。

次の例では、myCore デバッグ コアからトリガー ポート (TRIG) が削除されます。

```
delete_debug_port [get_debug_ports myCore/TRIG*]
```

注記: この例の場合、ILA コアには少なくとも 1 つの TRIG ポートが必要なので、myCore からすべての TRIG ポートが削除されるわけではありません。このコマンドでは、最後のポートを除き、TRIG0 からすべての TRIG ポートが削除されます。

関連項目

- [disconnect_debug_port](#)
- [get_debug_ports](#)

delete_drc_check

ユーザー定義 DRC チェックを削除します。

構文

```
delete_drc_check [-quiet] [-verbose] <name>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	削除するチェックの名前を指定します。これは通常 PREFIX-id という形式で指定します (PREFIX は 4 ～ 6 文字の略称、id はルールを識別する整数)。正しい名前を確認するには、 <code>get_drc_checks</code> を使用します。削除できるのは、ユーザー定義の DRC チェックのみです。

カテゴリ

DRC、Object (オブジェクト)

説明

現在のプロジェクトからユーザー定義デザイン ルール チェックを削除します。ユーザー定義デザイン ルール チェックは、`create_drc_checks` コマンドを使用して作成します。

注記: ツールであらかじめ定義されているルール チェックは削除できません。

ルール チェックを一度削除すると、復元するできません。`undo` コマンドは機能しません。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 現在のプロジェクトから削除するユーザー定義デザイン ルール チェックの名前を指定します。

例

次の例では、指定したデザイン ルール チェックを削除しています。

```
delete_drc_check LJH-1
```

関連項目

- [create_drc_check](#)

delete_drc_ruledeck

1 つまたは複数のユーザー定義 DRC ルール デック オブジェクトを削除します。

構文

```
delete_drc_ruledeck [-regex] [-nocase] [-filter <arg>] [-quiet] [-verbose]  
[<patterns>]
```

戻り値

drc_ruledeck オブジェクト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	drc_ruledeck オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

DRC、Object (オブジェクト)

説明

現在のプロジェクトから 1 つまたは複数のユーザー定義デザイン ルール デック オブジェクトを削除します。削除するルール デックは空である必要はありません。一度削除すると、復元できません。undo コマンドは機能しません。

注記: ツールであらかじめ定義されているルール デックは削除できません。

ルール デックとは、デザイン ルール チェックのグループで、I/O プランニングなどの FPGA デザイン フローの異なる段階で report_drc コマンドにより実行されます。ツールには定義済みのルール デックが含まれていますが、create_drc_ruledeck コマンドを使用して新しいユーザー定義のルール デックを作成できます。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` <args> (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、検索パターンにより返されるオブジェクトのリストに、指定したプロパティ値に基づくフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): 削除する DRC ルール デックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のプロジェクトのユーザー定義ツール デックがすべて削除されます。複数の検索パターンを指定して、異なる検索条件に基づいてルール デックを削除できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のプロジェクトからすべてのユーザー定義ルール デックを削除しています。

```
delete_drc_ruledeck
```

関連項目

- [create_drc_ruledeck](#)
- [list_property](#)
- [report_property](#)

delete_fileset

ファイルセットを削除します。

構文

```
delete_fileset [-merge <arg>] [-quiet] [-verbose] <fileset>
```

使用法

名前	説明
<code>[-merge]</code>	削除するファイルセットからのファイルを結合するファイルセットを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><fileset></code>	削除するファイルセットを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

指定したファイルセットを削除します。ただし、ファイルセットが削除できない場合でも、それを示すメッセージは表示されません。

引数

`-merge <arg>` (オプション): 削除するファイルセットからのファイルを結合するファイルセットを指定します。 `-merge` オプションを指定しない場合、ファイルセットのすべてのファイルがプロジェクトから削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<fileset>` (必須): 削除するファイルセットの名前を指定します。最後の制約ファイルセットまたはシミュレーションファイルセットは削除されません。その場合、エラー メッセージは表示されません。

例

次の例では、sim_2 というファイルセットを現在のプロジェクトから削除しています。

```
delete_fileset sim_2
```

注記: ファイルセットとそのファイルすべてがプロジェクトから削除されます。ファイルは、ハード ドライブからは削除されません。

関連項目

- [create_fileset](#)
- [current_fileset](#)

delete_hw_axi_txn

ハードウェア AXI トランザクション オブジェクトを削除します。

構文

```
delete_hw_axi_txn [-quiet] [-verbose] <hw_axi_txns>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_axi_txns></code>	削除するハードウェア AXI トランザクション オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア AXI (`hw_axi`) オブジェクトから指定の AXI トランザクション オブジェクト (`hw_axi_txn`) を削除します。

`create_hw_axi_txn` コマンドでは、既存のオブジェクトと同じ名前のオブジェクトを作成することはできません。新しい AXI トランザクション オブジェクトを作成する前に、このコマンドを使用して既存のオブジェクトを削除してください。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_axi_txns>` (必須): 削除する `hw_axi_txn` オブジェクトを指定します。`hw_axi_txn` は、`get_hw_axi_txns` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定の hw_axi に関連付けられている hw_axi_txn オブジェクトを削除しています。

```
delete_hw_axi_txn [get_hw_axi_txns readAxi1]
```

関連項目

- [get_hw_axis](#)
- [get_hw_axi_txns](#)
- [refresh_hw_axi](#)
- [reset_hw_axi](#)

delete_hw_bitstream

ハードウェア デバイスからハードウェア ビットストリーム オブジェクトを削除します。

構文

```
delete_hw_bitstream [-of_objects <args>] [-quiet] [-verbose]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
<code>[-of_objects]</code>	指定した hw_device の hw_bitstream オブジェクトを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

指定のハードウェア デバイス (hw_device) オブジェクトからハードウェア ビットストリーム (hw_bitstream) オブジェクトを削除します。

hw_device オブジェクトの PROGRAM.HW_BITSTREAM および PROGRAM.FILE プロパティがクリアされ、hw_bitstream オブジェクトが削除されます。

引数

`-of_objects <arg>` (オプション): 指定の hw_device に関連付けられている hw_bitstream オブジェクトを削除します。ターゲットは、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のハードウェア デバイスから hw_bitstream オブジェクトを削除しています。

```
delete_hw_bitstream -of_objects [current_hw_device]
```

関連項目

- [create_hw_bitstream](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [program_hw_devices](#)
- [set_property](#)
- [write_bitstream](#)

delete_hw_cfgmem

メモリからハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを削除します。

構文

```
delete_hw_cfgmem [-quiet] [-verbose] <cfgmem>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><cfgmem></code>	hw_cfgmem オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア デバイス (hw_device) から指定した hw_cfgmem オブジェクトを削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cfgmem>` (必須): 現在の hw_device から削除する hw_cfgmem オブジェクトを指定します。hw_cfgmem オブジェクトは、名前で指定するのではなく、`get_hw_cfgmems` または `current_hw_cfgmem` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、現在の hw_cfgmem オブジェクトを削除しています。

```
delete_hw_cfgmem [current_hw_cfgmem]
```

関連項目

- [create_hw_cfgmem](#)

- [current_hw_device](#)
- [get_hw_cfgmems](#)
- [get_property](#)
- [program_hw_cfgmem](#)

delete_hw_probe

ハードウェア プロブ オブジェクトを削除します。

構文

```
delete_hw_probe [-quiet] [-verbose] <hw_probes>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_probes></code>	削除するハードウェア プロブ オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア ILA (hw_ila) からユーザー定義プロブを削除します。ユーザー定義プロブは、`create_hw_probe` コマンドを使用して作成したものである必要があります。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_probes>`: 現在の ILA コアから削除するユーザー定義 hw_probe オブジェクトを指定します。プロブは、`get_hw_probes` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、現在の ILA コアからユーザー定義 hw_probe を削除しています。

```
delete_hw_probe [get_hw_probe probeAR]
```

関連項目

- [create_hw_probe](#)
- [current_hw_ila](#)
- [get_hw_ilas](#)
- [get_hw_probes](#)

delete_hw_target

hw_target を削除します。

構文

```
delete_hw_target [-quiet] [-verbose] [<target_object>]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<target_object>]	削除するターゲット オブジェクトを指定します。デフォルトは現在のハードウェア ターゲットです。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア サーバー (current_hw_server) から仮想ハードウェア ターゲットを削除します。

ハードウェア ターゲット (hw_target) は、create_hw_target コマンドを使用して作成された仮想ターゲットである必要があり、そうでない場合はエラーが返されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<target_object> (オプション): 削除する SVF または仮想ターゲットを指定します。ターゲットを指定しない場合は、現在のハードウェア ターゲット (current_hw_target) の削除が試みられます。

例

次の例では、指定の hw_target を削除しています。

```
delete_hw_target [lindex [get_hw_targets] 1]
```

関連項目

- [create_hw_target](#)

delete_interface

I/O ポート インターフェイスをプロジェクトから削除します。

構文

```
delete_interface [-all] [-quiet] [-verbose] <interfaces>...
```

使用法

名前	説明
[-all]	インターフェイスに付属するすべてのポートおよびバスも削除します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<interfaces>	削除する I/O ポート インターフェイスを指定します。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

既存のインターフェイスおよびオプションでそのインターフェイスに関連付けられているポートおよびバスをすべて削除します。

引数

-all (オプション): 指定のインターフェイスに関連付けられているポート、バス、および入れ子のインターフェイスをすべて削除します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<interfaces> (必須): 削除するインターフェイスの名前を指定します。

例

次の例では、指定のインターフェイスと、関連付けられているポートおよびバスがすべて削除されます。

```
delete_interface USB0
```

関連項目

- [create_interface](#)
- [create_port](#)

delete_ip_run

指定の IP に関連付けられているブロック ファイルセットと run を削除します。

構文

```
delete_ip_run [-force] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-force]</code>	ブロック ファイルセットおよび run を強制的に削除します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ブロック ファイルセットおよび run を削除する IP オブジェクトを <code>get_ips</code> または <code>get_files</code> コマンドを使用して指定します。

カテゴリ

Project (プロジェクト)、IPFlow (IP フロー)

説明

指定した IP モジュールのアウト オブ コンテキスト (OOC) 合成 run およびインプリメンテーション run を削除します。

run と共に run のディレクトリもプロジェクトから削除されます。ただし、run で作成され、IP ソース フォルダにコピーされた出力ファイル、DCP ファイル、Verilog および VHDL 構造ネットリストは、プロジェクトから削除されません。IP 出力ファイルをアップデートするには、`reset_target` または `generate_target` コマンドを使用する必要があります。



重要: IP オブジェクトは `get_ips` または `get_files` コマンドで指定する必要があり、run の名前または `get_runs` コマンドで返される run オブジェクトに基づいて run を削除することはできません。

引数

`-force`: run を強制的に削除します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): run を削除する IP オブジェクトまたはファイルを指定します。get_ips コマンドを使用して 1 つの IP オブジェクトを指定するか、get_files コマンドを使用して 1 つの IP ファイル (XCI) を指定します。

例

次の例では、指定した IP モジュールの OOC 合成 run およびインプリメンテーション run を削除しています。

```
delete_ip_run [get_ips add1]
```

注記: この例の場合、run 結果もハード ドライブのプロジェクト ディレクトリから削除されます。

関連項目

- [create_ip_run](#)
- [generate_target](#)
- [get_files](#)
- [get_ips](#)
- [reset_target](#)

delete_macros

マクロを削除します。

構文

```
delete_macros [-quiet] [-verbose] <macros>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><macros></code>	削除するマクロを指定します。

カテゴリ

XDC

説明

`create_macro` コマンドで定義した 1 つまたは複数のマクロを削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<macros>` (必須): 削除するマクロの名前を指定します。

例

次の例では、`usbMacro1` というマクロを削除しています。

```
delete_macros usbMacro1
```

関連項目

- [create_macro](#)

delete_partition_defs

既存のパーティション定義を削除します。

構文

```
delete_partition_defs [-merge <arg>] [-quiet] [-verbose] <partition_defs>
```

使用法

名前	説明
[-merge]	削除されたパーティション定義のデフォルト RM からのファイルを結合するファイルセットを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<partition_defs>	削除するパーティション定義を 1 つ以上指定します。

カテゴリ

Partition (パーティション)

説明

現在のプロジェクトから指定したパーティション定義 (partitionDef) オブジェクトを削除します。

このコマンドを実行すると、ファイル結合プロセスが返され、ファイルが結合されない場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-merge <arg> (オプション): 削除する partitionDef オブジェクトのデフォルト リコンフィギャラブル モジュール (DEFAULT_RM) からのファイルを結合するファイルセットの名前を指定します。ファイルはデフォルトの RM から指定したファイルセットに移動します。



ヒント: -merge オプションを指定しない場合、デフォルトの RM に含まれるすべてのファイルがプロジェクトから削除されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<partition_defs> (必須): 現在のプロジェクトから削除する partitionDef オブジェクトを 1 つまたは複数指定します。partitionDef は、名前で指定するか、または `get_partition_defs` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、現在のデザインからすべての partitionDef を削除し、各パーティションのデフォルト RM からのファイルをデザインのソース ファイルセットに結合しています。

```
delete_partition_defs -merge sources_1 [get_partition_defs]
```

関連項目

- [create_partition_def](#)
- [get_partition_defs](#)

delete_pblocks

Pblock を削除します。

構文

```
delete_pblocks [-hier] [-quiet] [-verbose] <pblocks>...
```

使用法

名前	説明
<code>[-hier]</code>	Pblock の子ブロックすべても削除します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><pblocks></code>	削除する Pblock を指定します。

カテゴリ

Floorplan (フロアプラン)、XDC

説明

現在のプロジェクトから指定した Pblock を削除します。Pblock は、`create_pblock` コマンドを使用して作成します。

引数

`-hier` (オプション): 指定した Pblock に含まれる Pblock も削除します。`-hier` オプションを指定せずに親 Pblock を削除した場合、入れ子の Pblock は 1 つ上のレベルに移動されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pblocks>` (必須): 削除する Pblock を 1 つ以上指定します。

例

次の例では、指定した Pblock と、その Pblock に含まれる Pblock が削除されます。

```
delete_pblocks -hier cpuEngine
```

関連項目

- [create_pblock](#)

delete_power_results

指定した名前の消費電力見積もり結果をメモリから削除します。

構文

```
delete_power_results -name <arg> [-quiet] [-verbose]
```

使用法

名前	説明
-name	削除する結果の名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Power (電力)

説明

特定の結果セットの消費電力解析結果を削除します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

-name <arg> (必須): 削除する結果の名前を指定します。名前は、ユーザーが定義したものか、report_power コマンドを実行したときに自動的に定義されたものです。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、消費電力解析を実行し、その結果を削除しています。

```
report_power -name my_set  
delete_power_results -name my_set
```

関連項目

- [power_opt_design](#)
- [report_power](#)
- [reset_switching_activity](#)
- [set_switching_activity](#)

delete_pr_configurations

既存のコンフィギュレーションを削除します。

構文

```
delete_pr_configurations [-quiet] [-verbose] <configs>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><configs></code>	削除するコンフィギュレーションを 1 つ以上指定します。

カテゴリ

Partition (パーティション)

説明

現在のプロジェクトから指定した PR コンフィギュレーションを削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<configs>` (必須): 現在のプロジェクトから削除するコンフィギュレーションを 1 つまたは複数指定します。コンフィギュレーションは、名前で指定するか、または `get_pr_configurations` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、指定した PR コンフィギュレーションが削除されます。

```
delete_pr_configurations [get_pr_configurations clockHigh]
```

関連項目

- [create_pr_configuration](#)
- [get_pr_configurations](#)
- [setup_pr_configurations](#)

delete_qor_suggestions

使用可能な QoR 推奨項目のリストを削除します。

構文

```
delete_qor_suggestions [-quiet] [-verbose] [<IDs>]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<IDs>]	名前を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Feasibility \(設計実現可能性\)](#)、[Timing \(タイミング\)](#)

説明

現在メモリに存在する QoR 推奨項目オブジェクトを削除します。推奨項目オブジェクトは、`report_qor_suggestion` コマンドを `-of_objects` オプションを使用せずに実行すると生成され、`read_qor_suggestions` コマンドを RQS ファイルを指定して実行すると読み込まれます。QoR オブジェクトを取得するには、`get_qor_suggestions` コマンドを使用します。

引数

<IDs> (必須): 推奨項目オブジェクト ID の Tcl リストを指定します。ID は、`get_qor_suggestions` コマンドを使用して取得できます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、メモリにあるすべての推奨項目オブジェクトを削除します。

```
delete_qor_suggestions [get_qor_suggestions]
```

次の例は、Utilization (使用率) カテゴリの推奨項目のみを削除します。

```
delete_qor_suggestions [get_qor_suggestions -filter {CATEGORY==Utilization}]
```

関連項目

- [get_qor_suggestions](#)
- [read_qor_suggestions](#)
- [report_qor_assessment](#)
- [write_qor_suggestions](#)

delete_reconfig_modules

既存のリコンフィギャラブル モジュールを削除します。

構文

```
delete_reconfig_modules [-merge <arg>] [-quiet] [-verbose] <rms>
```

使用法

名前	説明
<code>[-merge]</code>	削除するリコンフィギャラブル モジュールからのファイルを結合するファイルセットを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><rms></code>	削除するリコンフィギャラブル モジュールを 1 つ以上指定します。

カテゴリ

Partition (パーティション)

説明

現在のプロジェクトから指定したリコンフィギャラブル モジュール (RM) を削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-merge <arg>` (オプション): 削除する RM からのファイルを結合するファイルセットの名前を指定します。ファイルは削除する RM から指定したファイルセットに移動します。



ヒント: `-merge` オプションを指定しない場合、削除する RM に含まれるすべてのファイルがプロジェクトから削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<rms>` (必須): 現在のプロジェクトから削除する RM オブジェクトを 1 つまたは複数指定します。RM は、名前で指定するか、または `get_reconfigurable_modules` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、指定した RM が削除されます。

```
delete_reconfig_modules usbBlock
```

関連項目

- [create_reconfig_module](#)
- [get_reconfig_modules](#)

delete_report_configs

既存の設定可能なレポート オブジェクトを削除します。

構文

```
delete_report_configs [-quiet] [-verbose] <report_configs>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><report_configs></code>	削除する設定可能なレポート オブジェクトのリストを指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[Report \(レポート\)](#)

説明

現在のプロジェクトから指定のレポート オブジェクトを削除します。レポート オブジェクトは、`create_report_config` コマンドで作成します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<report_configs>` (必須): 現在のプロジェクトから削除するレポート オブジェクトのリストを 1 つまたは複数指定します。レポートは、`get_report_configs` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定した `report_config` オブジェクトを削除しています。

```
delete_report_configs [get_report_configs post_route_datasheet]
```

関連項目

- [create_report_config](#)
- [get_report_configs](#)

delete_rpm

RPM を削除します。

構文

```
delete_rpm [-quiet] [-verbose] <rpm>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><rpm></code>	削除する RPM を指定します。

カテゴリ

[Floorplan \(フロアプラン\)](#)

説明

デザインから指定した相対配置マクロ (RPM) を削除します。

RPM とは、複数のロジック エLEMENT (FFS、LUT、CY4、RAM など) を 1 つのセット (U_SET、H_SET、および HU_SET) にまとめたものです。セット内での各ELEMENTの配置は、相対ロケーション制約 (RLOC) により同じセット内のほかのELEMENTに相対して設定されます。RLOC 制約の設定されたロジック ELEMENTと共通セット名は、RPM 内で関連付けられます。これらの制約の定義方法は、『制約ガイド』 (UG625) を参照してください。

デザインから削除できるのは、ユーザー定義の RPM のみです。階層またはネットリストにより定義された RPM をこのコマンドで削除することはできません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<rpm>` (必須): 削除する RPM を指定します。

例

次の例では、デザインから指定した RPM (cs_ila_0/U0) を削除しています。

```
delete_rpm cs_ila_0/U0
```

delete_runs

既存の run を削除します。

構文

```
delete_runs [-noclean_dir] [-quiet] [-verbose] <runs>
```

使用法

名前	説明
[-noclean_dir]	すべての出力ファイルおよびディレクトリをディスクから削除しません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<runs>	変更する run を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

プロジェクトから run を削除し、指定しない場合はハード ドライブのプロジェクト ディレクトリからも run の結果を削除します。

引数

-noclean_dir: ハード ドライブから run 結果を削除しないように指定します。run はプロジェクトからは削除されますが、run ファイルはプロジェクト ディレクトリに残ったままになります。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<runs> (必須): プロジェクトから削除する合成 run またはインプリメンテーション run の名前を指定します。

例

次の例では、first_pass という run をプロジェクトから削除しています。

```
delete_runs first_pass
```

注記: この例の場合、run 結果もハード ドライブのプロジェクト ディレクトリから削除されます。

次の例では、first_pass という run は削除されますが、run 結果はハード ドライブに残ります。

```
delete_runs -noclean_dir first_pass
```

関連項目

- [create_run](#)
- [current_run](#)

delete_timing_results

メモリからタイミング結果のセットを削除します。

構文

```
delete_timing_results [-type <arg>] [-quiet] [-verbose] <name>
```

使用法

名前	説明
<code>[-type]</code>	削除するタイミング結果のタイプを指定します。有効な値は、 <code>bus_skew</code> 、 <code>check_timing</code> 、 <code>clock_interaction</code> 、 <code>clock_domain_crossings</code> 、 <code>config_timing</code> 、 <code>datasheet</code> 、 <code>pulse_width</code> 、 <code>slack_histogram</code> 、 <code>timing_path</code> 、 <code>timing_summary</code> です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	削除する結果の名前を指定します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

特定の結果セットから指定のタイミング結果を削除します。タイミング レポートのタイプとタイミング レポートの名前を指定する必要があります。これら両方を指定しないと、エラーが発生します。

引数

`-type <arg>` (オプション): 削除するタイミング結果のタイプを指定します。指定可能なタイプは次のとおりです。

- `bus_skew`: 指定した `report_bus_skew` レポートを削除します。
- `check_timing`: 指定した `check_timing` レポートを削除します。
- `clock_interaction`: 指定した `report_clock_interaction` レポートを削除します。
- `clock_domain_crossing`: 指定した CDC レポートを削除します。
- `config_timing`: 現在のタイミング設定レポートを削除します。
- `datasheet`: 指定した `report_datasheet` レポートを削除します。
- `pulse_width`: 指定した `report_pulse_width` レポートを削除します。
- `slack_histogram`: 指定した `create_slack_histogram` レポートを削除します。
- `timing_path`: 指定した `report_timing` レポートを削除します。
- `timing_summary`: 指定した `report_timing_summary` レポートを削除します。

注記: `-type` オプションのデフォルトは `timing_path` で、`report_timing` コマンドで生成されたレポートが削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 削除するタイミング結果の名前を指定します。

例

次の例では、メモリから指定した結果を削除しています。

```
delete_timing_results -type clock_interaction clkNets
```

関連項目

- [check_timing](#)
- [create_slack_histogram](#)
- [report_bus_skew](#)
- [report_clock_interaction](#)
- [report_cdc](#)
- [report_config_timing](#)
- [report_datasheet](#)
- [report_pulse_width](#)
- [report_timing](#)
- [report_timing_summary](#)

delete_utilization_results

指定した名前の使用率結果をメモリから削除します。

構文

```
delete_utilization_results -name <arg> [-quiet] [-verbose]
```

使用法

名前	説明
-name	削除する結果の名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

指定の結果セットから指定の使用率結果を削除します。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

-name <arg> (必須): 削除する結果の名前を指定します。

例

次の例では、メモリから指定した結果を削除しています。

```
delete_utilization_results -name SS01
```

関連項目

- [report_utilization](#)

delete_waivers

1 つまたは複数の DRC、METHODODOLOGY、CDC メッセージ除外を削除します。

構文

```
delete_waivers [-scoped] [-quiet] [-verbose] [<objects>...]
```

使用法

名前	説明
[-scoped]	オブジェクトのワイルドカードを設定されている現在のインスタンス (current_instance) に限定して解釈するよう指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	DRC、METHODODOLOGY 除外が設定されている除外オブジェクトまたはメッセージ オブジェクト (セル、ネット、サイトなど) を 1 つまたは複数指定します。

カテゴリ

[Waiver \(除外\)](#)、[Object \(オブジェクト\)](#)

説明

report_drc、report_methodology、または report_cdc コマンドを実行すると、デザインで検出された違反または状況に関するメッセージが返されます。これらの違反は、解決しないと先に進めないことがあります。create_waiver コマンドを使用すると、デザインで無視しても問題ない違反または状況を除外オブジェクトとして指定し、デザイン フローを先に進めることができます。

デザインで指定した除外オブジェクトを削除するには、delete_waivers コマンドを使用します。

引数

-scoped (オプション): オブジェクトのワイルドカードを設定されている現在のインスタンス (current_instance) に限定して解釈するよう指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<objects> (オプション): 除外オブジェクトを get_waivers コマンドを使用して指定します。

例

次の例では、デザインのすべての DRC 除外を削除しています。

```
delete_waivers [get_waivers -type DRC]
```

関連項目

- [create_waiver](#)
- [current_instance](#)
- [get_waivers](#)
- [report_cdc](#)
- [report_drc](#)
- [report_methodology](#)

describe

HDL オブジェクト (変数、信号、ワイヤ、またはレジスタ) のタイプおよび宣言情報を表示します。

構文

```
describe [-quiet] [-verbose] <hdl_object>
```

戻り値

選択したオブジェクトの説明

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hdl_object></code>	説明を表示する HDL オブジェクトまたは HDL スコープを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

HDL オブジェクト (変数、信号、ワイヤ、またはレジスタ) のタイプおよび宣言情報を表示し、指定のオブジェクトの HDL ソースのパスおよびファイル情報を示します。



ヒント: `describe` コマンドは、1 つの HDL オブジェクトに対してのみ実行できます。複数の HDL オブジェクトの簡単なレポートを取得する場合は、`report_objects` コマンドを使用してください。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の `wire` または `reg` エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の `real`、`realtime`、`time`、`event` などがあります。HDL 定数には、Verilog のパラメーターおよび `localparam`、VHDL ジェネリックおよび定数が含まれます。

このコマンドを実行すると、指定の HDL オブジェクトの説明が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hdl_object> (必須): 説明を取得する 1 つの HDL オブジェクトを指定します。

注記: オブジェクトは、名前または `get_objects` コマンドで指定します。

例

次の例では、オブジェクトの説明が現在のシミュレーションのスコープによって異なることを示しています。

```
current_scope testbench
/testbench
describe leds_n
    Signal: {leds_n[3:0]}
    Path: {/testbench/leds_n}
    Location: {File "C:/Data/ug937/sim/testbench.v" Line 9}
current_scope dut
/testbench/dut
describe leds_n
    Port(OUT): {LEDS_n[3:0]}
    Path: {/testbench/dut/LEDS_n}
    Location: {File "C:/Data/sources/sinegen_demo.vhd" Line 42}
```

関連項目

- [current_scope](#)
- [get_objects](#)
- [report_objects](#)

detect_hw_sio_links

RX および TX エンドポイント間のリンクを自動的に検出します。リンクを含む新しいリンク グループを作成します。

構文

```
detect_hw_sio_links [-force] [-quiet] [-verbose]
```

戻り値

検出されたリンクの新しいハードウェア SIO リンク グループ

使用法

名前	説明
<code>[-force]</code>	リンクを検出する前に既存のリンクをすべて削除します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

開いているハードウェア ターゲット上で定義されている GT トランスミッターとレシーバー間の既存の通信経路または前に定義された通信経路を自動的に検出します。

このコマンドは、シリアル I/O 解析機能を実行中にボード接続が変更された場合に使用できます。検出アルゴリズムは、変化する送信パターンを使用し、受信されるパターンのリンクを検出することにより、開いているハードウェア ターゲットで GT がどのように接続されているかを判断します。

1 つのハードウェア SIO リンク (hw_sio_link) で一度に使用できるのは IBERT デバッグ コアの個別の GT のトランスミッターまたはレシーバーのみなので、このコマンドでは既存のリンクで使用されている GT はチェックされません。`-force` オプションを使用すると、開いているハードウェア ターゲットをスキャンしてすべての GT をチェックする前に、既存のリンクをすべてクリアできます。

`detect_hw_sio_links` コマンドは検出されたリンクを定義し、それらの新しいリンクに関連付けるリンク グループを作成します。

このコマンドを実行すると、検出されたリンクの数と作成されたハードウェア SIO リンク グループ (hw_sio_linkgroup) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 既存のリンク オブジェクトを削除し、すべての GT を調べてリンクを検出します。これにより、現在のハードウェア コンフィギュレーションがリンク定義に正しく反映されることを確実にできます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、開いているハードウェア ターゲット上の IBERT デバッグ コアを調べ、既存の通信リンクを検出しています。

```
detect_hw_sio_links
```

注記: `-force` オプションが使用されていないので、既存のリンクで使用される GT は調べられません。

関連項目

- [create_hw_sio_link](#)
- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_linkgroups](#)

disconnect_bd_intf_net

インターフェイス ネット (intf_net) の接続を解除します。

構文

```
disconnect_bd_intf_net [-quiet] [-verbose] <intf_net> <objects>...
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<intf_net>	オブジェクトが接続されているインターフェイス ネットを指定します。
<objects>	intf_net から接続を解除するオブジェクトを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

IP インテグレーター サブシステム デザインの 1 つのインターフェイス ネットの接続を指定のオブジェクトから解除します。インターフェイスとは、IP インテグレーター サブシステム デザインで共通の機能を共有する信号をグループ化したものです。

このコマンドでは、IP サブシステム デザインのピンまたはポートから、ネット全体を削除せずに、指定のインターフェイス ネットを解除できます。ネット全体を削除するには、delete_bd_objs コマンドを使用します。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は TCL_ERROR が返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<intf_net> (必須): 指定のオブジェクトから接続を解除する IP サブシステム デザインの 1 つのインターフェイス ネットを指定します。ネットは、名前で指定するか、または `get_bd_intf_nets` コマンドで 1 つのオブジェクトを返すことにより指定します。

<objects> (必須): ネットの接続を解除するインターフェイス ピンまたはポート オブジェクトを指定します。インターフェイス ピンおよびポートは、`get_bd_intf_pins` または `get_bd_intf_ports` コマンドを使用してオブジェクトとして指定する必要があります。名前で指定することはできません。

例

次の例では、IP サブシステム デザインの唯一のインターフェイス ネットの接続を、接続されているすべてのインターフェイス ピンおよびポート オブジェクトから解除します。

```
disconnect_bd_intf_net [get_bd_intf_nets] \
[get_bd_intf_pins -of_objects [get_bd_intf_nets]] \
[get_bd_intf_ports -of_objects [get_bd_intf_nets]]
```

注記: この例では、デザインにはインターフェイス ネットが 1 つしかありません。複数のインターフェイス ネットが指定されると、`disconnect_bd_intf_net` コマンドからエラーが返されます。

関連項目

- [connect_bd_intf_net](#)
- [connect_bd_net](#)
- [delete_bd_objs](#)
- [get_bd_nets](#)
- [get_bd_pins](#)
- [get_bd_ports](#)

disconnect_bd_net

オブジェクトからネットの接続を解除します。

構文

```
disconnect_bd_net [-quiet] [-verbose] <net> <objects>...
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><net></code>	オブジェクトが接続されているネットを指定します。
<code><objects></code>	ネットから接続を解除するオブジェクトを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター サブシステム デザインの 1 つのネットの接続を指定のオブジェクトから解除します。

このコマンドでは、IP サブシステム デザインのピンまたはポートから、ネット全体を削除せずに、指定のネットを解除できます。ネット全体を削除するには、`delete_bd_objs` コマンドを使用します。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は TCL_ERROR が返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<net>` (必須): 指定のオブジェクトから接続を解除する IP サブシステム デザインの 1 つのネットを指定します。ネットは、名前で指定するか、または `get_bd_nets` コマンドで 1 つのオブジェクトを返すことにより指定します。

<objects> (必須): ネットの接続を解除するピンまたはポート オブジェクトを指定します。ピンおよびポートは、`get_bd_pins` または `get_bd_ports` コマンドを使用してオブジェクトとして指定する必要があります。名前で指定することはできません。

例

次の例では、名前で指定したネットの接続を指定したピン オブジェクトから解除しています。

```
disconnect_bd_net /vidout1_locked [get_bd_pins {vidOut1/locked newMod1/t1}]
```

関連項目

- [connect_bd_intf_net](#)
- [connect_bd_net](#)
- [delete_bd_objs](#)
- [disconnect_bd_intf_net](#)
- [get_bd_nets](#)
- [get_bd_pins](#)
- [get_bd_ports](#)

disconnect_debug_port

デバッグ ポート チャンネルからネットとピンの接続を解除します。

構文

```
disconnect_debug_port [-channel_index <arg>] [-quiet] [-verbose] <port>
```

使用法

名前	説明
<code>[-channel_index]</code>	ネット接続を解除するチャンネル インデックスを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><port></code>	デバッグ ポート名を指定します。

カテゴリ

[Debug \(デバッグ\)](#)

説明

デバッグ ポートから信号の接続を解除します。

ネットリスト デザインからの信号は、`connect_debug_port` コマンドを使用して ILA デバッグ コアのポートに接続されます。

ポートは単に接続を解除するだけでなく、`delete_debug_port` コマンドを使用してデバッグ コアから削除することもできます。

デバッグ コアのポート名を取得する必要がある場合は、`get_debug_ports` コマンドを使用するとコアのポートすべてをリストできます。プロジェクト内のコアと特定のパラメーターをすべてリストするには、`report_debug_core` コマンドを使用します。

引数

`-channel_index <value>` (オプション): 接続を解除するポートのチャンネル インデックスを指定します。

注記: `--channel_index` オプションを使用しない場合、ポート全体の接続が解除されます。

`<port>` (必須): 接続を解除するデバッグ コアのポート名を指定します。ポート名は、`core_name/port_name` という形式で指定する必要があります。例を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、myCore の PROBE1 ポートから指定したチャンネル インデックスのみの接続を解除しています。

```
disconnect_debug_port -channel_index 2 myCore/PROBE1
```

次の例のように `-channel_index` を指定しない場合、指定したポートのすべてのチャンネルの接続が解除されます。

```
disconnect_debug_port myCore/PROBE1
```

関連項目

- [connect_debug_port](#)
- [delete_debug_port](#)
- [get_debug_ports](#)
- [report_debug_core](#)

disconnect_hw_server

ハードウェア サーバーへの接続を閉じます。

構文

```
disconnect_hw_server [-quiet] [-verbose] [<hw_server>]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_server>]</code>	ハードウェア サーバーを指定します。デフォルトは、現在のハードウェア サーバーです。

カテゴリ

Hardware (ハードウェア)

説明

Vivado Design Suite から現在または指定の Vivado ツール ハードウェア サーバーの接続を解除します。

現在のハードウェア サーバーは、最後に接続したハードウェア サーバーか、`current_hw_server` コマンドを使用して手動で定義したハードウェア サーバーです。現在のハードウェア サーバー (`hw_server`) の接続を解除した場合は、新しく現在のハードウェア サーバー オブジェクトを定義するまで定義された現在のハードウェア サーバーはありません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_server>` (オプション): Vivado Design Suite から接続を解除する `hw_server` を指定します。ハードウェア サーバーを指定しない場合、現在のハードウェア サーバー (`current_hw_server`) の接続が解除されます。ハードウェア サーバー (`hw_server`) は、`get_hw_servers` または `current_hw_server` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、Vivado Design Suite から指定のハードウェア サーバーの接続を解除しています。

```
disconnect_hw_server [get_hw_server picasso]
```

関連項目

- [connect_hw_server](#)
- [current_hw_server](#)
- [get_hw_servers](#)
- [refresh_hw_server](#)

disconnect_net

ピンまたはポートからネットの接続を解除します。

構文

```
disconnect_net [-prune] [-net <arg>] [-objects <args>] [-pinlist <args>]
               [-quiet] [-verbose]
```

使用法

名前	説明
[-prune]	接続を解除する際に、プリミティブ インスタンス上のネットおよびピンまでのピン/ネット チェーンを、チェーンの各オブジェクトに残っている接続が 1 つのみの場合に削除します。
[-net]	接続を解除するネットを指定します。指定しない場合、最初のピンまたはポート オブジェクトに接続されているネットの接続が解除されます。
[-objects]	接続を解除するピンまたはポートを指定します。
[-pinlist]	接続を解除するピンおよびポート オブジェクトを 1 つまたは複数指定します。オブジェクトを名前で指定できます。-objects ほど柔軟ではありませんが、-objects より高速です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Netlist (ネットリスト)

説明

開いている合成済みデザインまたはインプリメント済みデザインのネットリストに含まれる 1 つ以上のピンまたはポートから、指定のネットの接続を解除します。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-prune` (オプション): 指定のネットの接続を解除したことにより未接続となった階層ピン、ポート、またはネットを削除します。`-objects` で指定されたピンは、接続は解除されますが、削除されることはありません。ただし、指定のピンに接続されたネットおよびピンまたはポートは削除されます。

`-net <arg>` (オプション): 接続を解除するネットを指定します。指定しない場合、最初のピンまたはポートに接続されているネットの接続が解除されます。

注記: `create_net` コマンドの `-bus_from` および `-bus_to` オプションを使用してバスを作成できますが、バスの各ビットを `disconnect_net` コマンドを使用してそれぞれ接続解除する必要があります。

`-objects <args>` (必須): ネットの接続を解除するピンまたはポート オブジェクトを指定します。

`-pinlist <args>` (オプション): 指定のネットから接続を削除するピンおよびポート オブジェクトを 1 つまたは複数指定します。オブジェクトは名前指定できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次のような接続ネットワークがあるとします。

```
leaf_cell1/pin1 > net1 > block1/pin1 >
    topnet
< block2/pin1 < net2 < leaf_cell2/pin1
```

次の例では、まず `block1/pin1` から信号 `topnet` の接続が解除され、`topnet`、`block2/pin1`、および `net2` に `-prune` が適用されますが、`leaf_cell2/pin1` には `-prune` は適用されません。

```
disconnect_net -prune -net topnet -objects [get_pins block1/pin1]
```

注記: `-prune` オプションにより `block1/pin1` は削除されないで、`net2` は削除されません。

次の例では、`dataBus` の指定したビットの接続を解除しています。

```
disconnect_net -net dataBus[1] -objects {dataIN[1] myDMA/data[1]}
```

次の例では、複数のピンまたはポートからネットの接続を解除する最も効率的なコーディング スタイルを示します。最初のコード例では、すべてのピンがピンまたはポートの 1 つのリストで指定されるので、`disconnect_net` コマンドは複数の `disconnect_net` コマンドを実行する 2 つ目のコード例よりも高速に実行されます。

```
disconnect_net -objects [list {pad_rin1_IBUF[0]_inst/O} {hcore0/pad_rin1[9]
[0]} \
{pad_rin1_IBUF[1]_inst/O} {hcore0/pad_rin1[9][1]} \
{pad_rin1_IBUF[2]_inst/O} {hcore0/pad_rin1[9][2]} \
{pad_rin1_IBUF[3]_inst/O} {hcore0/pad_rin1[9][3]} \
{pad_rin1_IBUF[4]_inst/O} {hcore0/pad_rin1[9][4]} \
{pad_rin1_IBUF[5]_inst/O} {hcore0/pad_rin1[9][5]} \
{pad_rin1_IBUF[6]_inst/O} {hcore0/pad_rin1[9][6]} \
{pad_rin1_IBUF[7]_inst/O} {hcore0/pad_rin1[9][7]} \
{pad_rin1_IBUF[8]_inst/O} {hcore0/pad_rin1[9][8]} \
{pad_rin1_IBUF[9]_inst/O} {hcore0/pad_rin1[9][9]} ]
```



ヒント: これらの例では `-net` オプションが指定されていないので、最初のピンまたはポートに接続されているネットの接続がその後のすべてのピンおよびポートから解除されます。

```
disconnect_net -objects [list {pad_rin1_IBUF[0]_inst/O} {hcore0/pad_rin1[9]
[0]}]
disconnect_net -objects [list {pad_rin1_IBUF[1]_inst/O} {hcore0/pad_rin1[9]
[1]}]
disconnect_net -objects [list {pad_rin1_IBUF[2]_inst/O} {hcore0/pad_rin1[9]
[2]}]
disconnect_net -objects [list {pad_rin1_IBUF[3]_inst/O} {hcore0/pad_rin1[9]
[3]}]
disconnect_net -objects [list {pad_rin1_IBUF[4]_inst/O} {hcore0/pad_rin1[9]
[4]}]
disconnect_net -objects [list {pad_rin1_IBUF[5]_inst/O} {hcore0/pad_rin1[9]
[5]}]
disconnect_net -objects [list {pad_rin1_IBUF[6]_inst/O} {hcore0/pad_rin1[9]
[6]}]
disconnect_net -objects [list {pad_rin1_IBUF[7]_inst/O} {hcore0/pad_rin1[9]
[7]}]
disconnect_net -objects [list {pad_rin1_IBUF[8]_inst/O} {hcore0/pad_rin1[9]
[8]}]
disconnect_net -objects [list {pad_rin1_IBUF[9]_inst/O} {hcore0/pad_rin1[9]
[9]}]
```

関連項目

- [connect_net](#)
- [remove_net](#)
- [create_net](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

display_hw_ila_data

ハードウェア ILA データをビューアーに表示します。

構文

```
display_hw_ila_data [-wcfg <arg>] [-reset] [-quiet] [-verbose]  
[<hw_ila_data>...]
```

使用法

名前	説明
[-wcfg]	代替波形設定ファイルをインポートします。
[-reset]	波形設定ファイルをデフォルトの設定にリセットします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_ila_data>]	ハードウェア ILA データ オブジェクトを指定します。デフォルトは、現在のハードウェア ILA データです。

カテゴリ

Hardware (ハードウェア)

説明

このコマンドは、Vivado Design Suite ロジック解析機能のグラフィカル ユーザー インターフェイスでのみ使用できます。Vivado ロジック解析機能の波形設定ウィンドウに指定の ILA デバッグ コア データ オブジェクトを表示します。

ILA デバッグ サンプル データは、upload_hw_ila_data コマンドを使用して動作中のデバイスから取得します。これによりハードウェア ILA データ (hw_ila_data) オブジェクトが作成され、write_hw_ila_data コマンドを使用してファイルとしてディスクに書き込むことができます。このコマンドは、その ILA データ ファイルを読み込みます。

波形ウィンドウでの ILA デバッグ コアの表示特性は、波形設定ファイルで決定されます。波形設定ファイルには、単に表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。

波形設定オブジェクトは、create_wave_config コマンドを使用して Vivado ロジック解析機能内に作成します。波形設定ファイルをディスクに保存するには save_wave_config コマンドを使用し、開くには open_wave_config コマンドを使用します。

open_wave_config コマンドは、波形設定ファイルを開き、現在のシミュレーションのデータ ソースにマップします。

引数

-wcfg <arg> (オプション): ILA データを表示する、save_wave_config コマンドで作成した波形設定ファイルを指定します。-wcfg オプションを指定しない場合は、ILA データは Vivado ロジック解析機能で指定されているデフォルトの波形設定で表示されます。

`-reset` (オプション): 波形ウィンドウをデフォルト設定にリセットし、指定の ILA データを表示します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_ila_data>` (オプション): 表示する `hw_ila_data` オブジェクトを 1 つ以上指定します。`hw_ila_data` は、`get_hw_ila_datas` または `current_hw_ila_data` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア ILA データを指定しない場合、現在のハードウェア ILA データ オブジェクト (`current_hw_ila_data object`) が表示されます。

例

次の例では、`hw_ila_data` ファイルを読み込み、その結果の `hw_ila_data` を Vivado ロジック解析機能の波形ウィンドウに表示しています。

```
read_hw_ila_data C:/hw_ila_file.ila
display_hw_ila_data [get_hw_ila_datas hw_ila_file]
```

関連項目

- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [upload_hw_ila_data](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [read_hw_ila_data](#)
- [run_hw_ila](#)
- [save_wave_config](#)

display_hw_sio_scan

既存のハードウェア SIO スキャンを表示します。

構文

```
display_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_scans></code>	ハードウェア SIO スキャン

カテゴリ

Hardware (ハードウェア)

説明

このコマンドは、Vivado Design Suite シリアル I/O 解析機能のグラフィカル ユーザー インターフェイスでのみ使用できます。Vivado IDE の [Scan Plots] ウィンドウに指定の SIO スキャン データ オブジェクトを表示します。

SIO スキャン データは、`read_hw_sio_scan` コマンドを使用してディスク上のファイルまたは `run_hw_sio_scan` コマンドを使用して作成されるハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトから読み出すことができます。表示されるプロットのタイプは、`hw_sio_scan` オブジェクトのスキャン タイプ (`<scan_type>`) によって決まります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_scans>` (オプション): Vivado IDE のハードウェア マネージャー機能の [Scan Plots] ウィンドウに表示する `hw_sio_scan` オブジェクトを 1 つまたは複数指定します。

例

次の例では、SIO スキャン データ ファイルをメモリに読み込み、作成された hw_sio_scan オブジェクトを表示しています。

```
display_hw_sio_scan [read_hw_sio_scan C:/Data/loopback1.csv]
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [run_hw_sio_scan](#)
- [read_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

encrypt

ファイルを IEEE 1735 の言語特定のキー ファイルで暗号化します。

構文

```
encrypt [-key <arg>] -lang <arg> [-ext <arg>] [-quiet] [-verbose]
<files>...
```

使用法

名前	説明
[-key]	暗号化に使用するキー ファイルを指定します。指定しない場合は、エンベデッド キーが使用されます。
-lang	入力/出力ファイルの HDL 言語を指定します。
[-ext]	暗号化されたファイルの拡張子を指定します。元のソース ファイルは保持されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	暗号化するファイルを指定します。

カテゴリ

FileIO (ファイル入力および出力)

説明



ヒント: encrypt コマンドの使用は制限されており、特別なライセンスが必要です。

暗号化ライセンスを所有している場合、このコマンドを使用して IEEE 1735 暗号化規格を使用して Verilog または VHDL ファイルを暗号化できます。

暗号化されたファイルは、サードパーティ IP ベンダーにより知的所有権を保護しながら、Vivado Design Suite で合成およびシミュレーション用に暗号化されたファイルを読み込むことができるようにするため提供されることがあります。暗号化前は、データは通常のテキストです。



重要: -ext オプションが使用されている場合を除き、指定されたファイルはその場で暗号化され、入力ファイルが暗号化されたファイルで上書きされます。

引数

-key <arg> (オプション): ザイリンクス公開キーを含む RSA キー ファイルを指定します。-key オプションを指定しない場合、指定のファイルに組み込まれたキーが使用されます。これらは 1735 サポートのプラグマまたは指定のファイルに組み込まれた指示子で、暗号キーを提供し、保護データの範囲を示します。



ヒント: ザイリンクス公開キーを入手するには、IEEE P1735 ワーキング グループから入手するか、またはザイリンクス販売代理店にご連絡ください。

`-lang [vhd1 | verilog]` (必須): 暗号化するソース ファイルの HDL 言語を指定します。有効な値は VHDL または verilog です。

`-ext <arg>` (オプション): 出力される暗号化されたファイルに使用する拡張子を指定します。元のソース ファイルは保持されます。



重要: このオプションを使用しない場合、元のソース ファイルは暗号化された出力ファイルで上書きされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): 暗号化する Verilog または VHDL ファイルの名前を指定します。

例

次の例では、指定した Verilog ファイルを指定したキー ファイルを使用して暗号化しています。

```
encrypt -lang verilog -key C:/Data/xilinx_rsa_key.txt C:/Data/design_1.v
```

注記: 指定したソース ファイルは、暗号化された出力ファイルで上書きされます。

関連項目

- [write_verilog](#)
- [write_vhdl](#)

endgroup

グループ単位で実行を取り消し/やり直しできるコマンド シーケンスを終了します。

構文

```
endgroup [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

グループ単位で実行を取り消し/やり直しできるコマンド シーケンスを終了します。コマンド シーケンスを開始するには、`startgroup` コマンドを使用します。

`undo` または `redo` で一度に取り消し/やり直しできるコマンド グループは複数作成できますが、コマンド グループを入れ子にすることはできません。`startgroup` を使用して新しいコマンド シーケンスを作成する前に、`endgroup` を使用してコマンド シーケンスを終了する必要があります。



ヒント: `startgroup/endgroup` コマンドは、`undo` コマンドで一度に取り消しできる、または `redo` コマンドを使用して一度にやり直しできる関連コマンドのシーケンスをサポートするためのものですが、一部の Tcl コマンドは `endgroup` を意図せずにトリガーすることがあり、コマンドによっては `UNDO` または `REDO` コマンドはサポートされません。この制限は完全に定義されていません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、まず `startgroup` を実行し、関連するコマンドのシーケンスを実行して、`endgroup` を実行します。このコマンド シーケンスは、グループ単位で実行を取り消すことができます。

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEng
add_cells_to_pblock pblock_wbArbEngine \
    [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEng \
    [get_cells [list usbEngine1/usbEngineSRAM]] -clear_locs
endgroup
```

関連項目

- [startgroup](#)
- [redo](#)
- [undo](#)

exclude_bd_addr_seg

アドレス空間からセグメントを除外します。

構文

```
exclude_bd_addr_seg [-target_address_space <arg>] [-quiet] [-verbose]  
[<segment_to_exclude>]
```

戻り値

新しく除外セグメント オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
[-target_address_space]	スレーブ セグメントを除外するターゲット アドレス空間を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<segment_to_exclude>]	除外するセグメントを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

スパス接続をサポートし、不要なデバイス リソースを削除するため、指定の AXI ペリフェラル アドレス セグメントを、マップされている AXI マスターによるアクセスから除外します。

このコマンドを使用すると、特定のペリフェラルが特定の AXI マスターによりアクセスされないようにすることができます。たとえば、AXI ペリフェラル P0 および P1 が 2 つのマスター M0 および M1 に接続されている場合、スパス接続を使用して M0 で P0 および P1 の両方にアクセスできるようにし、M1 では P1 にアクセスできるが P0 にはアクセスできないようにすることができます。

IP インテグレーター ブロック デザインでは、AXI ペリフェラルのアドレス セグメントは次の 3 つのいずれかのステートになります。

- Unmapped (マップされていない): AXI ペリフェラル (スレーブ インターフェイス) は、AXI マスターに接続されていますが、ペリフェラルはマスターのアドレス空間のアドレス セグメントに割り当てられておらず、マスターで認識されません。
- Mapped (マップ済み): AXI ペリフェラルは AXI マスターに接続され、アドレス セグメントまたは範囲に割り当てられており、マスターでアクセス可能です。
- Excluded (除外): AXI ペリフェラルは AXI マスターにマップされ、アドレスに割り当てられていますが、マスターでアクセスできません。マスターのアドレス空間の AXI スレーブが割り当てられているアドレス セグメントは、占有されているとみなされます。

アドレス セグメントの除外は、複数のマスターに接続されているペリフェラルへのアクセスを制限することが目的です。ペリフェラル インターフェイスがマスターに接続されており、そのマスターのアドレス セグメントにマップされていない場合、`validate_bd_design` コマンドを実行したときにクリティカル警告が返されますが、ペリフェラルをマップした後に除外すると、AXI マスターとペリフェラルの間 (マルチプレクサーとでコードなど) のアクセスを提供する接続に必要なリソースを削除でき、インプリメント済みデザインでリソースを節約できます。



ヒント: `assign_bd_address` を実行すると、IP インテグレーターでマップされていないアドレス セグメントがアドレス空間にマップされますが、除外されているアドレス空間はマップされません。

このコマンドには、マップ済みアドレス セグメントに対して実行するか、マップされていないアドレス セグメントに対して実行するかによって、2 つの構文があります。

```
exclude_bd_addr_seg <master_addr_seg>
exclude_bd_addr_seg -target_address_space <master_addr_space>
<slave_addr_seg>
```

2 つ目のコマンド構文では、スレーブ セグメントを指定すると、スレーブがまず指定の AXI マスターのアドレス空間に割り当てられてから、マスターによるアクセスから除外されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-target_address_space <arg>` (オプション): アドレス セグメントをマップするターゲット アドレス空間を指定します。このオプションを使用すると、通常 `assign_bd_address` コマンドで実行されるマップ割り当てを実行でき、1 つのコマンドでアドレス セグメントをアドレス空間に割り当て、除外できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<segment_to_exclude>`: AXI マスターによるアクセスから除外する 1 つのアドレス セグメント オブジェクト (`bd_addr_seg`) を指定します。

例

次の例では、AXI ペリフェラル (スレーブ) で定義されたアドレス セグメントを AXI マスターのアドレス空間に割り当て、そのアドレス セグメントをマスターによるアクセスから除外しています。

```
assign_bd_address [get_bd_addr_segs {axi-gpio-1/S_AXI/Reg}]
exclude_bd_addr_seg [get_bd_addr_segs microblaze-1/Data/SEG-axi-gpio-1-Reg]
```

関連項目

- [assign_bd_address](#)
- [create_bd_addr_seg](#)
- [get_bd_addr_segs](#)
- [get_bd_addr_spaces](#)

- [include_bd_addr_seg](#)

execute_hw_svf

current_hw_target で SVF ファイルを実行します。

構文

```
execute_hw_svf [-quiet] [-verbose] <file_name>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file_name>	SVF ファイルの名前を指定します。

カテゴリ

Hardware (ハードウェア)

説明

Vivado ハードウェア マネージャーでは、SVF (Serial Vector Format) ファイルを使用したハードウェア デバイスのプログラムがサポートされています。SVF ファイルは、プログラム命令とコンフィギュレーション データを含む ASCII ファイルです。これらのファイルは、ATE マシンおよびエンベデッド コントローラーでバウンダリスキャン操作を実行するために使用されます。SVF ファイルは、ビットストリームをザイリンクス デバイスに直接プログラム、またはフラッシュ メモリ デバイスに間接プログラムするのに必要な JTAG コマンドを含む ASCII ファイルです。SVF ファイルは、write_hw_svf コマンドを使用して記述し、execute_hw_svf コマンドを使用してデバイスのプログラムに使用します。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

execute_hw_svf コマンドは、SVF コマンドを Vivado Tcl コマンドに変換して指定したターゲットに対して実行します。このプロセスは、SVF ファイルの大きさによって多少時間がかかることがあります。このコマンドを実行するには、SVF ファイルで指定されているデバイス チェーンに一致する JTAG チェーンを含む、開いている現在の hw_target オブジェクトが必要です。



ヒント: execute_hw_svf コマンドは汎用 SVF リーダーではなく、Vivado ツールで記述された SVF ファイルを読み出して実行する場合にのみ使用してください。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、`execute_hw_svf` コマンドからのすべてのメッセージを返します。このオプションは、このファイルの実行中に Vivado ツールで実行される擬似 SVF コマンドをすべて表示するので、SVF ファイルの実行に関する問題をデバッグする場合に使用します。

<file_name>: 実行する SVF ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定の SVF コマンドを詳細モードで実行し、実行されたコマンドすべてを表示しています。

```
open_hw_target {houdini26:3121/xilinx_tcf/Digilent/210203327996A}
execute_hw_svf -verbose C:/Data/k7_design.svf
```

関連項目

- [create_hw_device](#)
- [create_hw_target](#)
- [current_hw_target](#)
- [current_project](#)
- [get_hw_probes](#)
- [open_hw_target](#)
- [program_hw_devices](#)
- [write_hw_svf](#)

export_as_example_design

現在のデザインをスタティック サンプル デザインとしてエクスポートします。

構文

```
export_as_example_design -vlnv <arg> [-no_ip_version] [-force] [-quiet]  
[-verbose] -directory <arg>
```

使用法

名前	説明
-vlnv	生成するサンプル デザインの VLNv を指定します。
[-no_ip_version]	create_bd_cell コマンドの IP VLNv の一部として IP バージョンを含めないことを指定します。
[-force]	ディレクトリが存在しない場合にディレクトリを作成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
-directory	サンプル デザインを生成するディレクトリを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

export_bd_synth

サードパーティ合成ツールで使用するためのブロック デバイス (BD) のデザイン チェックポイントおよびスタブ ファイルを作成します。必要に応じて合成を実行します。

構文

```
export_bd_synth [-force] [-keep] [-verbose] [-quiet] <file>
```

戻り値

なし (正常に実行されなかった場合はエラー)

使用法

名前	説明
<code>[-force]</code>	既存のデザイン チェックポイントおよびスタブ ファイルを上書きします。
<code>[-keep]</code>	一時ディレクトリおよびプロジェクトを保持します。
<code>[-verbose]</code>	詳細なメッセージを表示します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code><file></code>	合成済みチェックポイントを保存するブロック デザイン ファイルを指定します。

カテゴリ

[Synthesis \(合成\)](#)、[xilinx_tclstore \(ザイリンクス Tcl Store\)](#)、[projutils \(プロジェクト ユーティリティ\)](#)

説明

ブロック デザイン (.bd) を合成し、デザインに必要なサブデザイン (アウト オブ コンテキスト合成された IP など) と統合して、合成済みデザイン全体に対して 1 つのデザイン チェックポイント (.dcp) およびほかの合成ツールで使用するための HDL スタブ ファイルを生成します。出力ファイルは、ソース BD ファイルと同じディレクトリに配置されます。

引数

`-force` (オプション): 指定したデザイン チェックポイントおよびスタブ ファイルが存在する場合に上書きします。

`-keep` (オプション): `export_bd_synth` コマンドの完了後に一時ディレクトリおよびプロジェクトを保持します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 合成済みチェックポイントをエクスポートするブロック デザイン ファイル (.bd) を指定します。

例

次の例では、指定したブロック デザインの合成済みチェックポイントとスタブ ファイルを生成しています。

```
export_bd_synth [get_files block_1.bd]
```

関連項目

- [get_files](#)

export_ip_user_files

プロジェクトから IP/IP インテグレーター ユーザー ファイルを生成およびエクスポートします。1 つまたは複数の IP で実行されるよう適用範囲を設定できます。

構文

```
export_ip_user_files [-of_objects <arg>] [-ip_user_files_dir <arg>]
  [-ipstatic_source_dir <arg>] [-lib_map_path <arg>] [-no_script] [-sync]
  [-reset] [-force] [-quiet] [-verbose]
```

戻り値

エクスポートされたファイルのリスト

使用法

名前	説明
[-of_objects]	ファイルをエクスポートする IP、IP インテグレーター、またはファイルセット オブジェクトを指定します。
[-ip_user_files_dir]	スタティク、ダイナミック、ラッパー、ネットリスト、スクリプト、および MEM ファイルのシミュレーション ベース ディレクトリへのパスを指定します。デフォルトはありません。
[-ipstatic_source_dir]	IP スタティク ファイルへのディレクトリ パスを指定します。
[-lib_map_path]	コンパイルされたシミュレーション ライブラリのディレクトリ パスを指定します。
[-no_script]	シミュレーション スクリプトをエクスポートしません。デフォルトは 1 です。
[-sync]	IP/IP インテグレーターのダイナミック ファイルおよびシミュレーション スクリプト ファイルを削除します。
[-reset]	IP/IP インテグレーターのスタティク、ダイナミック、およびシミュレーション スクリプト ファイルをすべて削除します。
[-force]	既存のファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)、[xilinx Tcl Store \(ザイリンクス Tcl Store\)](#)

説明

スタティク、ダイナミック、ネットリスト、verilog/vhdl スタブ、メモリ初期化ファイルを使用して IP ユーザー ファイルのリポジトリをエクスポートします。

引数

-of_objects <arg> (オプション): IP のスタティク ファイルおよびダイナミック ファイルをエクスポートする必要があるターゲット オブジェクト、IP、ブロック デザイン (.bd) またはファイルセットを指定します。

`-ip_user_files_dir <arg>` (オプション): ダイナミック ファイルおよび IP のその他のスタティックでないファイルの IP ユーザー ファイルのベース ディレクトリ パスを指定します。このオプションを使用しない場合、デフォルトでは `IP.USER_FILES_DIR` プロジェクト プロパティで指定されているパスが使用されます。

`-ipstatic_source_dir <arg>` (オプション): スタティック IP ファイルへのディレクトリ パスを指定します。このオプションを使用しない場合、デフォルトでは `SIM.IPSTATIC_SOURCE_DIR` プロジェクト プロパティで指定されているパスが使用されます。

注記: `-ip_user_files_dir` オプションを指定すると、デフォルトで IP スタティック ファイルが `ipstatic` という名前の下位ディレクトリにエクスポートされます。このオプションも指定すると、IP スタティック ファイルがこのオプションで指定したパスにエクスポートされます。

`-lib_map_path <arg>` (オプション): コンパイル済みのシミュレーション ライブラリのディレクトリ パスを指定します。

`-no_script` (オプション): シミュレーション スクリプトをエクスポートしません。

`-sync` (オプション): IP/IP インテグレーターのダイナミック ファイルおよびシミュレーション スクリプト ファイルを削除します。

`-reset` (オプション): IP/IP インテグレーターのスタティック、ダイナミック、およびシミュレーション スクリプト ファイルをすべて削除します。

`-force` (オプション): 既存のファイルを上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`char_fifo` IP のダイナミック ファイルを `<project>/<project>.ip_user_files/ip/char_fifo` ディレクトリに、スタティック ファイルを `<project>/<project>.ip_user_files/ipstatic` ディレクトリにエクスポートしています。

```
export_ip_user_files -of_objects [get_ips char_fifo]
```

関連項目

- [get_files](#)
- [get_ips](#)

export_simulation

指定したシミュレータを使用してスタンドアロン シミュレーションを実行するためのスクリプトと関連のデータ ファイルをエクスポートします。

構文

```
export_simulation [-simulator <arg>] [-of_objects <arg>]
  [-ip_user_files_dir <arg>] [-ipstatic_source_dir <arg>]
  [-lib_map_path <arg>] [-script_name <arg>] [-directory <arg>]
  [-runtime <arg>] [-define <arg>] [-generic <arg>] [-include <arg>]
  [-use_ip_compiled_libs] [-absolute_path] [-export_source_files]
  [-32bit] [-force] [-quiet] [-verbose]
```

戻り値

なし

使用法

名前	説明
[-simulator]	シミュレーション スクリプトを作成するシミュレータを指定します。有効な値は all、xsim、modelsim、questa、ies、xcelium、vcs、riviera、activehdl で、デフォルトは all です。
[-of_objects]	指定のオブジェクトのシミュレーション スクリプトをエクスポートします。デフォルトはありません。
[-ip_user_files_dir]	スタティック、ダイナミック、およびデータ ファイルのエクスポートされた IP/BD (ブロック デザイン) へのディレクトリ パスを指定します。
[-ipstatic_source_dir]	エクスポートされた IP/BD スタティック ファイルへのディレクトリ パスを指定します。
[-lib_map_path]	コンパイル済みのシミュレーション ライブラリのディレクトリ パスを指定します。指定しない場合は、生成されたスクリプトのヘッダーの手順に従い、シミュレーション ライブラリのマップ情報を手動で供給してください。デフォルトはありません。
[-script_name]	出力スクリプトのファイル名を指定します。指定しない場合は、デフォルト名のファイルが生成されます。デフォルトは top_module.sh です。
[-directory]	生成したシミュレーション スクリプトの保存先ディレクトリを指定します。デフォルトは export_sim です。
[-runtime]	シミュレーションを実行する時間を指定します。デフォルトでは、シミュレーションが完全に実行されるか、論理的なブレイクまたは終了条件に達するまで実行されます。
[-define]	このオプションで指定したリストから Verilog 定義を読み出します。
[-generic]	このオプションで指定したリストから VHDL ジェネリックを読み出します。
[-include]	このオプションで指定したリストからインクルード ディレクトリ パスを読み出します。

名前	説明
<code>[-use_ip_compiled_libs]</code>	コンパイル中にコンパイル済みの IP スタティック ライブラリを参照します。このオプションには、コンパイル済み IP ライブラリを使用してスクリプトを生成するため、 <code>-ip_user_files_dir</code> および <code>-ipstatic_source_dir</code> オプションが必要です。
<code>[-absolute_path]</code>	すべてのファイル パスを絶対パスにします。
<code>[-export_source_files]</code>	IP/BD デザイン ファイルを出力ディレクトリにコピーします。
<code>[-32bit]</code>	32 ビット コンパイルを実行します。
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)、[xilinx_tclstore \(ザイリンクス Tcl Store\)](#)

説明

ターゲット シミュレータのシミュレーション スクリプトをエクスポートします。現在のところ、Cadence 社 Incisive Enterprise Simulator (`ies`) および Synopsys 社 VCS MX シミュレータ (`vcs_mx`) がサポートされています。生成されたスクリプトには、デザインをコンパイル、エラボレート、シミュレーションするためのシミュレータ コマンドが含まれます。

このコマンドは、指定のオブジェクトのシミュレーション コンパイル順を取得し、この情報をターゲット シミュレータのコンパイラ コマンドおよびデフォルト オプションと共にテキスト ファイルにエクスポートします。オブジェクトとしては、シミュレーション ファイルセットまたは IP を指定できます。オブジェクトを指定しない場合、`export_simulation` コマンドでシミュレーション最上位に対するスクリプトが生成されます。

Verilog のインクルード ディレクトリまたは Verilog の `define` 文を含むファイルへのパスは、コンパイラ コマンドラインに追加されます。

デフォルトでは、コンパイラ コマンドラインのデザイン ソース ファイルおよびインクルード ディレクトリ パスは、生成されるスクリプトで定義される `reference_dir` 変数に相対的に設定されます。これらのパスを絶対パスにするには、`-absolute_path` オプションを使用します。

このコマンドでは、ファイルセットからのデータ ファイル (存在する場合) も出力ディレクトリにコピーされます。デザインに Verilog ソースが含まれる場合、生成されるスクリプトは `gbl.v` もソフトウェア インストール パスから出力ディレクトリにコピーします。

ターゲット シミュレータのデフォルト `.do` ファイルが出力ディレクトリに作成され、スクリプトのコンパイラ コマンドで参照されます。

注記: 生成されたスクリプトでシミュレーションを実行するには、その前に `compile_simlib` コマンドでこのスクリプトを生成するときに指定したコンパイル済みライブラリ ディレクトリ パスを指定してシミュレーション ライブラリをコンパイルする必要があります。生成されたシミュレーション スクリプトには、コンパイル済みライブラリ ディレクトリからのターゲット シミュレータのセットアップ ファイルが含まれます。

このコマンドを実行しても、何も返されません。

引数

`-of_objects <arg>` (オプション): シミュレーション スクリプト ファイルを生成する必要があるターゲット オブジェクトを指定します。ターゲット オブジェクトとしては、シミュレーション ファイルセット (simset) または IP コアを指定できます。このオプションを使用しない場合、現在のシミュレーション ファイルセットに対するシミュレーション スクリプトが生成されます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-lib_map_path <arg>` (オプション): ザイリンクス シミュレーション ライブラリがコンパイルされた、コンパイル済みシミュレーション ライブラリのディレクトリ パスを指定します。詳細は、生成されたスクリプトのヘッダー セクションを参照してください。

`-script_name <arg>` (オプション): シェル スクリプトの名前を指定します。このオプションを指定しない場合、`-of_objects` で指定されたオブジェクト タイプに基づいて、次の形式の名前が使用されます。

- `<simulation_top_name>_sim_<simulator>.sh`
- `<ip_name>_sim_<simulator>.sh`

`-absolute_path` (オプション): スクリプトで使用されるソースおよびインクルード ディレクトリ パスを絶対パスにします。デフォルトでは、すべてのパスは `-dir` オプションで指定されたディレクトリを基準とする相対パスで記述されます。シミュレーション スクリプトに、`-directory` オプションで指定されたディレクトリ パスへの `reference_dir` 変数が設定されます。

`-32bit` (オプション): 32 ビット シミュレーションを実行するよう指定します。このオプションを指定しない場合、デフォルトの 64 ビット オプションがシミュレーション コマンド ラインに追加されます。

`-force` (オプション): 指定したスクリプト ファイルが存在する場合に上書きします。スクリプト ファイルが存在する場合に `-force` が指定されていないと、エラー メッセージが表示されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`-dir <arg>` (必須): シミュレーション ファイルをエクスポートするディレクトリ パスを指定します。

`-simulator [ies | vcs_mx]` (必須): シミュレーション スクリプトのターゲット シミュレータを指定します。有効な値は `ies` または `vcs_mx` です。

例

次の例では、IES シミュレータ用のシミュレーション スクリプト ファイルを現在のディレクトリに生成しています。

```
export_simulation -simulator ies -directory .
```

次の例では、現在のディレクトリにある既存のスクリプト ファイルを上書きしています。

```
export_simulation -force -simulator ies -directory .
```

次の例では、./test_sim ディレクトリに test_ies.sh という名前のシミュレーション スクリプト ファイルを生成しています。

```
export_simulation -simulator ies -directory ./test_sim \
  -script_name test_ies.sh
```

次の例では、シミュレーションの最上位を top_tb に設定してプロジェクト用に top_tb_sim_ies.sh という名前のシミュレーション スクリプト ファイルを ./test_sim ディレクトリに生成しています。このコマンドでは、データ ファイル (.mif、.coe、.dat など) も ./test_sim ディレクトリにコピーされます。

```
export_simulation -simulator ies -directory ./test_sim
```

次の例では、accum_0 IP に対して accum_0_sim_ies.sh という IES シミュレータ用のスクリプト ファイルを指定の出力ディレクトリに生成しています。

```
export_simulation -of_objects [get_files accum_0.xci] \
  -simulator ies -directory test_sim
```

次の例では、accum_0 IP に対して accum_0_sim_vcs_mx.sh という VCS_MX シミュレータ用のスクリプト ファイルを指定の出力ディレクトリに生成しています。

```
export_simulation -of_objects [get_ips accum_0] -simulator vcs_mx \
  -directory test_sim
```

次の例では、最上位が fifo_tb に設定されているシミュレーション セット sim_fifo_test に対して fifo_tb_sim_vcs_mx.sh という IES シミュレータ用のスクリプト ファイルを指定の出力ディレクトリに生成しています。

```
export_simulation -of_objects [get_filesets sim_fifo_test] \
  -simulator ies -directory test_sim
```

次の例では、32 ビット バージョンのシミュレータ コンパイラ用にデザイン ソース ファイルをコンパイルして、VCS_MX シミュレータ用の top_tb_sim_vcs_mx.sh というスクリプト ファイルをエクスポートしています。64 ビット オプションはコマンド ラインに追加されません。

```
export_simulation -force -32bit -simulator vcs_mx -directory
test_bft_vcs_mx
```

次の例では、IES シミュレータ用にコンパイルされたライブラリを参照するため、/sim_libs/ius/lin64/lib/cds.lib ファイルに ./test_sim/cds.lib ファイル パスを含めます (INCLUDE /sim_libs/ius/lin64/lib/cds.lib)。

```
export_simulation -lib_map_path "/sim_libs/ius/lin64/lib" \
  -simulator ies -directory "test_sim"
```

次の例では、VCS_MX シミュレータ用にコンパイルされたライブラリを参照するため、./test_sim/synopsys_sim.setup ファイルに /sim_libs/vcs/lin64/lib/synopsys_sim.setup ファイル パスを含めます (OTHERS=/sim_libs/vcs/lin64/lib/synopsys_sim.setup)。

```
export_simulation -lib_map_path "/sim_libs/vcs/lin64/lib" \
  -simulator vcs_mx -directory "test_sim"
```

次の例では、`./test_sim/ies` ディレクトリにスクリプト ファイルを生成し、IES シミュレータでコンパイル、エラボレート、およびシミュレーションしています。

```
export_simulation -lib_map_path "/sim_libs/ies/lin64/lib" \  
-simulator ies -directory "./test_sim/ies"  
cd test_sim/ies  
./top_tb_sim_ies.sh
```

次の例では、`./test_sim/vcs_mx` ディレクトリにスクリプト ファイルを生成し、VCS_MX シミュレータでコンパイル、エラボレート、およびシミュレーションしています。

```
export_simulation -lib_map_path "/sim_libs/vcs/lin64/lib" \  
-simulator vcs_mx -directory "./test_sim/vcs_mx"  
cd test_sim/vcs_mx  
./top_tb_sim_vcs_mx.sh
```

関連項目

- [compile_simlib](#)
- [current_fileset](#)
- [get_files](#)
- [get_ips](#)

extract_files

コア コンテナからファイルをディスクに抽出します。

構文

```
extract_files [-base_dir <arg>] [-force] [-no_ip_dir] [-no_paths] [-quiet]  
             [-verbose] <files>...
```

戻り値

新しいパスで抽出されたファイルのリスト

使用法

名前	説明
[-base_dir]	抽出されたファイルのベース ディレクトリを指定します。デフォルトは ip_files です。
[-force]	既存のファイルを上書きします。
[-no_ip_dir]	IP ディレクトリを抽出ディレクトリに含めないよう指定します。
[-no_paths]	ファイルを抽出する際にディレクトリを含めないよう指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	抽出するファイルの名前を指定します。

カテゴリ

IPFlow (IP フロー)

説明

コア コンテナ フォーマットの IP からファイルを抽出します。

IP のコア コンテナ フォーマットは圧縮された ZIP ファイルで、デザインのファイル構造を削減し、ツール パフォーマンスを向上します。

このコマンドを実行すると、コア コンテナ IP から抽出されたファイルのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-base_dir <arg> (オプション): 抽出されたファイルを保存するディレクトリを指定します。デフォルトでは、extract_files コマンドで現在の作業ディレクトリの ip_files というフォルダーにファイルが保存されます。

-force (オプション): 同じ名前のファイルが存在する場合に上書きします。

`-no_ip_dir` (オプション): IP サブフォルダーを抽出されたファイルのパスに含めないよう指定します。この場合、ファイルは指定されたディレクトリにコア コンテナー IP の名前のサブフォルダーなしで抽出されます。

`-no_paths` (オプション): コア コンテナーの IP サブフォルダーを抽出されたファイルに含めないよう指定します。このオプションを使用すると、すべてのファイルが最上位 `ip_files` フォルダーまたは `-base_dir` オプションで指定したフォルダーに抽出されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): ファイルを抽出する IP コア コンテナー (`.XCIX`) ファイルの名前を指定します。

例

次の例では、指定のコア コンテナー フォーマットの IP からファイルを指定のベース ディレクトリに抽出しています。

```
extract_files -base_dir C:/Data [get_files char_fifo.xcix]
```

関連項目

- [convert_ips](#)
- [create_ip](#)
- [get_files](#)
- [get_ips](#)

filter

オブジェクトのリストにフィルターを適用し、新しいリストを作成します。

構文

```
filter [-regexp] [-nocase] [-quiet] [-verbose] [<objects>] [<filter>]
```

戻り値

新しいリスト

使用法

名前	説明
<code>[-regexp]</code>	<code>=~</code> および <code>!=~</code> 演算子で正規表現を使用します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<objects>]</code>	フィルターを適用するオブジェクトのリストを指定します。
<code>[<filter>]</code>	式を使用してリストをフィルター処理します。

カテゴリ

Object (オブジェクト)、PropertyAndParameter (プロパティおよびパラメーター)、XDC

説明

オブジェクトのリストから、指定したフィルター検索パターンに一致するオブジェクトのリストを返します。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に `['*']` を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): フィルターを適用するオブジェクトのリストを指定します。オブジェクトのリストを指定するには、`get_parts` などの `get_*` コマンドのいずれかを使用できます。

`<filter>` (オプション): フィルター処理に使用する検索パターンを指定します。指定した検索パターンにより、オブジェクトのプロパティ 値に基づいたオブジェクトのリストがフィルター処理されます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。part オブジェクトの場合、結果をフィルター処理できるプロパティには DEVICE、FAMILY、SPEED などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

例

次の例では、指定したスピード グレードでパーツのリストがフィルター処理されます。

```
filter [get_parts] {speed == -3}
```

次の例では、スピード グレード -3 または -2 でパーツのリストがフィルター処理されます。いずれかのスピード グレードに一致するすべてのパーツが返されます。

```
filter [get_parts] {speed == -3 || speed == -2}
```

次の例では正規表現を使用し、デザインに含まれる VStatus ポートで、0 文字以上の文字列と角カッコ内に 0～9 が 1 回以上現れるものを検索しています。

```
filter -regexp [get_ports] {NAME =~ VStatus.*\[[0-9]+\]}
```

関連項目

- [get_parts](#)
- [get_ports](#)

find_bd_objs

指定のオブジェクトと指定の関係を持つピン、ポート、またはインターフェイスを検索します。

構文

```
find_bd_objs -relation <arg> [-boundary_type <arg>] [-thru_hier]
              [-stop_at_interconnect] [-end_type <arg>] [-quiet] [-verbose]
              <objects>...
```

戻り値

ピン、ポート、またはインターフェイス オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
-relation	指定のオブジェクトとの関連を指定します。有効な値は、connected_to、addressable_slave、addressing_master です。connected_to に設定すると、指定のソース オブジェクトに接続されているピン、ポート、またはインターフェイスが階層の境界を超えて検索されます。
[-boundary_type]	ソース オブジェクトが階層ブロックのピンまたはインターフェイス ピン上にある場合に使用します。有効な値は、空の文字列 (同じレベルの場合、デフォルト)、lower、および all です。lower に設定すると、階層内から検索されます。このオプションは、-relation が connected_to に設定されている場合のみ有効です。
[-thru_hier]	階層ブロックの境界を無視します。-boundary_type が lower に設定されている場合、このオプションは親階層ブロック内の階層ブロックにのみ適用されます。
[-stop_at_interconnect]	-thru_hier オプションを使用している場合に、AXI インターコネクトの境界で検索を停止します。
[-end_type]	ピンまたはポート オブジェクト、およびバス インターフェイス ピンまたはポート オブジェクトに対してのみ使用可能です。ピン/ポートの場合: デフォルトでは、ソース オブジェクトのシンク オブジェクトと、シンク オブジェクトのソース オブジェクトが返されます。シンク オブジェクトに対してこのオプションが all に指定されている場合、ソース オブジェクトとそのソース オブジェクトに接続されているその他のシンク オブジェクトが返されます。バス インターフェイス ピン/ポートの場合: デフォルトでは、モニターでないインターフェイスである 端接続を返します。monitor に設定すると、モニター インターフェイスのみが返されます。all に設定すると、端接続およびモニター インターフェイスの両方が返されます。このオプションは、-relation が connected_to に設定されている場合のみ有効です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	検索の元となる 1 つまたは複数のソース オブジェクトを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

指定のオブジェクトと指定の関係を持つピン、ポート、またはインターフェイスを検索します。

このコマンドを実行すると、ピン、ポート、またはインターフェイス オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-relation <arg>` (必須): 指定のオブジェクトと検索するオブジェクトとの関係を指定します。有効な値は次のとおりです。

- `connected_to`: 指定のソース オブジェクトに接続されているピン、ポート、またはインターフェイスを検索します。検索は、指定のオブジェクトと同じ階層レベル内で実行されます。階層の境界を越えて検索を実行するには、`-thru_hier` オプションを使用します。
- `addressable_slave`: 指定の検索オブジェクトによりスレーブとしてアドレス指定可能なピン、ポート、またはインターフェイスを検索します。
- `addressing_master`: 指定の検索オブジェクトをマスターとしてアドレス指定するピン、ポート、またはインターフェイスを検索します。

`-boundary_type [upper | lower | all]` (オプション): 指定のソース オブジェクトが階層モジュールのピンまたはインターフェイス ピンを含む場合に使用します。デフォルトは `upper` で、関連のオブジェクトがピンが検出された階層モジュールと同じレベルで検索されます。`lower` に設定すると、検索はピンがある階層モジュール内で実行されます。`all` に設定すると、階層モジュールのレベルと階層モジュール内で関連オブジェクトが検索されます。

注記: このオプションは、`-relation connected_to` オプションと共に使用する必要があります。

`-thru_hier` (オプション): オブジェクトの検索を実行する際に階層ブロックの境界を無視します。`-boundary_type lower` と共に使用した場合、関連オブジェクトの検索はピンがある階層モジュール内から開始し、その下位階層すべてが検索されます。

`-stop_at_interconnect` (オプション): `-thru_hier` を使用する場合に、検索を AXI インターコネクトの境界で停止します。ほかの階層ブロックは検索されます。

`-end_type all` (オプション): このオプションは、`<objects>` のタイプがブロック デザインのピン (`bd_pin`) およびブロック デザインのポート (`bd_port`) の場合に指定できます。デフォルトでは、指定のソース オブジェクトに接続されているすべてのシンク ピンと、シンク オブジェクトのソース オブジェクトが返されます。シンク オブジェクトに対して `-end_type all` を指定すると、指定のシンク オブジェクトのソース ピンまたはポートと、そのソース オブジェクトに接続されているその他のシンク オブジェクトが返されます。

注記: このオプションは、`-relation connected_to` オプションと共に使用する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 関連オブジェクトを検索するのに使用する 1 つまたは複数のオブジェクトを指定します。

例

次の例では、ブロック デザイン階層のすべてのレベルで指定したオブジェクトに接続されているピン、ポート、およびインターフェイスを検索しています。

```
find_bd_objs -relation connected_to -thru_hier \  
[get_bd_pins /proc_sys_reset_1/peripheral_aresetn]
```

関連項目

- [create_bd_cell](#)
- [create_bd_intf_net](#)
- [create_bd_intf_pin](#)
- [create_bd_intf_port](#)
- [create_bd_net](#)
- [create_bd_pin](#)
- [create_bd_port](#)

find_routing_path

2 つのノード間の配線パスを検索します。

構文

```
find_routing_path [-allow_overlap] [-max_nodes <arg>] [-min_nodes <arg>]
                  [-from <args>] [-to <args>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-allow_overlap]</code>	ソリューションに既存の配線を使用するノードが含まれるのを許容します。
<code>[-max_nodes]</code>	ソリューションに許容される最大ノード数 (始点ノードと終点ノードを含む) を指定します。デフォルトは 100 です。
<code>[-min_nodes]</code>	ソリューションに許容される最小ノード数 (始点ノードと終点ノードを含む) を指定します。デフォルトは 2 です。
<code>[-from]</code>	配線パスの始点を指定します。
<code>[-to]</code>	配線パスの終点を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)

説明

インプリメント済みデザインに含まれる未配線または部分的に配線されたネット上の 2 つのノード間の配線ソリューションを検索します。

このコマンドは、ネットの FIXED_ROUTE プロパティを割り当てるために配線パスを定義するのに使用できます。FIXED_ROUTE プロパティは後で再利用できるよう XDC ファイルに保存できます。手動配線および FIXED_ROUTE プロパティの使用例は、『Vivado Design Suite チュートリアル: インプリメンテーション』(UG986) を参照してください。

配線パスの始点と終点のノードを定義する必要があり、配線パスに使用できる始点および終点を含めたノードの最大数と最小数を指定できます。ノードは、`get_nodes` コマンドを使用してオブジェクトとして指定する必要があります。未配線のネット オブジェクトでは、ノードがネットに割り当てられていないので、ネットの BEL ピン (`bel_pin`) またはサイト ピン (`site_pin`) への関連性を使用してノードを検索できます。

- ネット > BEL ピン > BEL > タイル > ノード
- ネット > サイト ピン > タイル > ノード

部分的に配線されたネットでは、ネットへの直接の関連性を使用してノードを検索できます。これらのオブジェクトの関連性は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) を参照してください。

`find_routing_path` コマンドを実行すると、始点から終点までで検索される配線パスを表すノードのリストが返されるか、コマンドが正常に実行されたが結果がない場合は「no path found」というメッセージが返され、コマンドが正常に実行されなかった場合はエラーが返されます。

返されたノードのリストを、下の例に示すように `set_property` コマンドを使用して `FIXED_ROUTE` プロパティに割り当てることができます。



ヒント: `report_property` コマンドでは、`FIXED_ROUTE` プロパティの文字列は返されません。 `get_property` コマンドを使用してください。

引数

`-allow_overlap` (オプション): 制限の緩い配線をイネーブルにします。これにより競合が発生する可能性があるため、後で解決する必要があります。重なった配線は、クリーンアップして配線競合を削除する必要があります。配線競合は、`report_route_status` コマンドを使用して特定できます。

`-max_nodes <arg>` (オプション): 配線パスに使用できる `-from` ノードおよび `-to` ノードを含むノードの最大数を指定します。デフォルトは 100 です。

`-min_nodes <arg>` (オプション): 配線パスに使用できる `-from` ノードおよび `-to` ノードを含むノードの最小数を指定します。デフォルトは 2 で、2 以上の値を指定する必要があります。



ヒント: このオプションにより、曲がりくねった配線が生成され、タイミング遅延が追加されることがあります。

`-from <arg>` (オプション): 配線パスの開始ノードを指定します。ノードは、`get_nodes` コマンドを使用してオブジェクトとして指定する必要があります。

`-to <arg>` (オプション): 配線パスの終了ノードを指定します。ノードは、`get_nodes` コマンドを使用してオブジェクトとして指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`-from` オプションで始点、`-to` オプションで終点を指定して指定のネット配線パスを検索し、そのパスを指定の Tcl 変数に割り当てた後、その Tcl 変数を使用してパスをネットの `FIXED_ROUTE` プロパティに割り当てています。

```
set fndPath [find_routing_path -from [lindex [get_nodes -of \
[get_site_pins -of [get_nets wbOutputData_OBUF[14]]] 0] -to \
[lindex [get_nodes -of [get_site_pins -of \
[get_nets wbOutputData_OBUF[14]]] 1]]
set_property FIXED_ROUTE $fndPath [get_nets wbOutputData_OBUF[14]]
```

関連項目

- [get_bel_pins](#)
- [get_nets](#)
- [get_nodes](#)
- [get_property](#)
- [get_site_pins](#)
- [report_property](#)
- [report_route_status](#)
- [set_property](#)

find_top

供給されているファイル、ファイルセット、またはアクティブ ファイルセットから最上位モジュールの候補を検索します。ランク付けされた候補のリストを返します。

構文

```
find_top [-fileset <arg>] [-files <args>] [-return_file_paths] [-quiet]
        [-verbose]
```

使用法

名前	説明
[-fileset]	最上位モジュールの候補を検索するファイルセットを指定します。
[-files]	最上位モジュールの候補を検索するファイルを指定します。
[-return_file_paths]	返される各最上位にファイル パスも含めます。返される値は、最上位名とファイル パスが交互に並べられた文字列のリスト 1 つになります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のファイルセットで定義されているファイル、指定したファイルセットで定義されているファイル、または指定したファイルのリストから、最上位モジュールの候補を検索します。

このコマンドの戻り値は、デザインの最上位として最適な候補から順にモジュールをリストします。lindex コマンドを使用してインデックス 0 を指定すると、最適な候補を最上位モジュールとして選択できます。

引数

-fileset <arg> (オプション): 最上位モジュールの候補を検索するシミュレーションまたはソース ファイルセットを指定します。デフォルトでは、現在のデザインの現在のファイルセットが検索されます。

-files <arg> (オプション): 最上位モジュールの候補を検索するファイルを指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、合成用に現在のデザインの最上位モジュールの最適候補を選択しています。

```
synth_design -top [lindex [find_top] 0]
```

注記: `find_top` では複数の候補が返されますが、インデックス 0 を指定すると合成に最適な候補が選択されます。

次の例では、指定のファイルのリストから最上位モジュールの最適な候補が返されます。

```
find_top -files [get_files -filter {NAME =~ *or1200*}]
```

次の例では、`find_top` コマンドの結果を `$topVar` に設定し、`set_property` コマンドでこの変数を使用して現在のファイルセットの差上位モジュールを定義しています。

```
set topVar [ find_top -files [get_files -filter {NAME =~ *usb*} ] ]
set_property top $topVar [current_fileset]
```

関連項目

- [set_property](#)
- [synth_design](#)

flush_vcd

VCD シミュレーション出力を VCD 出力ファイルに保存します (\$dumpflush Verilog システム タスクと同等)。

構文

```
flush_vcd [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在メモリにある HDL 信号情報を VCD (Value Change Dump) ファイルに記述します。

VCD は、ヘッダー情報、変数定義、HDL 信号値の変更の詳細を含む ASCII 形式のファイルです。VCD ファイルを使用すると、VCD ビューアーでシミュレーション 結果を表示したり、デザインの消費電力見積もりを実行したりできます。

注記: `flush_vcd` コマンドを使用する前に、`open_vcd` コマンドを実行して VCD ファイルを開く必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、VCD バッファの内容を現在の VCD ファイルに記述しています。

```
flush_vcd
```

関連項目

- [open_vcd](#)

generate_base_platform

指定した配線済みチェックポイントに基づいて、ベース プラットフォームを生成します。

構文

```
generate_base_platform [-source <arg>] [-reconfig_platform <arg>]  
                        [-base_platform <arg>] [-reconfig_platform_prefix <arg>] [-quiet]  
                        [-verbose]
```

使用法

名前	説明
[-source]	(必須) 配線済みチェックポイントのパスを指定します。
[-reconfig_platform]	(必須) リコンフィギャラブル プラットフォーム モジュールの名前を指定します。
[-base_platform]	(オプション) 出力ファイルの名前を指定します。デフォルトのファイル名は base_platform です。
[-reconfig_platform_prefix]	(オプション) リコンフィギャラブル プラットフォーム モジュールからのポート名の接頭辞を指定します。デフォルトの接頭辞は RL_ です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

generate_mem_files

すべてのシミュレーション .mem ファイルを生成します。

構文

```
generate_mem_files [-force] [-quiet] [-verbose] <directory>
```

戻り値

ディレクトリ名

使用法

名前	説明
<code>[-force]</code>	既存の .mem ファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><directory></code>	.mem ファイルを出力するディレクトリを指定します。英数字を使用したディレクトリ名を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Simulation \(シミュレーション\)](#)

説明

このコマンドは、エンベデッド プロセッサ ベースのデザインと関連付けられているソフトウェア開発キット (SDK) からの ELF (Executable Linkable File) に対し、デザインのブロック メモリ マップ (BMM) と ELF ファイルのプログラム データを結合してシミュレーションで使用するメモリ (MEM) ファイルを作成します。

MEM ファイルは、アドレス ブロックという連続アドレス空間を形成するために、ザイリンクス デバイス上の各ブロック RAM がどのようにグループ化されているかを記述し、ELF データをメモリにマップするテキスト ファイルです。

MEM ファイルの名前と生成されるファイル数は、プロセッサ システム IP コアまたは IP インテグレーター ブロック デザインで指定されたメモリ マップ データにより決定されます。

このコマンドを実行すると、MEM ファイルが記述されたディレクトリが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定の出力ディレクトリが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<directory>` (必須): メモリ ファイルを記述するディレクトリの名前を指定します。

例

次の例では、ブロック RAM マップと ELF ファイル データを結合し、シミュレーションで使用する MEM ファイルを指定のディレクトリに生成しています。

```
generate_mem_files C:/Data/gen_mem
```

関連項目

- [write_bmm](#)
- [write_mem_info](#)

generate_pblock

スタティック領域を除いた Pblock を生成します。

構文

```
generate_pblock [-cell <arg>] [-inverse_pblock <arg>]  
                [-nested_pblock <arg>] [-nested_width <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-cell]	Pblock に追加するセルを指定します。
[-inverse_pblock]	反転 Pblock の名前を指定します。この Pblock には、スタティック Pblock に含まれないすべてが含まれます。
[-nested_pblock]	スタティック Pblock に隣接する反転 Pblock の中にある入れ子の Pblock の名前を指定します。
[-nested_width]	入れ子の Pblock の幅を指定します。入れ子の Pblock の高さは、隣接するスタティック Pblock と同じになります。デフォルトは 3 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

generate_peripheral

ペリフェラル オブジェクトの出力ファイルを生成します。

構文

```
generate_peripheral [-driver] [-example_design] [-bfm_example_design]
                    [-debug_hw_example_design] [-enable_interrupt] [-force] [-quiet]
                    [-verbose] <peripheral>
```

使用法

名前	説明
<code>[-driver]</code>	ペリフェラル オブジェクトのドライバーを生成します。
<code>[-example_design]</code>	ペリフェラルのサポートされるすべてのサンプル デザインを生成します。
<code>[-bfm_example_design]</code>	ペリフェラルの BFM シミュレーション サンプル デザインを生成します。
<code>[-debug_hw_example_design]</code>	ペリフェラルのデバッグ ハードウェア サンプル デザインを生成します。
<code>[-enable_interrupt]</code>	割り込みサポートを含むペリフェラルを生成します。
<code>[-force]</code>	リポジトリの既存の IP を上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><peripheral></code>	ペリフェラル オブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[CreatePeripheral \(ペリフェラルの作成\)](#)

説明

指定したペリフェラル オブジェクトの出力ファイルを生成します。出力ファイルは、`create_peripheral` コマンドで IP を作成したときに指定した IP リポジトリの IP の名前の下に保存されます。

引数

`-driver` (オプション): 生成されたペリフェラルのソフトウェアでアドレス指定可能なレジスタのオフセットを含むソフトウェア ドライバー ファイルと、マスク、レジスタ アクセス マクロ、ユーティリティ 機能を作成します。ソフトウェア ドライバー セルフ テスト サンプル ファイルには、ペリフェラルのさまざまなハードウェア機能をテストするためのセルフ テスト サンプル コードが含まれています。

`-example_design` (オプション): 指定したペリフェラルのサンプル デザインを生成します。これには、IP インテグレーターで新しいペリフェラルを組み込んだブロック デザインを作成する `design.tcl`、サンプル デザインのシミュレーションに使用する `design_tb.v` というテストベンチが含まれます。

注記: このオプションは現在のところ、`add_peripheral_interface` コマンドの `-axi_type` オプションで定義された AXI-Stream インターフェイスではサポートされていません。

-bfm_example_design (オプション): Vivado Design Suite の IP インテグレーターを使用して、ブロック デザイン を生成する Tcl スクリプトと、AXI ペリフェラルの読み出しおよび書き込み動作をテストするバス ファンクション モデル (BFM) を生成する Tcl スクリプトを作成します。

注記: AXI4 BFM は、シミュレーションで使用するためのライセンスが必要です。

-debug_hw_example_design (オプション): ハードウェア マネージャー機能で JTAG-to-AXI デバッグ コアを使用して AXI ペリフェラルをデバッグするためのブロック デザインを生成する Tcl スクリプトを作成します。ハードウェア マネージャーの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

-enable_interrupt (オプション): ペリフェラルに割り込みをイネーブルにする割り込みピンと割り込みをサポートするロジックを追加します。

-force (オプション): IP リポジトリに最新の出力ファイルが存在する場合でも、それらを上書きします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<peripheral> (必須): 出力ファイルを生成するペリフェラルを指定します。ペリフェラルは create_peripheral コマンドで作成し、そのときにこのコマンドおよびその他の関連コマンドで処理しやすいように Tcl 変数に格納してください。例を参照してください。

例

次の例では、VLNV 属性を指定して新しい AXI ペリフェラルを作成し、そのペリフェラル オブジェクトを後の処理用に Tcl 変数に格納して、そのペリフェラルに AXI スレーブ インターフェイスを追加し、出力ファイルを生成しています。

```
set perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
    {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
generate_peripheral -driver -bfm_example_design \
    -enable_interrupt $perifObj
write_peripheral $perifObj
set_property ip_repo_paths C:/Data/new_periph [current_fileset]
update_ip_catalog -rebuild
```

関連項目

- [add_peripheral_interface](#)
- [create_peripheral](#)
- [write_peripheral](#)

generate_reports

設定可能なレポート オブジェクトのセットを生成します。

構文

```
generate_reports [-jobs <arg>] [-quiet] [-verbose] <report_configs>...
```

使用法

名前	説明
<code>[-jobs]</code>	並列実行するジョブ数を指定します。デフォルトは 1 です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><report_configs></code>	生成する設定可能なレポート オブジェクトのリストを指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[Report \(レポート\)](#)

説明

`create_report_config` コマンドで作成した指定のレポート オブジェクトを生成します。

レポート オブジェクトは合成 run またはインプリメンテーション run の特定の段階に関連付けられているので、レポートを生成するにはそれらの段階が完了している必要があります。段階が完了していない場合、またはレポートがイネーブルでない場合、`generate_report` コマンドでエラーが返されます。

引数

`-jobs <arg>` (オプション): 指定したレポート オブジェクトを生成するのに使用するジョブの数を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<report_configs>` (必須): 生成するレポート オブジェクトを 1 つまたは複数指定します。レポートは、`get_report_configs` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したレポート オブジェクトを生成しています。

```
generate_report [get_report_configs post_route_datasheet]
```



ヒント: レポートが既に生成されている場合 (STATE プロパティが GENERATED)、レポートは再生成されません。

関連項目

- [create_report_config](#)
- [delete_report_configs](#)
- [get_report_configs](#)

generate_rl_platform

ベース プラットフォームおよびラッパー モジュールに基づいて新しいプラットフォームを生成します。

構文

```
generate_rl_platform [-use_source <arg>] [-reconfig_platform <arg>]  
                    [-base_platform <arg>] [-platform <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-use_source]	ラッパー チェックポイントのパスを指定します。
[-reconfig_platform]	(必須) リコンフィギャラブル プラットフォーム モジュールの名前を指定します。
[-base_platform]	(必須) ベース プラットフォーム チェックポイントのパスを指定します。
[-platform]	(オプション) 新しいプラットフォーム チェックポイントのパスを指定します。デフォルトのファイル名は rl_platform.dcp です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

generate_shx_platform

HD.RECONFIGURABLE および関連のプロパティをサブセルに移動します。

構文

```
generate_shx_platform [-base_platform <arg>] [-wrapper <arg>]  
                      [-output <arg>] [-reconfig_platform <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-base_platform]	ベース プラットフォーム DCP へのパスを指定します。
[-wrapper]	ラッパー DCP へのパスを指定します。
[-output]	出力 DCP の名前を指定します。
[-reconfig_platform]	リコンフィギュラブル プラットフォームの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

generate_target

指定したソースのターゲット データを生成します。

構文

```
generate_target [-force] [-quiet] [-verbose] <name> <objects>
```

使用法

名前	説明
<code>[-force]</code>	ターゲット データを強制的に再生成します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	生成するターゲットを指定します。all を指定すると、サポートされるすべてのターゲットが生成されます。
<code><objects></code>	データを生成するオブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[IPIntegrator \(IP インテグレーター\)](#)

説明

指定した IP オブジェクト (`get_ips`)、あるいは IP コア (`.xci` および `.xco`)、DSP モジュール (`.slx` または `.mdl`)、またはブロック デザイン (`.bd`) の指定したソース ファイルのターゲット データを生成します。生成されるターゲット データは、FPGA デザイン フローで IP またはブロック デザインをサポートするのに必要なファイルです。

インスタンシエーション テンプレート、合成ネットリスト、シミュレーション ネットリストなどが標準のターゲットです。IP によっては、その他のターゲットをサポートするものもあります。オブジェクトに対して使用可能なターゲットは、`SUPPORTED_TARGETS` プロパティを確認するか、`list_targets` コマンドを使用してデザイン ソース ファイルのターゲットをリストできます。

引数

`-force` (オプション): ターゲット データの再生成を強制し、既存のターゲット データ ファイルを上書きします。 `-force` を指定しない場合、最新のターゲット データは再生成されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name> (必須): 指定のソース用に作成するターゲット データのタイプの名前を指定します。IP コアでサポートされるターゲットは、オブジェクトの SUPPORTED_TARGETS プロパティにリストされています。このプロパティをクエリすると、オブジェクトでサポートされているターゲットを確認できます。標準的な値は次のとおりです。

- `all`: 指定の IP またはファイルのサンプル プロジェクトを除くすべてのターゲットを生成します。
注記: IP コアのサンプル プロジェクトを生成するには、`open_example_project` コマンドを使用します。
- `instantiation_template`: IP コアの RTL モジュール定義を現在のデザインに追加するためのインスタンス化テンプレートを生成します。インスタンス化テンプレートは、デザイン階層の任意のレベルにコピーできます。
- `synthesis`: ネイティブ IP の合成中に使用される HDL ファイルまたは Vivado 合成で生成された合成済みネットリスト ファイル (DCP) を生成します。
- `simulation`: シミュレーションで使用される HDL ファイルを生成します。
- `implementation`: 現在のデザインの IP コア、DSP モジュール、またはエンベデッド プロセッサをインプリメントするのに必要なデータを生成します。

<objects> (必須): ターゲットの生成元となるオブジェクトを指定します。サポートされるオブジェクトには、IP コア オブジェクト (`get_ips`)、ソース ファイル (`.xci` または `.xco`)、IP インテグレーターからのブロック デザイン ファイル (`.bd`)、および System Generator からインポートされた DSP モジュール (`.slx` または `.mdl`) があります。

注記: ファイル名を指定するのではなく、`get_files` を使用してファイル オブジェクトを指定します。

例

次の例では、現在のプロジェクトに含まれるすべての IP コアの変更ログを生成し、ターゲットが最新の場合でも強制的に再生成しています。

```
generate_target changelog [get_ips] -force
```

次の例では、現在のプロジェクトに含まれるすべての IP コアのインスタンス化テンプレートおよび合成ターゲットを生成しています。

```
generate_target {instantiation_template synthesis} [get_ips]
```



ヒント: 複数のターゲットを指定するには、この例のように波かっこを使用します。-force オプションを使用していないので、最新でないターゲットのみが再生成されます。

次の例では、指定したブロック デザインのすべてのターゲットを生成しています。

```
generate_target all \
[get_files C:/Data/project_mb/project_mb.srscs/sources_1/bd/base_mb/
base_mb.bd]
```



重要: `get_ips` を使用してブロック デザイン内の個々の IP のターゲットを生成することはできません。エラーが返されます。

次の例では、指定した IP オブジェクトの SUPPORTED_TARGETS プロパティを取得し、その IP のサンプル プロジェクトを生成しています。

```
get_property SUPPORTED_TARGETS [get_ips blk_mem*]
open_example_project -dir C:/Data/examples -force [get_ips blk_mem*]
```

関連項目

- [add_files](#)
- [create_ip](#)
- [create_sysgen](#)
- [import_ip](#)
- [list_targets](#)
- [open_example_project](#)
- [read_ip](#)
- [report_property](#)
- [reset_target](#)

get_bd_addr_segs

セグメントのリストを取得します。

構文

```
get_bd_addr_segs [-regexp] [-hierarchical] [-filter <arg>]
                  [-of_objects <args>] [-excluded] [-addressed] [-unaddressed]
                  [-addressing] [-addressables] [-quiet] [-verbose] [<patterns>]
```

戻り値

セグメント オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセグメント、インターフェイス、またはレジスタのセグメントを取得します。
[-excluded]	除外されたマッピング済みセグメントを取得します。
[-addressed]	-of_objects で指定したオブジェクトのアドレス指定されたセグメントを取得します。
[-unaddressed]	指定したオブジェクトのアドレス指定されていないセグメントを取得します。
[-addressing]	-of_objects で指定したオブジェクトのアドレス指定セグメントを取得します。
[-addressables]	-of_objects で指定したオブジェクトのアドレス指定可能なセグメントを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインのアドレス セグメントのリストを取得します。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルまたは現在のインスタンスのアドレス セグメントを取得します。このオプションを指定しない場合、サブシステム デザイン階層の最上位のアドレス セグメントのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「`U1/*`」を指定すると、名前に「`U1/`」を含むオブジェクトが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「`U1/*`」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_addr_segs` で返されたオブジェクトのリストに、アドレス セグメントのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP インテグレーター アドレス セグメント オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`OFFSET`、`RANGE`、`USAGE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): `get_bd_addr_spaces` コマンドで返される指定のアドレス空間または `get_bd_cells` および `get_bd_intf_pins` で返されるセルまたはインターフェイス ピンのアドレス空間に割り当てられているアドレス セグメントを取得します。`get_bd_addr_segs` コマンドを使用してマスター アドレス セグメントに関連付けられているスレーブ アドレス セグメント (address_segment) を取得することもできます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

-excluded (オプション): `-of_objects` オプションで指定されたアドレス空間から除外されているアドレス セグメントを取得します。

-addressing (オプション): `-of_objects` オプションで指定されたアドレス空間のアドレス指定セグメントを取得します。アドレス空間で使用可能なすべてのセグメントが返されます。

-addressed (オプション): `-of_objects` オプションで指定されたアドレス空間のアドレス指定されたセグメントを取得します。使用されているアドレス空間のセグメントが返されます。

-addressables (オプション): `-of_objects` オプションで指定されたオブジェクトのアドレス空間の、アドレス セグメントが割り当てられていないアドレス指定可能なセグメントを取得します。

注記: `-addressables`、`-addressed`、および `-addressing` オプションを同時に使用することはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): アドレス セグメントを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在の IP サブシステム デザインに含まれるすべてのアドレス セグメントのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてアドレス セグメントを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエLEMENTとして指定します。

例

次の例では、指定したアドレス空間のアドレス セグメントを取得しています。

```
get_bd_addr_segs -of_objects [get_bd_addr_spaces -of_objects \
[get_bd_cells /microblaze_1]]
/microblaze_1/Data/SEG1
/microblaze_1/Data/SEG3
/microblaze_1/Instruction/SEG2
```

注記: 検索パターンに一致するオブジェクトがない場合は、警告メッセージが表示されます。

関連項目

- [create_bd_addr_seg](#)
- [exclude_bd_addr_seg](#)
- [get_bd_addr_spaces](#)
- [get_bd_cells](#)
- [get_bd_intf_pins](#)
- [include_bd_addr_seg](#)

get_bd_addr_spaces

アドレス空間のリストを取得します。

構文

```
get_bd_addr_spaces [-regexp] [-hierarchical] [-filter <arg>]
                  [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

addr_space オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセグメントまたはインターフェイスのアドレス空間を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在の IP インテグレーター サブシステム デザインのアドレス空間のリストを取得します。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルまたは現在のインスタンスのアドレス空間を取得します。このオプションを指定しない場合、デザイン階層の最上位のアドレス空間のみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「U1/*」を指定すると、名前に「U1/」を含むオブジェクトが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「U1/*」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_addr_spaces` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP インテグレーター アドレス空間オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、RANGE、OFFSET などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`==`」、不一致「`!=`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定した IP インテグレーター アドレス セグメント、セル、またはインターフェイス ピン オブジェクトを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): アドレス空間を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、現在の IP サブシステム デザインに含まれるすべてのアドレス空間のリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてアドレス空間を検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、現在の IP インテグレーター サブシステム デザインのすべてのアドレス空間を 1 行に 1 つずつリストしています。

```
join [get_bd_addr_spaces] \n
/mdm_1/S_AXI
/microblaze_1/Data
/microblaze_1/Instruction
/microblaze_1_axi_intc/s_axi
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB1
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB2
/microblaze_1_local_memory/dlmb_bram_if_cntlr/SLMB3
/microblaze_1_local_memory/dlmb_bram_if_cntlr/S_AXI_CTRL
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB1
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB2
/microblaze_1_local_memory/ilmb_bram_if_cntlr/SLMB3
/microblaze_1_local_memory/ilmb_bram_if_cntlr/S_AXI_CTRL
/microblaze_1_local_memory/lmb_bram/S_1
```

注記: 検索パターンに一致するオブジェクトがない場合は、警告メッセージが表示されます。

次の例では、現在のサブシステム デザインのすべてのアドレス空間を取得し、3 番目にリストされるアドレス空間に設定されているすべてのプロパティを取得しています。

```
report_property -all [lindex [get_bd_addr_spaces] 2 ]
Property  Type      Read-only  Visible  Value
CLASS     string    true      true     bd_addr_space
NAME      string    false     true     /microblaze_1/Instruction
OFFSET    string    false     true
PATH      string    true      true     /microblaze_1/Instruction
RANGE     string    false     true     -1
TYPE      string    false     true
```

関連項目

- [create_bd_addr_seg](#)
- [exclude_bd_addr_seg](#)
- [get_bd_addr_segs](#)
- [get_bd_cells](#)
- [get_bd_intf_pins](#)
- [include_bd_addr_seg](#)

get_bd_cells

ブロック図のセルのリストを取得します。

構文

```
get_bd_cells [-regexp] [-hierarchical] [-filter <arg>] [-of_objects <args>]  
             [-quiet] [-verbose] [<patterns>]
```

戻り値

ブロック図のセル オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピンまたはネットのセルを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在の IP インテグレーター サブシステム デザインまたは現在のサブシステム インスタンスのセルのリストを取得します。IP インテグレーター サブシステム セルは、IP インテグレーター カタログからの IP コア、または create_bd_cell コマンドでサブシステム デザインに作成された階層モジュールです。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-hierarchical (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルまたは現在のインスタンスのセルを取得します。このオプションを指定しない場合、デザイン階層の最上位のセルのみが取得されます。**-hierarchical** を使用する場合、検索パターンは完全な階層セル名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「U1/*」を指定すると、名前に「U1/」を含むインスタンスが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: **-regexp** と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「U1/*」を指定した場合に意図した結果が得られます。

-filter <args> (オプション): 結果のリストに指定した式のフィルターを適用します。**-filter** オプションを使用すると、**get_bd_cells** で返されたオブジェクトのリストに、ブロック デザイン セルのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、**report_property** または **list_property** コマンドで確認できます。IP インテグレーター セル オブジェクトのリストをフィルター処理するのに使用できるプロパティには、VLNV、TYPE、LOCATION などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「*****」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*****) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (**==**)、不等価 (**!=**)、一致 (**=~**)、不一致 (**!~**) です。数値比較演算子 **<**、**>**、**<=**、および **>=** も使用できます。複数のフィルター式を **AND** (**&&**) および **OR** (**||**) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (**bool**) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <arg> (オプション): **get_bd_nets** で指定したネット オブジェクト、あるいは **get_bd_pins** または **get_bd_intf_pins** コマンドで指定したピンに接続されているサブシステム セルを取得します。

注記: **-of_objects** オプションでは、オブジェクトを名前で指定するのではなく、**get_*** コマンド (**get_cells**、**get_pins** など) を使用して指定する必要があります。また、**-of_objects** を検索パターン (**<pattern>**) と共に使用することはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、**TCL_OK** が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、**set_msg_config** コマンドで定義できます。

<patterns> (オプション): サブシステム セルを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*****) で、現在の IP サブシステム デザインに含まれるすべてのセルのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてセルを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、指定の IP インテグレーター サブシステムのピンを含むセルのリストを取得し、そのリストから重複を削除しています。

```
lsort -unique [get_bd_cells -of_objects [get_bd_pins -hierarchical *aclk*]]
```

注記: 検索パターンに一致するセルがない場合は、警告メッセージが表示されます。

次の例では、サブシステム デザイン階層のすべてのレベルにあるすべてのセルのリストを取得し、そのリストをフィルターして指定のテキストを含むセルまたは階層を含むもののみを取得しています。

```
get_bd_cells -hierarchical -filter {NAME=~"/newMod1/*"}
```

関連項目

- [create_bd_cell](#)
- [get_bd_intf_pins](#)
- [get_bd_nets](#)
- [get_bd_pins](#)
- [list_property](#)
- [report_property](#)

get_bd_designs

デザインのリストを取得します。

構文

```
get_bd_designs [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
               [-verbose] [<patterns>...]
```

戻り値

デザイン オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定したブロック デザイン セル、ピン、またはネットの図を取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	検索パターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在のプロジェクトで開いている IP サブシステム デザインで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトで開いている IP サブシステム デザインすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_designs` で返されたオブジェクトのリストに、ブロック デザインのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP サブシステム デザイン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`FILE_NAME` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (`*`) を使用すると、定義値が `""` のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を `AND (&&)` および `OR (||)` で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したブロック デザインのセル (`bd_cell`)、ブロック デザインのピン (`bd_pin`)、またはブロック デザインのネット (`bd_net`) オブジェクトに関連付けられている BD デザインを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): デザインを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (`*`) で、すべての IP サブシステム デザインが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてデザインを検索できます。

例

次の例では、現在のプロジェクトで開いている IP サブシステム デザインすべてのリストを取得しています。

```
get_bd_designs
```

関連項目

- [create_bd_design](#)
- [current_bd_design](#)
- [open_bd_design](#)

- [report_property](#)

get_bd_intf_nets

インターフェイス ネットのリストを取得します。

構文

```
get_bd_intf_nets [-regexp] [-hierarchical] [-filter <arg>]
                 [-boundary_type <arg>] [-of_objects <args>] [-quiet] [-verbose]
                 [<patterns>]
```

戻り値

ピン オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-boundary_type]	ソース オブジェクトが階層ブロックのインターフェイス ピン上にある場合に使用します。有効な値は、upper、lower、both です。lower に設定すると、下位階層から検索が実行されます。このオプションは、connected_to 関係にのみ有効です。デフォルトは upper です。
[-of_objects]	指定したセル、インターフェイス ピンまたはポート オブジェクトに接続されているネットのリストを取得します。リストは 1 つのオブジェクト タイプである必要があります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに含まれるインターフェイス ネットで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのインターフェイス ネットのリストが返されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルのインターフェイス ネットを取得します。このオプションを指定しない場合、サブシステム デザインの最上位、または現在のサブシステム インスタンスのインターフェイス ネットのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「`/bridge_1/*`」を指定すると、名前に「`/bridge_1/`」を含むネットが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「`U1/*`」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_intf_nets` で返されたオブジェクトのリストに、ブロック デザイン インターフェイス ネットのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP インテグレーター インターフェイス ネット オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、PATH があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-boundary_type [upper | lower | both]` (オプション): 指定の階層ピンのレベル (upper)、ピンまたはポートの下レベル (lower)、またはその両方のネットを取得します。有効な値は、upper、lower、both です。デフォルト値は upper です。

注記: このオプションを使用する場合、`-of_objects` オプションを使用して階層ピンを指定する必要があります。

`-of_objects <args>` (オプション): 指定した IP インテグレーター サブシステムのセル、ピン、またはポート オブジェクトに接続されているネットのリストを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): IP サブシステムのインターフェイス ネットを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在の IP インテグレーター サブシステム デザインに含まれるすべてのインターフェイス ネットのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてネットを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、IP インテグレーター階層モジュールの指定のピンに接続されているインターフェイス ネットを取得しています。階層モジュールのレベルのネットと階層モジュール内のネットの両方が返されます。

```
get_bd_intf_nets -boundary_type both -of_objects [get_bd_pins /newMod1/
ac1k]
```

注記: 検索パターンに一致するインターフェイス ネットがない場合は、警告メッセージが表示されます。

次の例では、IP インテグレーター サブシステム デザインのすべての階層レベルのインターフェイス ネットを取得しています。

```
get_bd_intf_nets -hierarchical
```

関連項目

- [connect_debug_port](#)
- [get_cells](#)
- [get_clocks](#)
- [get_pins](#)
- [get_ports](#)
- [list_property](#)
- [report_property](#)

get_bd_intf_pins

インターフェイス ピンのリストを取得します。

構文

```
get_bd_intf_pins [-regexp] [-hierarchical] [-filter <arg>]
                 [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

ピン オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセル、インターフェイス ネットまたはピン オブジェクトに接続されているピンを取得します。リストは 1 つのオブジェクト タイプである必要があります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在の IP サブシステム デザインに含まれるインターフェイス ピン オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのインターフェイス ピンのリストが返されます。

ピンおよびインターフェイス ピンは、IP インテグレーター セルまたは階層モジュールの外部接続です。ポートおよびインターフェイス ポートは、IP サブシステム デザインの外部接続です。ポート オブジェクトを選択するには、`get_bd_ports` および `get_bd_intf_ports` コマンドを使用します。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルのインターフェイス ピンを取得します。このオプションを指定しない場合、サブシステム デザインの最上位、または現在のサブシステム インスタンスのインターフェイス ピンのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「`/bridge_1/*`」を指定すると、名前に「`/bridge_1/`」を含むインターフェイス ピンが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「`U1/*`」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_intf_pins` で返されたオブジェクトのリストに、ブロック デザイン インターフェイス ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP インテグレーター インターフェイス ピン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、DIR、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したブロック デザインのピン (bd_pin)、ブロック デザインのセル (bd_cell)、またはブロック デザインのインターフェイス ネット (bd_intf_net) に接続されているピンを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): IP インテグレーター サブシステムのインターフェイス ピンを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在の IP インテグレーター サブシステム デザインまたはインスタンスに含まれるすべてのインターフェイス ピンのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、指定したセルに接続されているインターフェイス ピンのリストが取得されます。

```
get_bd_intf_pins -of [get_bd_cells new_vidOut_1]
```

注記: 検索パターンに一致するインターフェイス ピンがない場合は、警告メッセージが表示されます。

次の例では、IP インテグレーター サブシステム デザインのすべての階層レベルのインターフェイス ピンで、指定の名前パターンに一致するものを取得しています。

```
get_bd_intf_pins -hierarchical m_apb*
```

次の例では、指定したサブシステムのネットに接続されているインターフェイス ピンのリストを取得しています。

```
get_bd_intf_pins -of [get_bd_intf_nets vidout_1_vtg_ce]
```

関連項目

- [create_bd_net](#)
- [create_bd_port](#)
- [get_bd_intf_ports](#)
- [get_bd_pins](#)
- [list_property](#)
- [report_property](#)

get_bd_intf_ports

インターフェイス ポートのリストを取得します。

構文

```
get_bd_intf_ports [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
                  [-verbose] [<patterns>]
```

戻り値

ポート オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したインターフェイス ネットまたはポート オブジェクトに接続されているポートを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに含まれるインターフェイス ポート オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのインターフェイス ポートのリストが返されます。

ポートおよびインターフェイス ポートは、IP サブシステム デザインの外部接続です。ピンおよびインターフェイスピンは、IP インテグレーター セルまたは階層モジュールの外部接続です。ピン オブジェクトを選択するには、`get_bd_pins` および `get_bd_intf_pins` コマンドを使用します。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_intf_ports` で返されたオブジェクトのリストに、ブロック デザイン インターフェイス ポートのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP サブシステムのインターフェイス ポート オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DIR`、`MODE`、`LOCATION` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): `get_bd_intf_nets` で返された指定の IP サブシステムのインターフェイス ネットに接続されているインターフェイス ポート、または `get_bd_ports` コマンドで返されたブロック デザインのポートが含まれるインターフェイス ポートを返します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): インターフェイス ポートを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、サブシステム デザインに含まれるすべてのインターフェイス ポートのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてインターフェイス ポートを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、サブシステム デザインでマスター モードで動作しているインターフェイス ポートを取得しています。

```
get_bd_intf_ports -filter {MODE=="master"}
```

注記: 検索パターンに一致するインターフェイス ポートがない場合は、警告メッセージが表示されます。

次の例では、指定したブロック デザインのポートが含まれるブロック デザインのインターフェイス ポートを取得しています。

```
get_bd_intf_ports -of [get_bd_ports sys_diff_clock_clk_n]
```

関連項目

- [create_bd_intf_net](#)
- [create_bd_intf_port](#)
- [get_bd_intf_nets](#)
- [get_bd_intf_pins](#)
- [get_bd_nets](#)
- [get_bd_pins](#)
- [get_bd_ports](#)
- [list_property](#)
- [report_property](#)

get_bd_nets

ネットのリストを取得します。

構文

```
get_bd_nets [-regexp] [-hierarchical] [-filter <arg>]
            [-boundary_type <arg>] [-of_objects <args>] [-quiet] [-verbose]
            [<patterns>]
```

戻り値

ピン オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-boundary_type]	ソース オブジェクトが階層ブロックのピン上にある場合に使用します。有効な値は、upper、lower、both です。lower に設定すると、下位階層から検索が実行されます。このオプションは、connected_to 関係にのみ有効です。デフォルトは upper です。
[-of_objects]	指定したセル、ピンまたはポート オブジェクトに接続されているネットのリストを取得します。リストは 1 つのオブジェクトタイプである必要があります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに含まれるネットで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのネットのリストが返されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルのネットを取得します。このオプションを指定しない場合、サブシステム デザインの最上位、または現在のサブシステム インスタンスのネットのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「`/bridge_1/*`」を指定すると、名前に「`/bridge_1/`」を含むネットが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「`U1/*`」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_nets` で返されたオブジェクトのリストに、ブロック デザイン ネットのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP インテグレーター サブシステムのネット オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、PATH があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-boundary_type [upper | lower | both]` (オプション): 指定の階層ピンのレベル (upper)、ピンまたはポートの下レベル (lower)、またはその両方のネットを取得します。有効な値は、upper、lower、both です。デフォルト値は upper です。

注記: このオプションを使用する場合、`-of_objects` オプションを使用して階層ピンを指定する必要があります。

`-of_objects <args>` (オプション): 指定したセル、ピン、またはポート オブジェクトに接続されているネットのリストを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): IP サブシステムのネットを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在の IP インテグレーター サブシステム デザインに含まれるすべてのネットのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてネットを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、IP インテグレーター階層モジュールの指定のピンに接続されているネットを取得しています。階層モジュールのレベルのネットと階層モジュール内のネットの両方が返されます。

```
get_bd_nets -boundary_type both -of_objects [get_bd_pins /newMod1/ac1k]
```

注記: 検索パターンに一致するネットがない場合は、警告メッセージが表示されます。

次の例では、IP インテグレーター サブシステム デザインのすべての階層レベルのネットを取得しています。

```
get_bd_nets -hierarchical
```

関連項目

- [connect_debug_port](#)
- [get_cells](#)
- [get_clocks](#)
- [get_pins](#)
- [get_ports](#)
- [list_property](#)
- [report_property](#)

get_bd_pins

ピンのリストを取得します。

構文

```
get_bd_pins [-regexp] [-hierarchical] [-filter <arg>] [-of_objects <args>]  
            [-quiet] [-verbose] [<patterns>]
```

戻り値

ピン オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-hierarchical]	階層セルを含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセル、ネット、またはインターフェイス ピン オブジェクトに接続されているピンを取得します。リストは 1 つのオブジェクト タイプである必要があります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP サブシステム デザインに含まれるピン オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのピンのリストが返されます。

ピンおよびインターフェイス ピンは、IP インテグレーター セルまたは階層モジュールの外部接続です。ポートおよびインターフェイス ポートは、IP サブシステム デザインの外部接続です。ポート オブジェクトを選択するには、`get_bd_ports` および `get_bd_intf_ports` コマンドを使用します。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-hierarchical` (オプション): IP インテグレーター サブシステム デザインのすべての階層レベルのピンを取得します。このオプションを指定しない場合、サブシステム デザインの最上位、または現在のサブシステム インスタンスのピンのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「`/bridge_1/*`」を指定すると、名前に「`/bridge_1/`」を含むピンが階層の各レベルで検索され、意図した結果が得られない可能性があります。

注記: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「`U1/*`」を指定した場合に意図した結果が得られます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_pins` で返されるオブジェクトのリストに、ブロック デザインのピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ブロック デザインのピン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、DIR、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定した IP サブシステムのインターフェイス ピン、ブロック デザインのインターフェイス ピン (`bd_intf_pin`)、ブロック デザインのセル (`bd_cell`)、またはブロック デザインのネット (`bd_net`) に接続されるピンを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): IP インテグレーター サブシステムのピンを検索するパターンを指定します。検索パターンは、`<bd_cell_name>/<bd_pin_name>` という形式で指定する必要があります。ワイルドカード「*/」を使用すると、ダイアグラム内のすべてのブロック デザインのセル (`bd_cell`) にあるすべてのブロック デザインのピン (`bd_pin`) オブジェクトが返されます。

注記: 複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、指定したセルに接続されているピンのリストが取得されます。

```
get_bd_pins -of [get_bd_cells new_vidOut_1]
```

注記: 検索パターンに一致するピンがない場合は、警告メッセージが表示されます。

次の例では、IP インテグレーター サブシステム デザインのすべての階層レベルのピンで、指定の名前パターンに一致するものを取得しています。

```
get_bd_pins -hierarchical LMB*
```

次の例では、指定したサブシステムのネットに接続されているピンのリストを取得しています。

```
get_bd_pins -of [get_bd_nets vidout_1_vtg_ce]
```

関連項目

- [create_bd_net](#)
- [create_bd_port](#)
- [get_bd_intf_pins](#)
- [get_bd_intf_ports](#)
- [list_property](#)
- [report_property](#)

get_bd_ports

ポートのリストを取得します。

構文

```
get_bd_ports [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
              [-verbose] [<patterns>]
```

戻り値

ポート オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したネットまたはインターフェイス ポート オブジェクトに接続されているブロック デザインのポートを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在の IP インテグレーター サブシステム デザインに含まれるポート オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、サブシステム デザインに含まれるすべてのポートのリストが返されます。

ポートおよびインターフェイス ポートは、IP サブシステム デザインの外部接続です。ピンおよびインターフェイスピンは、IP インテグレーター セルまたは階層モジュールの外部接続です。ピン オブジェクトを選択するには、`get_bd_pins` および `get_bd_intf_pins` コマンドを使用します。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bd_ports` で返されたオブジェクトのリストに、ブロック デザイン ポートのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP サブシステムのポート オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DIR`、`MODE`、`LOCATION` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): `get_bd_nets` で返された指定の IP サブシステムのネットに接続されているポート、または `get_bd_intf_ports` コマンドで返されたインターフェイス ポートに含まれるポートを返します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ポートを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、サブシステム デザインに含まれるすべてのポートのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてポートを検索できます。

注記: 複数の検索パターンは {} で囲み、1 つのエレメントとして指定します。

例

次の例では、指定した IP サブシステムのネットに接続されているポートを取得しています。

```
get_bd_ports -of_objects [get_bd_nets bridge_1_apb_m]
```

注記: 検索パターンに一致するポートがない場合は、警告メッセージが表示されます。

関連項目

- [create_bd_net](#)
- [create_bd_port](#)
- [get_bd_intf_pins](#)
- [get_bd_intf_ports](#)
- [get_bd_pins](#)
- [list_property](#)
- [report_property](#)

get_bd_regs

レジスタのリストを取得します。

構文

```
get_bd_regs [-of_objects <args>] [-quiet] [-verbose]
```

戻り値

レジスタ オブジェクトのリスト、エラーが発生した場合は ""。

使用法

名前	説明
[-of_objects]	指定したセグメント、インターフェイス ピン、外部インターフェイス ポートのレジスタを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

アドレス セグメント、インターフェイス ピン、または外部インターフェイス ポートのオブジェクトを使用して、レジスタのリストを取得します。レジスタ オブジェクトは、レジスタの表示名とメモリ オフセット アドレスを示します。このコマンドには、`-of_objects` オプションを使用してアドレス セグメント、インターフェイス ピン、または外部インターフェイス ポートのオブジェクトを指定する必要があるため、デフォルトの動作はありません。このコマンドは、アドレス セグメント オブジェクトから直接レジスタを取得するか、指定したインターフェイス ピンまたはインターフェイス ポート オブジェクトに関連付けられているアドレス セグメントからレジスタを取得します。

引数

`-of_objects <args>` (必須): アドレス セグメント、インターフェイス ピン、または外部インターフェイス ポートのオブジェクトを `get_bd_addr_segs`、`get_bd_intf_pins`、または `get_bd_intf_ports` コマンドを使用して指定します。指定したオブジェクトがアドレス セグメントの場合は、そのアドレス セグメントに直接関連付けられているレジスタが取得されます。指定したオブジェクトがインターフェイス ピンまたは外部インターフェイス ポートの場合は、それらのオブジェクトに関連付けられているアドレス セグメントが検索され、それらのアドレス セグメントに関連付けられているレジスタが取得されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、外部インターフェイス ピンを作成し、アドレス空間を割り当てて、それらのアドレス空間に関連付けられているレジスタを取得します。

```
% create_bd_cell -type ip -vlnv xilinx.com:ip:axi-gpio:2.0 axi-gpio_0
% make_bd_intf_pins_external [get_bd_intf_pins axi-gpio_0/S_AXI]
% assign_bd_address
% get_bd_addr_segs
/axi-gpio_0/S_AXI/Reg /S_AXI_0/SEG_axi-gpio_0-Reg

% set seg [get_bd_addr_segs /axi-gpio_0/S_AXI/Reg]
/axi-gpio_0/S_AXI/Reg

% get_bd_regs -of_objects $seg
GPIO_DATA GPIO_TRI GPIO2_DATA GPIO2_TRI GIER IP_IER IP_ISR

%set regs [get_bd_regs -of_objects $seg]
GPIO_DATA GPIO_TRI GPIO2_DATA GPIO2_TRI GIER IP_IER IP_ISR

% set reg [lindex $regs 0]
GPIO_DATA

% report_property $reg
Property      Type      Read-only  Visible  Value
CLASS         string   true      true     bd_reg
DISPLAYNAME   string   true      true     Channel_1_GPIO_DATA
NAME          string   true      true     GPIO_DATA
OFFSET        string   true      true     0x0
```

関連項目

- [get_bd_addr_segs](#)
- [get_bd_intf_pins](#)
- [get_bd_intf_ports](#)
- [report_property](#)

get_bel_pins

BEL ピン (bel_pin) のリストを取得します。デザインが読み込まれている場合、構築されたサイト タイプ BEL を取得します。

構文

```
get_bel_pins [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
              [-quiet] [-verbose] [<patterns>]
```

戻り値

bel_pin オブジェクト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した BEL、サイト、ピン、またはネットの bel_pin を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	bel_pin を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

指定した BEL のピンで、指定した検索パターンに一致するもののリストを返します。

デフォルトでは、デバイス上にあるすべての BEL のピンすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bel_pins` で返されたオブジェクトのリストに、BEL ピンのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。`bel_pin` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`IS_INVERTED` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): このオプションは `get_bels` コマンドと共に使用でき、指定した BEL のピンを返します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): bel_pin オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、デバイス上のすべての bel_pin のリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、デバイス上の指定したサイト範囲に関連する指定した BEL のピンが返されます。

```
get_bel_pins -of_objects [get_bels -of_objects [get_sites \
    -range {SLICE_X0Y0 SLICE_X1Y1}]] ]
```

次の例では、デバイス上にあるすべての BEL のクロック イネーブル (CE) ピンが返されます。

```
get_bel_pins *CE
```

関連項目

- [get_bels](#)
- [get_sites](#)
- [list_property](#)
- [report_property](#)

get_bels

BEL のリストを取得します。デザインが読み込まれている場合、構築されたサイト タイプ BEL を取得します。

構文

```
get_bels [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
         [-include_routing_bels] [-quiet] [-verbose] [<patterns>]
```

戻り値

BEL

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した SLR、タイル、サイト、セル、clock_region、ネットの BEL を取得します。
[-include_routing_bels]	一致する配線 BEL も検索して含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	BEL を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[XDC](#)

説明

基本エレメント (BEL) とは、フリップフロップ、LUT、キャリー ロジックなど、スライスを構成するロジックの構築ブロックです。このコマンドは、開いているデザインのターゲット デバイスの BEL で、指定した検索パターンに一致するものをリストします。

デフォルトでは、デバイスのすべての BEL のリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_bels` で返されたオブジェクトのリストに、BEL のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。BEL オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IS_OCCUPIED`、`TYPE` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ " *RESET*" }
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したセル、ネット、サイト、クロック領域 (clock_region)、または SLR に関連付けられている BEL を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): BEL を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、デバイスの BEL すべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて BEL を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、ターゲット パーツの BEL の総数が返されます。

```
llength [get_bels]
```

次の例では、指定のサイトに関連する BEL が返されます。

```
get_bels -of_objects [get_sites PHASER_IN_PHY_X0Y5]
```

関連項目

- [get_sites](#)
- [list_property](#)
- [report_property](#)

get_board_bus_nets

ボード バス ネット オブジェクトのリストを取得します。

構文

```
get_board_bus_nets [-regexp] [-nocase] [-all] [-filter <arg>]
                   -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボードのバス ネットのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	イネーブルのバスおよびディスエーブルのバスすべてを返します。
[-filter]	式を使用してリストをフィルター処理します。
-of_objects	指定した board_bus、board_component、board_component_pin タイプの board_bus_net オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード ネット名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

Object (オブジェクト)、Board (ボード)

説明

指定した接続バス オブジェクト、ボード コンポーネント、またはボード コンポーネント ピン オブジェクトの個別の接続バス ネットのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』 (UG895) を参照してください。

接続バスは、サイリンクス デバイス (part0) とボード上のほかのコンポーネントとの接続を定義します。バス ネットは、接続バスの個々の接続を定義します。

このコマンドを実行すると、接続バス ネットのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および **-filter** オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「**.**」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド **regexp** はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、**-regexp** オプションを使用した場合にのみ適用されます。

-all (オプション): 指定したオブジェクトのボード インターフェイス ファイルで定義されている接続バス ネットすべてのリストを返します。

-filter (オプション): 結果のリストに指定した式のフィルターを適用します。**-filter** オプションを使用すると、**get_board_bus_nets** で返されたオブジェクトのリストに、バス ネットのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、**report_property** または **list_property** コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード バス ネット オブジェクトのリストをフィルター処理するのに使用できるプロパティには、BUS、NAME、DELAY などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「**.**」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <args> (必須): 指定したボード バス (board_bus)、ボード コンポーネント (board_component)、またはボード コンポーネント ピン (board_component_pin) オブジェクトの接続バスのネットのリストを取得します。

注記: **-of_objects** オプションでは、オブジェクトを名前で指定するのではなく、**get_*** コマンド (**get_cells**、**get_pins** など) を使用して指定する必要があります。また、**-of_objects** を検索パターン (<pattern>) と共に使用することはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボード バス ネットを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトの接続バス ネットすべてのリストが取得されます。

例

次の例では、現在のボードの指定したコンポーネントに関連付けられている接続バス ネットを取得しています。

```
get_board_bus_nets -of_objects [get_board_components {*iic-main*}]
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_buses

ボード バス オブジェクトのリストを取得します。

構文

```
get_board_buses [-regex] [-nocase] [-all] [-filter <arg>]
                 [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボードのバスのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	イネーブルのバスおよびディスエーブルのバスすべてを返します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board、board_component、board_bus_net タイプの board_bus オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード バス名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regex を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されている接続バスのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

接続バスは、ザイリンクス デバイス (part0) とボード上のほかのコンポーネントとの接続を定義します。

このコマンドを実行すると、バスのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): 現在のボードのボード インターフェイス ファイルで定義されている接続バスすべてのリストを返します。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_buses` で返されたオブジェクトのリストに、バスのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード バス オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`COMPONENT_1`、`COMPONENT_2`、`DELAY` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> (オプション): 指定したボード (board)、ボード コンポーネント (board_component)、またはボード バス ネット (board_bus_net) オブジェクトの接続バスのリストを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボード バスを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のボード上で定義されている接続バスすべてのリストが取得されます。

例

次の例では、現在のボードの指定したコンポーネントに関連付けられている接続バスを取得しています。

```
get_board_buses -of_objects [get_board_components sgmi]
```

関連項目

- [current_board_part](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_component_interfaces

VLNV で指定されたバス定義をインプリメントする、ボード コンポーネントのインターフェイスのリストを取得します。

構文

```
get_board_component_interfaces [-regexp] [-nocase] [-all] [-filter <arg>]
                                [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

バス インターフェイスのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	イネーブルのインターフェイスおよびディスエーブルのインターフェイスすべてを返します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board、board_component の board_component_interface オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	バス インターフェイスを検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボード上にあるコンポーネントで定義されているインターフェイスを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、定義されているコンポーネント、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895)を参照してください。

コンポーネント インターフェイスは、コンポーネント上またはコンポーネント モードにある関連する信号のグループを定義します。

このコマンドを実行すると、コンポーネント インターフェイスのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): 現在のボードのボード インターフェイス ファイルで定義されているコンポーネント インターフェイスすべてのリストを返します。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_component_interfaces` で返されたオブジェクトのリストに、コンポーネント インターフェイスのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。コンポーネント インターフェイス オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、BUSDEF_NAME があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したボード (board) またはボード コンポーネント (board_component) オブジェクトのコンポーネント インターフェイスのリストを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): コンポーネント インターフェイスを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のボード上で定義されているコンポーネント インターフェイスすべてのリストが取得されます。

例

次の例では、指定したボード コンポーネントのボード インターフェイス ファイルで定義されているコンポーネント インターフェイスを取得しています。

```
get_board_component_interfaces -of_objects [get_board_components *part0*]
```

次の例では、ボード インターフェイス ファイルで定義されているコンポーネント インターフェイスを取得し、その情報を使用して現在のプロジェクトにインターフェイスを作成しています。

```
#Get and Create Interfaces for I/O Ports
foreach x [get_board_component_interfaces] {
  create_interface $x }
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_component_modes

コンポーネント モード オブジェクトのリストを取得します。

構文

```
get_board_component_modes [-regexp] [-nocase] [-all] [-filter <arg>]
                             -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボードのコンポーネント モードのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	イネーブルのインターフェイスおよびディスエーブルのインターフェイスすべてを返します。
[-filter]	式を使用してリストをフィルター処理します。
-of_objects	指定した board_component オブジェクトの board_component_mode オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード コンポーネント モード名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されているコンポーネント モードのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

コンポーネント モードは、ボード上のコンポーネントのさまざまな操作モード、インターフェイス、およびそれらのモードに適切な IP を定義します。

このコマンドを実行すると、コンポーネント モードのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): 現在のボードのボード インターフェイス ファイルで定義されているコンポーネント モードすべてのリストを返します。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_component_modes` で返されたオブジェクトのリストに、コンポーネント モードのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。コンポーネント モード オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、INTERFACES があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (必須): 指定したボード コンポーネント (board_component) のコンポーネント モードを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): コンポーネント モードを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したボード コンポーネント上で定義されているコンポーネント モードすべてのリストが取得されます。

例

次の例では、指定したボードのボード インターフェイス ファイルで定義されているコンポーネント モードを取得しています。

```
get_board_component_modes -of_objects [get_board_components *part0*]
```

関連項目

- [current_board](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_component_pins

ボード パーツ ピン オブジェクトのリストを取得します。

構文

```
get_board_component_pins [-regexp] [-nocase] [-filter <arg>]
                          -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボード パーツのピンのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
-of_objects	指定した board_component、board_bus_net の board_component_pin オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード コンポーネント ピン名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードの指定したボード コンポーネントのボード コンポーネント ピンのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード コンポーネントは、ボード インターフェイス ファイルで定義されているボードのコンポーネントを定義します。コンポーネント ピンは、コンポーネントの個々のピンです。

このコマンドを実行すると、コンポーネント ピン オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_component_pins` で返されたオブジェクトのリストに、コンポーネント ピンのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード コンポーネント ピン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、IOSTANDARD、PIN_INDEX などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> (必須): 指定したボード バス ネット (board_bus_net) またはボード コンポーネント (board_component) オブジェクトのコンポーネント ピンを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ボード コンポーネント ピンを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトのコンポーネント ピンすべてのリストが取得されます。

例

次の例では、現在のボードの指定したザイリンクス デバイス (part0) コンポーネントに関連付けられているコンポーネント ピンを取得しています。

```
get_board_component_pins -of_objects [get_board_components *part0*]
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_components](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_components

ボードで使用可能なコンポーネントのリストを取得します。

構文

```
get_board_components [-regexp] [-nocase] [-all] [-filter <arg>]
                     [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

コンポーネント オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	イネーブルのコンポーネントおよびディスエーブルのコンポーネントすべてを返します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board、board_bus、または board_component_pin オブジェクトの board_component オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	コンポーネント名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されているコンポーネントのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

このコマンドを実行すると、コンポーネントのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): 現在のボードのボード インターフェイス ファイルで定義されているコンポーネントすべてのリストを返します。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_components` で返されたオブジェクトのリストに、コンポーネントのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード コンポーネント オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`COMPONENT_NAME`、`TYPE`、`DESCRIPTION` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したボード (board)、ボード バス (board_bus)、ボード コンポーネント ピン (board_component_pin) オブジェクトのコンポーネントを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ボード コンポーネントを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のボード上で定義されているボード コンポーネントすべてのリストが取得されます。

例

次の例では、指定したボードのボード インターフェイス ファイルで定義されているコンポーネントを取得しています。

```
get_board_components -of_objects [get_boards zed]
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_interface_ports

インターフェイス ポート オブジェクトのリストを取得します。

構文

```
get_board_interface_ports [-regexp] [-nocase] [-filter <arg>]
                           -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

戻り値

インターフェイス ポートのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
-of_objects	指定した board_component_interface、board_component_pin の board_component_port オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	インターフェイス ポート名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されているコンポーネント インターフェイスに割り当てられている物理ポートのリストを取得します。get_board_component_interfaces コマンドで返されたコンポーネント インターフェイス、または get_board_component_pins コマンドで返されたコンポーネント ピンからのインターフェイス ポートを返すことができます。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード インターフェイスファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895)を参照してください。

ボード インターフェイス ファイルでは、コンポーネント インターフェイスにインターフェイス ファイルで定義されている論理ポートと、ザイリンクス デバイス (part0) のコンポーネント ピンに関連する物理ポートとのマップが含まれます。

このコマンドを実行すると、指定したコンポーネント インターフェイスの物理ポートのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): 現在のボードのボード インターフェイス ファイルで定義されているコンポーネント インターフェイスすべてのリストを返します。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。 `-filter` オプションを使用すると、`get_board_interface_ports` で返されたオブジェクトのリストに、インターフェイス ポートのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。インターフェイス ポート オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`LOGICAL_NAME`、`DIRECTION` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (必須): 指定したボード コンポーネント インターフェイスまたはボード コンポーネント ピン オブジェクトの物理インターフェイス ポートを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 物理インターフェイス ポートを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトで定義されているコンポーネント インターフェイス ポートすべてのリストが取得されます。

例

次の例では、指定したボード コンポーネントのボード インターフェイス ファイルで定義されているボード インターフェイス ポートを取得しています。

```
get_board_interface_ports -of_objects \
    [get_board_component_interfaces *gmii*]
```

関連項目

- [current_board_part](#)
- [get_board_component_interfaces](#)
- [get_board_component_pins](#)
- [get_board_components](#)
- [get_boards](#)

get_board_ip_preferences

IP プリファレンス オブジェクトのリストを取得します。

構文

```
get_board_ip_preferences [-regexp] [-nocase] [-filter <arg>]
                          -of_objects <args> [-quiet] [-verbose] [<patterns>...]
```

戻り値

コンポーネントの IP プリファレンスのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
-of_objects	指定した board_component_mode、board_component_interface の ip_preference オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	IP プリファレンスを検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されている IP プリファレンスのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

IP プリファレンスは、ボード インターフェイス ファイルでコンポーネント インターフェイスに接続するのに適した IP を定義します。

このコマンドを実行すると、指定したコンポーネント インターフェイスまたはコンポーネント モードに適した IP のリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_ip_preferences` で返されたオブジェクトのリストに、IP プリファレンス (`ip_preference`) オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。`ip_preference` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IP_NAME`、`IP_VENDOR` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> (必須): 指定したボード コンポーネント モード (`board_component_mode`) またはボード コンポーネント インターフェイス (`board_component_interface`) オブジェクトの IP プリファレンスを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): IP プリファレンスを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したコンポーネント モードまたはコンポーネント インターフェイスで定義されている IP プリファレンスすべてのリストが取得されます。

例

次の例では、指定したコンポーネント インターフェイスの IP プリファレンスを取得しています。

```
get_board_ip_preferences -of_objects \  
  [get_board_component_interfaces *clock*]
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_jumpers

ボード ジャンパー オブジェクトのリストを取得します。

構文

```
get_board_jumpers [-regex] [-nocase] [-filter <arg>] [-of_objects <args>]
                  [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボードのジャンパーのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board の board_jumper オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ジャンパー名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regex を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されているジャンパーのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード ファイルで定義されているジャンパーは、ボードの特定のコンポーネント モードおよびインターフェイスをイネーブルにするために使用できます。

このコマンドを実行すると、ジャンパーのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_jumpers` で返されたオブジェクトのリストに、ジャンパーのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード ジャンパー オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、CURRENT_VALUE があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したボードのジャンパーを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ボード ジャンパーを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のボード上で定義されているボード ジャンパーすべてのリストが取得されます。

例

次の例では、指定したボードのボード インターフェイス ファイルで定義されているジャンパーを取得しています。

```
get_board_jumpers -of_objects [get_boards kc705]
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_parameters

ボード パラメーター オブジェクトのリストを取得します。

構文

```
get_board_parameters [-regexp] [-nocase] [-filter <arg>]
                    [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボードのパラメーターのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board、board_component、board_component_interface の board_parameter オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	パラメーター名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のボードのボード インターフェイス ファイルで定義されているパラメーターのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード ファイルで定義されているパラメーターは、ボードのカスタムまたはユーザー定義特性を指定します。

このコマンドを実行すると、ボード パラメーターのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_parameters` で返されたオブジェクトのリストに、パラメーター オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード パラメーター オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、VALUE、VALUE_TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> (オプション): 指定したボード、ボード コンポーネント、またはボード コンポーネント インターフェイスのパラメーターを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ボード パラメーターを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のボード上で定義されているボード パラメーターすべてのリストが取得されます。

例

次の例では、現在のボードのボード インターフェイス ファイルで定義されているパラメーターを取得しています。

```
get_board_parameters
```

関連項目

- [current_board_part](#)
- [get_board_buses](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)
- [get_boards](#)

get_board_part_interfaces

VLNV で指定されたバス定義をインプリメントする、ボード パーツのインターフェイスのリストを取得します。

構文

```
get_board_part_interfaces [-regexp] [-nocase] [-filter <arg>]  
[-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

バス インターフェイスのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board、board_component の board_component_interface オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	バス インターフェイスを検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のプロジェクトまたは開いているデザインで使用されているボードのザイリンクス デバイスまたは BOARD_PART プロパティで指定されている現在のボード パーツで定義されているインターフェイスのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード パーツは、ボード レベル システム内でのザイリンクス デバイスを表しており、ボード インターフェイス ファイルの part0 コンポーネントで表されます。current_board_part コマンドは、現在のプロジェクトで使用されているボード パーツを返します。

現在のボード パーツで定義されているインターフェイスは、関連する信号のグループを定義し、IP インテグレーターのブロック デザインのエレメントをすばやく接続したり、サイリンクス IP カタログから IP を設定する際に使用できます。現在のボード パーツ (current_board_part) で使用可能なインターフェイスは、create_bd_interface_port または create_bd_port コマンドを使用して IP サブシステム デザインに必要なインターフェイスを定義するのに使用するか、create_interface および create_port コマンドを使用して FPGA デザイン全体で使用可能なインターフェイスを定義するのに使用できます。

このコマンドを実行すると、現在のボード パーツで使用可能なインターフェイスのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp オプションを使用した場合にのみ適用されます。

-filter <args> (オプション): 結果のリストに指定した式のフィルターを適用します。-filter オプションを使用すると、get_board_part_interfaces で返されたオブジェクトのリストに、インターフェイスのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、report_property または list_property コマンドで確認できます。次に例を示します。

```
report_property -all [get_board_part_interfaces ddr3*]
```

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <args> (オプション): get_boards コマンドで指定したボード オブジェクトまたは get_board_components で指定したボード コンポーネントのインターフェイスを取得します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボード パーツ インターフェイスを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のプロジェクトまたはデザインで使用可能なすべてのインターフェイスのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてインターフェイスを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のボード パーツで定義されているインターフェイスすべてのリストを取得しています。

```
join [get_board_part_interfaces] \n
```

関連項目

- [create_interface](#)
- [current_board_part](#)
- [current_project](#)
- [get_boards](#)
- [get_board_components](#)
- [get_board_part_pins](#)
- [get_board_parts](#)

get_board_part_pins

ボード パーツ ピン オブジェクトのリストを取得します。

構文

```
get_board_part_pins [-regexp] [-nocase] [-filter <arg>]
                    [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボード パーツのピンのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した board_part_interface の board_part_pin オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード パーツ ピン名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のプロジェクトまたはデザインで使用されている現在のボード パーツのコンポーネント ピン オブジェクトのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、サイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード パーツは、ボード レベル システム内でのサイリンクス デバイスを表しており、ボード インターフェイス ファイルの part0 コンポーネントで表されます。current_board_part コマンドは、現在のプロジェクトで使用されているボード パーツを返します。

ボード パーツ ピンは、サイリンクス デバイスにインプリメントされているインターフェイスのコンポーネント ピンを表します。コンポーネント ピンには、LOC、IOSTANDARD、SLEW などのプロパティがあります。ボード パーツ ピンはスカラーまたはベクターで、常にビットで表されます。

ボード パーツ ピンは、`create_port` および `set_property PACKAGE_PIN` コマンドを使用して、最上位 FPGA デザインにポートを定義および配置するのに使用できます。

このコマンドを実行すると、コンポーネント ピンのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_board_part_pins` で返されたオブジェクトのリストに、ボード パーツ ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。次に例を示します。

```
report_property [get_board_part_pins RESET]
```

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したボード パーツ インターフェイス オブジェクトまたは現在のボード パーツのボード インターフェイス ポートのピン割り当てを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボード パーツ ピンを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のプロジェクトまたはデザインで使用可能なすべてのボード パーツ ピンのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてインターフェイスを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定したボード パーツ インターフェイスの物理ピンを取得しています。

```
get_board_part_pins -of [get_board_part_interfaces push_buttons_5bits]
```

次の例では、現在のボードの `leds_8bits` ピンのプロパティに基づいて、現在のデザインの指定したピンに `PACKAGE_PIN` および `IOSTANDARD` プロパティを割り当てています。

```
set_property PACKAGE_PIN [get_property LOC \
    [get_board_part_pins leds_8bits_TRI_O[1]]] [get_ports LEDS_n[1]]
set_property IOSTANDARD [get_property IOSTANDARD \
    [get_board_part_pins leds_8bits_TRI_O[1]]] [get_ports LEDS_n[1]]
```

次の例では、`leds_8bits` ボード パーツ インターフェイスに割り当てられているボード パーツ ピンのリストを取得し、それらのピンを Tcl 変数 `$boardPins` に保存して、それら各ピンの `LOC` プロパティを表示しています。

```
set boardPins [get_board_part_pins -of \
    [get_board_part_interfaces -filter {NAME == led_8bits}]]
foreach pin $boardPins {puts "The location of $pin is: \
    [get_property LOC $pin]"}
The location of leds_8bits_tri_o[0] is: AB8
The location of leds_8bits_tri_o[1] is: AA8
The location of leds_8bits_tri_o[2] is: AC9
The location of leds_8bits_tri_o[3] is: AB9
The location of leds_8bits_tri_o[4] is: AE26
The location of leds_8bits_tri_o[5] is: G19
The location of leds_8bits_tri_o[6] is: E18
The location of leds_8bits_tri_o[7] is: F16
```

関連項目

- [create_interface](#)
- [current_board_part](#)
- [current_project](#)
- [get_board_part_interfaces](#)
- [get_board_parts](#)

get_board_parts

プロジェクトで使用可能なボード パーツ (board_part) のリストを取得します。

構文

```
get_board_parts [-regex] [-nocase] [-latest_file_version]
                [-latest_hw_revision] [-filter <arg>] [-quiet] [-verbose]
                [<patterns>...]
```

戻り値

board_part オブジェクトのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-latest_file_version]	ファイル バージョンごとに最新のバージョンのみを表示します。
[-latest_hw_revision]	ボード リビジョンごとに最新のバージョンのみを表示します。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ボード パーツ名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regex を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[Board \(ボード\)](#)

説明

ボード インターフェイス ファイルで定義されている、現在のプロジェクトまたはデザインで使用可能なボード リポジトリ内のボード パーツのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダーにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

ボード パーツは、ボード レベル システム内でのザイリンクス デバイスを表しており、ボード インターフェイス ファイルの part0 コンポーネントで表されます。current_board_part コマンドは、現在のプロジェクトで使用されているボード パーツを返します。使用するボードの異なる定義方法は、current_board_part コマンドを参照してください。

このコマンドを実行すると、現在のボード リポジトリ定義されている、ボード インターフェイス ファイルで使用可能なザイリンクス デバイス (part0) のリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および **-filter** オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「**.**」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド **regexp** はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、**-regexp** オプションを使用した場合にのみ適用されます。

-latest_file_version (オプション): 最新のボード インターフェイス ファイルで定義されているボード パーツを返します。ボード インターフェイス ファイルには、複数のバージョンがある場合があります。このオプションを使用すると、最新バージョンのボード パーツのみが返されます。ボード インターフェイス ファイルの詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

-latest_hw_revision (オプション): ボード インターフェイス ファイルで示されているボードの互換性のある最新ハードウェア リビジョンで定義されているボード パーツを返します。ボード インターフェイス ファイルには、ボードの互換性のある複数のバージョンが示されている場合があります。このオプションを使用すると、最新バージョンのみが返されます。

-filter (オプション): 結果のリストに指定した式のフィルターを適用します。**-filter** オプションを使用すると、**get_board_parts** で返されたオブジェクトのリストに、ボード パーツのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、**report_property** または **list_property** コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード パーツ オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、PART_NAME、BOARD_NAME などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「**.**」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (! =~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボード パーツを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトで使用可能なすべてのボード パーツのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてボードを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定した検索パターンに一致するすべてのボード パーツを取得しています。

```
get_board_parts -filter {BOARD_NAME=~z*}
```

次の例では、指定した検索パターンに一致するすべてのボード パーツを取得しています。

```
get_board_parts {*av* *kc*}
```

関連項目

- [current_board_part](#)
- [get_board_part_interfaces](#)
- [get_board_part_pins](#)
- [list_property](#)
- [report_property](#)
- [set_property](#)

get_boards

プロジェクトで使用可能なボードのリストを取得します。

構文

```
get_boards [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
           [-quiet] [-verbose] [<patterns>...]
```

戻り値

ボード オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定した board_component の board オブジェクトを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	ボード名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[Board \(ボード\)](#)

説明

現在のプロジェクトで使用可能な評価ボードのリストを取得します。

Vivado Design Suite インストール エリアの data/boards フォルダにあるボード ファイル board.xml には、ボード属性に関する情報が保存されています。ボードは、ザイリンクス デバイスを含むシステム全体を表しており、クロック制約、I/O ポート割り当て、サポートされるインターフェイスなど、デザインの主要な部分を定義するのに役立ちます。カスタム ボード インターフェイス ファイルを使用して、カスタム ボードを定義できます。詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。

プロジェクトで使用されているボードを取得するには、current_board_part コマンドを使用します。

ボードは、次の方法で指定できます。

- プロジェクトを作成するときに New Project ウィザードの [Default Part] ページでボードを選択。
- 例に示すように現在のプロジェクトの BOARD プロパティを設定。
- Vivado IDE の [Settings] ダイアログ ボックスの [General] ページで [Project device] にボードを選択。

プロジェクトの作成方法およびプロジェクト設定の詳細は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) を参照してください。



重要: `set_property` コマンドでボードを指定すると、ターゲット デバイスも BOARD プロパティに合わせて変更されます。

このコマンドを実行すると、現在のプロジェクトで使用可能なボードのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_boards` で返されたオブジェクトのリストに、ボードのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ボード オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、DEVICE、FAMILY などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): `get_board_components` コマンドで指定したボード コンポーネント オブジェクトのボードを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ボードを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトで使用可能なすべてのボードのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてボードを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定の評価ボードのプロパティをレポートしています。

```
report_property [get_boards -filter {LIBRARY_NAME==artix7}]
```

次の例では、指定した検索パターンに一致するすべてのボードを取得しています。

```
get_boards {*ar* *kc*}
```

関連項目

- [current_board_part](#)
- [list_property](#)
- [report_property](#)
- [set_property](#)

get_cdc_violations

前回の report_cdc コマンド実行からの CDC 違反リストを取得します。

構文

```
get_cdc_violations [-name <arg>] [-regexp] [-filter <arg>] [-nocase]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

CDC 違反オブジェクトのリスト

使用法

名前	説明
[-name]	指定した名前の結果を取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	CDC 違反を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

Object (オブジェクト)

説明

report_cdc コマンドを実行したときにデザインで検出された違反オブジェクトのリストを取得します。個々の違反オブジェクトの詳細は、report_property または list_property コマンドを使用して取得します。

違反オブジェクトは、現在のデザインのクロック乗せ換えパスに関連付けられています。設計手法違反オブジェクトに関連付けられているデザイン オブジェクトは、該当する get_* コマンド (get_cells、get_nets など) で -of_objects オプションを使用して取得できます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-name <arg> (オプション): 指定した名前の CDC 結果セットの違反を取得します。このオプションを使用する場合、report_cdc コマンドを -name オプションを使用して実行する必要があります。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_cdc_violations` で返されたオブジェクトのリストに、違反のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 違反を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべての違反が取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて違反を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、現在のデザインの CDC 違反をレポートし、これらの違反のリストを取得しています。

```
report_cdc  
get_cdc_violations
```

次の例では、指定した名前の CDC レポートの違反をリストし、違反に関連付けられているピンを取得しています。

```
report_cdc -name cdc_1  
get_pins -of_objects [get_cdc_violations -name cdc_1]
```

関連項目

- [list_property](#)
- [report_cdc](#)
- [report_property](#)

get_cells

現在のデザインのセルのリストを取得します。

構文

```
get_cells [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]  
          [-of_objects <args>] [-match_style <arg>] [-include_replicated_objects]  
          [-quiet] [-verbose] [<patterns>]
```

戻り値

セル オブジェクトのリスト

使用法

名前	説明
[-hsc]	階層区切り文字を指定します。デフォルトはスラッシュ (/) です。
[-hierarchical]	すべての階層レベルで検索します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピン、タイミング パス、ネット、BEL、クロック領域、サイト、または DRC 違反のセルを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-include_replicated_objects]	パターンを検索する際に複製されたオブジェクトを含めます。このオプションは、patterns を指定した場合にのみ有効です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	セル名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

SDC、XDC、Object (オブジェクト)

説明

現在のデザインに含まれるセル オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、開いているデザインの `current_instance` コマンドで指定されている現在のインスタンスに含まれるすべてのセルのリストが返されます。

-hierarchical オプションを使用すると、現在のデザインのすべての階層からセルを取得できます。

`get_cells` コマンドには、物理最適化 (`phys_opt_design`) 中にデザインに追加された複製されたセルも取得するオプションがあります。 `-include_replicated_objects` オプションを使用すると、オブジェクトのオリジナルのセルと複製されたセルが返されます。このオプションを使用して、セルに適用された制約またはプロパティが複製されたセルにも適用されていることを確認できます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。 `lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-hsc <arg>` (オプション): 階層区切り文字を指定します。デフォルトの階層区切り文字は `/` です。

`-hierarchical` (オプション): 現在のインスタンスの階層レベルから開始し、デザイン階層のすべてのレベルからセルを取得します。このオプションを指定しない場合、現在のインスタンスの階層からのセルのみが取得されます。 `-hierarchical` を使用する場合、検索パターンは完全な階層セル名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして `「U1/」` を指定すると、名前に `「U1/」` を含むインスタンスが階層の各レベルで検索され、意図した結果が得られない可能性があります。これを下の例に示します。



重要: `-regexp` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして `「U1/」` を指定した場合に意図した結果が得られます。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に `「.*」` を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。 `-filter` オプションを使用すると、`get_cells` で返されたオブジェクトのリストに、セルのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。 `cell` オブジェクトの場合、結果をフィルター処理できるプロパティには `IS_PARTITION`、`IS_PRIMITIVE`、`IS_LOC_FIXED` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に `「.*」` を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が `""` のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価(==)、不等価(!=)、一致(=~)、不一致(!~)です。数値比較演算子<、>、<=、および>=も使用できます。複数のフィルター式をAND(&&)およびOR(||)で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型(boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <arg> (オプション): 指定したピン、タイミング パス、ネット、BEL、クロック領域、サイト、または DRC 違反オブジェクトに接続されているセルを取得します。

注記: -of_objects オプションでは、オブジェクトを名前で指定するのではなく、get_* コマンド (get_cells、get_pins など) を使用して指定する必要があります。また、-of_objects を検索パターン (<pattern>) と共に使用することはできません。

-match_style (オプション): 検索パターンが UCF 制約または SDC 制約に一致することを示します。デフォルトは SDC です。

-include_replicated_objects (オプション): 最適化での複製により追加されたセルも含めます。このオプションは、<patterns> を指定している場合にのみ有効で、パターンに一致する複製されたセル インスタンスを返します。デフォルトでは、get_cells コマンドで複製されたセルは返されません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): セルを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべてのセルのリストが返されます。複数の検索パターンを指定して、異なる検索条件に基づいてセルを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、BFT サンプル デザインに含まれるセルで NAME および REF_NAME が正規表現を使用したフィルター式に一致するものを取得しています。最初のコマンドでは検索パターンで NAME が指定されており、2 番目のコマンドでは NAME プロパティにフィルターを適用しています。これらのコマンドは、同じ結果を返します。

```
get_cells -regexp -filter { REF_NAME =~ FD.* } .*validFor.*
get_cells -regexp -filter { NAME =~ .*validFor.* && REF_NAME =~ FD.* }
```

次の例では、すべての階層レベルで cpu または fft で開始するセルを検索し、各セルの後に改行文字を付けて 1 行ごとに表示しています。

```
join [get_cells -hier {cpu* fft*}] \n
```

次の例では、`get_cells` で返された 2 番目のオブジェクトに設定されているプロパティとその値を取得しています。

```
report_property [lindex [get_cells] 1]
```

注記: 検索パターンに一致するセルがない場合は、警告メッセージが表示されます。

次の例では、デザインのすべての階層レベルにインスタンス化されているライブラリ セルのリストを表示しています。リストは名前順に並べられるので、各セルは 1 回のみ表示されます。

```
foreach cell [lsort -unique [get_property LIB_CELL [get_cells -hier \
-filter {IS_PRIMITIVE==1}]]] {puts $cell}
```

次の例では、`-regexp` を使用した場合と使用しない場合の `-hierarchical` オプションの効果を示しています。

```
get_cells -hierarchical *mmcm*
mmcm_replicator_inst_1
mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
get_cells -hierarchical -regexp .*mmcm.*
mmcm_replicator_inst_1
mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/GND
mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/MMCM_Base
```

注記: 最初の例 (`-regexp` なし) では検索パターンが階層の各レベルに適用されるため、最後の 2 つのセル (GND および MMCM_Base) は一致せず、返されません。

次の例では、現在のデザインに対して `report_drc` コマンドを実行し、DRC 違反に関連するセルを取得しています。

```
report_drc -name drc_1
get_cells -of_objects [get_drc_violations]
```

関連項目

- [current_instance](#)
- [get_lib_cells](#)
- [get_nets](#)
- [get_pins](#)
- [list_property](#)
- [phys_opt_design](#)
- [report_drc](#)
- [report_property](#)

get_cfgmem_parts

ツールで使用可能なコンフィギュレーション メモリ パーツ (cfgmem_part) のリストを取得します。

構文

```
get_cfgmem_parts [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]  
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

cfgmem_part オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した part、hw_device の cfgmem_part オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	コンフィギュレーション メモリ パーツ オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

Vivado Design Suite または Vivado Lab Edition でサポートされるコンフィギュレーション フラッシュ メモリ デバイスのリストを取得します。

ザイリンクス® FPGA は、外部不揮発性メモリ デバイスからコンフィギュレーションするか、マイクロプロセッサ、DSP プロセッサ、マイクロコントローラー、PC、ボード テスターなどの外部スマート ソースからコンフィギュレーションできます。コンフィギュレーション データパスには、コンフィギュレーションのデバイス ピン要件を最小限に抑えるために使用するシリアル データパスと、高パフォーマンス用あるいはプロセッサ、x8 または x16 パラレルフラッシュ メモリなどの外部データ ソースに適した業界標準インターフェイスに使用する 8 ビット、16 ビット、または 32 ビット データパスがあります。

ザイリンクス FPGA にデザイン特定のデータを読み込む (プログラムする) プロセスは、コンフィギュレーションと呼ばれます。create_hw_cfgmem を使用して、ハードウェア デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成し、ハードウェア デバイス (hw_device) に関連付けたり、write_cfgmem コマンドで作成したメモリ コンフィギュレーション ファイルからのビットストリームおよびその他のデータでコンフィギュレーション メモリをプログラムできます。hw_cfgmem オブジェクトをプログラムするには、program_hw_cfgmem コマンドを使用します。

コンフィギュレーション メモリ パーツは、Vivado Design Suite のハードウェア マネージャーのハードウェア コンフィギュレーション メモリを定義し、FPGA ハードウェアでのデザインのプログラムおよびデバッグを可能にします。現在のハードウェア デバイスで使用するコンフィギュレーション メモリを定義するには、create_hw_cfgmem コマンドを使用します。

このコマンドを実行すると、指定の検索条件に一致する cfgmem_part オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp オプションを使用した場合にのみ適用されます。

-filter <args> (オプション): 結果のリストに指定した式のフィルターを適用します。-filter オプションを使用すると、get_cfgmem_parts で返されたオブジェクトのリストに、cfgmem_parts オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、report_property または list_property コマンドで確認できます。cfgmem_part オブジェクトのリストをフィルター処理するのに使用できるプロパティには、COMPATIBLE_PARTS、DATA_WIDTH、MEM_DENSITY などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <arg> (オプション): 指定したパーツまたは hw_device オブジェクトの cfgmem_part を取得します。



重要: Vivado Lab Edition ではパーツ オブジェクトはサポートされず、hw_device オブジェクトのみがサポートされます。パーツまたは hw_device オブジェクトは、名前で指定するのではなく、`get_parts`、`get_hw_devices`、または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `cfgmem_part` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、Vivado Design Suite で現在使用可能なすべてのコンフィギュレーション メモリ パーツのリストが取得されます。

例

次の例では、現在の hw_device と互換性のある `cfgmem_part` を取得しています。

```
get_cfgmem_parts -of_objects [current_hw_device]
```

次の例では、ターゲット パーツと互換性のある `cfgmem_part` オブジェクトで、メモリ容量が指定の量以上であるもののリストを取得しています。

```
get_cfgmem_parts -of [get_parts [get_property PART [current_hw_device]]] \
-filter {MEM_DENSITY > 128}
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [delete_hw_cfgmem](#)
- [get_parts](#)
- [get_property](#)
- [program_hw_cfgmem](#)
- [set_property](#)
- [write_cfgmem](#)

get_clock_regions

現在のデバイスのクロック領域 (clock_region) を取得します。

構文

```
get_clock_regions [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

clock_region オブジェクト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したタイル、サイト、SLR、セル、またはパッケージ バンクの clock_region を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

ターゲット デバイスのクロック領域で、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、開いているデザインのデバイスのクロック領域すべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_clock_regions` で返されたオブジェクトのリストに、クロック領域のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。クロック領域オブジェクトの場合、結果をフィルター処理できるプロパティには `COLUMN_INDEX`、`HIGH_X`、`LOW_X` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`==~`」、不一致「`!=~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): `get_slrs` コマンドで指定した SLR に含まれるクロック領域、または指定したタイル、デバイス サイト、I/O バンク、セルのクロック領域を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン「`<pattern>`」と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): クロック領域を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、デバイスのクロック領域すべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてクロック領域を検索できます。

注記: 複数の検索パターンは波かっこ「`{ }`」またはダブルクォーテーション「`""`」で囲み、1 つのエレメントとして指定します。

例

次の例では、指定の検索パターンに一致するクロック領域が返されます。

```
get_clock_regions X0 *
```

次の例では、指定のプロパティが設定されているクロック領域が返されます。

```
get_clock_regions -filter {LOW_X==0}
```

注記: これらの 2 つの例では、同じクロック領域のリストが返されます。

次の例では、指定した ILA デバッグ トリガーが割り当てられている (配置されている) クロック領域が返されます。

```
get_clock_regions -of_objects [get_cells -hierarchical basic_trigger*]
```

関連項目

- [get_sites](#)
- [list_property](#)
- [report_property](#)

get_clocks

現在のデザインのクロックのリストを取得します。

構文

```
get_clocks [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
           [-match_style <arg>] [-include_generated_clocks] [-quiet] [-verbose]
           [<patterns>]
```

戻り値

クロックのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピン、ネット、またはセルのクロックを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-include_generated_clocks]	自動推論されたクロックおよび生成クロックも含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	クロック名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

SDC、XDC、Object (オブジェクト)

説明

現在のデザインに含まれるクロックで、検索パターンに一致するもののリストを取得します。デフォルトでは、all_clocks コマンドと同様に、デザインのすべてのクロックのリストが返されます。

クロックを作成するには、create_clock または create_generated_clock コマンドを使用するか、MMCM の出力などで、ツールで自動的に生成することもできます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` <args> (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_clocks` で返されたオブジェクトのリストに、クロックのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。クロック オブジェクトの場合、結果をフィルター処理できるプロパティには `PERIOD`、`WAVEFORM`、`IS_GENERATED` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <args> (オプション): 指定したセル、ピン、ポート、またはネット オブジェクトに接続されているクロックを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-match_style` (オプション): 検索パターンが UCF 制約または SDC 制約に一致することを示します。デフォルトは SDC です。

`-include_generated_clocks` (オプション): 生成クロックも含めたすべてのクロックで、指定の検索パターンに一致するものを返します。このオプションは、<patterns> を指定して指定のマスター クロックから生成クロックを取得する場合に使用してください。

注記: `get_generated_clocks` コマンドを使用すると、生成クロックのみを取得できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): クロックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべてのクロックが返されます。複数の検索パターンを指定して、異なる検索条件に基づいてクロックを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例は、複数の検索パターンに一致するクロックのリストを取得します。

```
get_clocks {*clock *ck *Clk}
```

注記: 検索パターンに一致するクロックがない場合は、警告メッセージが表示されます。

次の例は、マスター クロック オブジェクトと、そのクロックから生成クロックを取得します。

```
get_clocks -include_generated_clocks wbClk
```

次の例は、指定したクロックに設定されているプロパティとその値を取得します。

```
report_property -all [get_clocks wbClk]
```

関連項目

- [all_clocks](#)
- [create_clock](#)
- [create_generated_clock](#)
- [get_generated_clocks](#)
- [list_property](#)
- [report_property](#)

get_dashboard_gadgets

プロジェクト サマリ ダッシュボードを作成します。

構文

```
get_dashboard_gadgets [-quiet] [-verbose] [<patterns>...]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	ガジェット名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) です。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)

説明

現在のプロジェクトに含まれるダッシュボード ガジェットのリストを取得します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ダッシュボード ガジェットを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのダッシュボード ガジェットが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてダッシュボード ガジェットを検索できます。

例

次の例では、現在のプロジェクトに含まれるダッシュボード ガジェットを取得しています。

```
get_dashboard_gadgets
```

関連項目

- [create_dashboard_gadget](#)

- [delete_dashboard_gadgets](#)
- [move_dashboard_gadget](#)

get_debug_cores

現在のデザインのデバッグ コアのリストを取得します。

構文

```
get_debug_cores [-filter <arg>] [-of_objects <args>] [-regexp] [-nocase]
                [-quiet] [-verbose] [<patterns>]
```

戻り値

デバッグ コア オブジェクトのリスト

使用法

名前	説明
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したデバッグ ポートまたはネットのコアを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	デバッグ コアを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Debug \(デバッグ\)](#)、[XDC](#)

説明

現在のプロジェクトの Vivado Lab Edition デバッグ コアで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトのデバッグ コアすべてのリストが取得されます。

デバッグ コアは、`create_debug_core` コマンドを使用してプロジェクトに追加できます。ILA デバッグ コア (`labtools_ila_v3`) をプロジェクトに追加すると、デバッグ ハブ コア (`labtools_xsdbmasterlib_v2`) 内に含まれ、CLK ポートと PROBE ポートがデフォルトで含まれます。`create_debug_port` コマンドを使用して、デバッグ コアにポートを追加することもできます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_debug_cores` で返されたオブジェクトのリストに、デバッグ コアのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したデバッグ ポートまたはネットに接続されている ILA デバッグ コアを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): デバッグ コアを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのデバッグ コアが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてデバッグ コアを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のプロジェクトの Vivado Lab Edition デバッグ コアのリストを取得しています。

```
get_debug_cores
```

注記: デバッグ ハブ コアがプロジェクトのデバッグ コアの 1 つとして返されます。このコアは直接作成できませんが、ILA コアをプロジェクトに追加すると自動的に追加されます。

次の例では、指定したデバッグ コアのプロパティを取得しています。

```
report_property [get_debug_cores myCore]
```

関連項目

- [create_debug_core](#)
- [create_debug_port](#)
- [get_debug_ports](#)
- [list_property](#)
- [report_property](#)
- [set_property](#)

get_debug_ports

現在のデザインのデバッグ ポートのリストを取得します。

構文

```
get_debug_ports [-filter <arg>] [-of_objects <args>] [-regexp] [-nocase]  
                [-quiet] [-verbose] [<patterns>]
```

戻り値

デバッグ ポート オブジェクトのリスト

使用法

名前	説明
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したデバッグ コアのポートを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	デバッグ ポートを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Debug \(デバッグ\)](#)、[XDC](#)

説明

現在のプロジェクトの ILA デバッグ コアで定義されているポートで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトのデバッグ ポートすべてのリストが取得されます。

デバッグ ポートは、`create_debug_core` コマンドで ILA デバッグ コアを作成すると定義されます。`create_debug_port` コマンドを使用して、ポートを既存のデバッグ コアに追加することもできます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_debug_ports` で返されたオブジェクトのリストに、デバッグ ポートのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。デバッグ ポートオブジェクトのリストをフィルター処理するのに使用できるプロパティには、`PORT_WIDTH`、`MATCH_TYPE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を `AND` («`&&`») および `OR` («`||`») で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したデバッグ コアのデバッグ ポートを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン («`<pattern>`») と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン («`<patterns>`») および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): デバッグ ポートを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのデバッグ ポートが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてデバッグ ポートを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のプロジェクトの ILA デバッグ コアのポートで、PORT_WIDTH プロパティが 8 のもののリストを取得しています。

```
get_debug_ports -filter {PORT_WIDTH==8}
```

次の例では、指定したデバッグ ポートに設定されているプロパティを取得しています。

```
report_property [get_debug_ports myCore/PROBE1]
```

注記: デバッグ ポートは、core_name/port_name の形式で指定します。

関連項目

- [create_debug_core](#)
- [create_debug_port](#)
- [list_property](#)
- [report_property](#)

get_designs

現在のプロジェクトに含まれるデザインのリストを取得します。

構文

```
get_designs [-regex] [-nocase] [-filter <arg>] [-quiet] [-verbose]
[<patterns>]
```

戻り値

デザイン オブジェクトのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	デザイン名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

現在のプロジェクトで開いているデザインで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトで開いているデザインすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regex (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regex はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regex.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_designs` で返されたオブジェクトのリストに、デザインのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。デザイン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`CONSTRSET`、`PART` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): デザインを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのデザインが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてデザインを検索できます。

例

次の例では、現在のプロジェクトで開いているデザインすべてのリストを取得しています。

```
get_designs
```

次の例では、検索パターンに一致するデザインに設定されているプロパティを取得しています。

```
report_property [get_designs r*]
```

注記: 検索パターンに一致するデザインがない場合は、警告メッセージが表示されます。

関連項目

- [report_property](#)

get_drc_checks

DRC ルール チェック オブジェクトのリストを取得します。

構文

```
get_drc_checks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-abbrev <arg>] [-ruledecks <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

DRC ルール チェック オブジェクトのリスト

使用法

名前	説明
[-of_objects]	指定した drc_ruledeck の rule_check オブジェクトを追加します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-abbrev]	指定した略称のルールで ID が最大のものを取得します。
[-ruledecks]	DRC ルール チェックのコンテナであるルール デックを指定します。デフォルトは default です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	rule_check オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

現在定義済みの DRC チェックのリストを取得します。リストには、デフォルトであらかじめ定義されているデザイン ルール チェックと、create_drc_check コマンドで作成したユーザー定義チェックの両方が含まれます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-of_objects <args> (オプション): 指定したルール デック オブジェクトのデザイン ルール チェックを取得します。ルール デック オブジェクトは、get_drc_ruledecks コマンドを使用して指定する必要があります。

注記: `-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_drc_checks` で返されるルール チェックのリストに、プロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (! =~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-abbrev <arg>` (オプション): 指定したルール名または略称のデザイン ルール チェックを取得します。

`-ruledecks <args>` (オプション): 指定したルール デックのデザイン ルール チェックを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): デザイン ルール チェックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのルール チェックが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてルール チェックを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、すべての AVAL デザイン ルール チェックのリストを取得しています。

```
get_drc_checks AVAL*
```

次の例では、指定したルール デッキのチェックを取得しています。

```
get_drc_checks -of_objects [get_drc_ruledecks placer_checks]
```

関連項目

- [create_drc_check](#)
- [get_drc_ruledecks](#)
- [list_property](#)
- [report_drc](#)
- [report_property](#)

get_drc_ruledecks

DRC ルール デック オブジェクトのリストを取得します。

構文

```
get_drc_ruledecks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

drc_ruledeck オブジェクト

使用法

名前	説明
[-of_objects]	指定した rule_check の drc_ruledeck オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	drc_ruledeck オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

report_drc コマンドで使用するため、現在定義されているルール デックのリストを取得します。

ルール デックはデザイン ルール チェックのグループで、I/O プランニングや配置などの FPGA デザイン フローの異なる段階で report_drc コマンドにより実行されます。ツールには定義済みのルール デックが含まれていますが、create_drc_ruledeck コマンドを使用して新しいユーザー定義のルール デックを作成し、add_drc_checks コマンドを使用してチェックを追加できます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-of_objects <args> (オプション): 指定したルール チェック オブジェクトの DRC ルール デックを取得します。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_drc_ruledecks` で返されたオブジェクトのリストに、ルール デックのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ルール デックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのルール デックが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてルール デックを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、現在のプロジェクトで定義されているすべてのルール チェックのリストを取得しています。

```
get_drc_ruledecks
```

次の例では、placer_checks ルール デックの各チェックを個別の行でリストしています。

```
foreach rule [get_drc_checks -of_objects \  
  [get_drc_ruledecks placer_checks]] {puts $rule}
```

関連項目

- [add_drc_checks](#)
- [create_drc_ruledeck](#)
- [list_property](#)
- [report_drc](#)
- [report_property](#)

get_drc_violations

前回の report_drc コマンド実行からの DRC 違反のリストを取得します。

構文

```
get_drc_violations [-name <arg>] [-regexp] [-filter <arg>] [-nocase]  
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

DRC 違反オブジェクトのリスト

使用法

名前	説明
[-name]	指定した名前の結果を取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	DRC 違反を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

report_drc コマンドを実行したときにデザインで検出された違反オブジェクトのリストを取得します。違反オブジェクトは、DRC を実行したときに、Vivado ツールの内部デザイン ルール チェックまたは create_drc_violation コマンドで作成したユーザー定義 DRC チェックにより作成されます。個々の違反オブジェクトの詳細は、report_property または list_property コマンドを使用して取得します。

違反オブジェクトは、現在のデザインのセル、ネット、ピン、ポート、または現在のデバイス上のサイトに関連付けられています。DRC 違反オブジェクトに関連付けられているデザイン オブジェクトは、該当する get_* コマンド (get_cells、get_nets など) で -of_objects オプションを使用して取得できます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-name <arg>` (オプション): 指定した DRC 結果セットの違反を取得します。このオプションを使用する場合、`report_drc` コマンドを `-name` オプションを使用して実行している必要があります。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_drc_violations` で返されたオブジェクトのリストに、違反のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 違反を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべての違反が取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて違反を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のデザインの DRC 違反をレポートし、これらの違反のリストを取得しています。

```
report_drc
get_drc_violations
```

次の例では、指定した DRC 違反のプロパティを取得しています。

```
report_property [lindex [get_drc_violations] 0]
```

次の例では、現在のデザインの指定した DRC レポートに含まれる違反のリストを取得し、未指定の I/O 規格ルール (NSTD) の違反に関連付けられているポートを取得しています。

```
get_drc_violations -name drc_1
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

関連項目

- [create_drc_check](#)
- [create_drc_violation](#)
- [get_cells](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)
- [get_sites](#)
- [report_drc](#)

get_example_designs

サンプル デザインのリストを取得します。

構文

```
get_example_designs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
[<patterns>...]
```

戻り値

デザイン オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	検索パターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または <code>-regexp</code> を指定している場合は「.*」です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

Vivado Design Suite の現在のリリースで使用可能なサンプル デザインのリストを返します。正常に実行されなかった場合はエラーを返します。

サンプル デザインは、開いたり、Vivado Design Suite にインスタンスエートしたり、指定のターゲット デバイスまたはボードにプログラムできます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_example_designs` で返されたオブジェクトのリストに、デザインのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。サンプル デザイン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、SUPPORTED_BOARDS、SUPPORTED_PARTS などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): サンプル デザインを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、サポートされるすべてのサンプル デザインが取得されます。

例

次の例では、現在のリリースに含まれるサンプル デザインのプロパティをレポートしています。

```
foreach x [get_example_designs] {
  puts "***** Design $x *****"
  report_property -all $x}
```

注記: レポートされるプロパティには、サンプル デザインと互換性のあるパーツを示す PARTS プロパティが含まれます。

関連項目

- [instantiate_example_design](#)

get_files

ソース ファイルのリストを取得します。

構文

```
get_files [-regex] [-nocase] [-filter <arg>] [-compile_order <arg>]
          [-used_in <arg>] [-references] [-all] [-of_objects <args>] [-quiet]
          [-verbose] [<patterns>]
```

戻り値

ファイル オブジェクトのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-compile_order]	タイプおよびコンパイル順でファイルを取得します。-used_in と共に使用する必要があります。
[-used_in]	特定の手順のファイルを取得します。-compile_order と共に使用する必要があります。
[-references]	指定したオブジェクトで参照されているファイルを取得します。-of_objects と共に使用する必要があります。
[-all]	すべての内部ファイルを含めます。
[-of_objects]	指定したファイル、ファイルセット、IP、リコンフィギュラブル モジュール (reconfig_module) のファイル オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ファイル名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)

説明

現在のプロジェクトに含まれるファイルで、検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトのファイルすべてのリストが取得されます。

get_files コマンドは、プロジェクト、デザイン、または IP コアやブロック デザインなどのサブデザインに含まれるファイルのコンピューターで解読可能なリストを返します。get_files で返された結果は、-of_objects、-compile_order、-used_in、-filter などのコマンド オプションを使用してフィルターできます。

合成、シミュレーション、またはインプリメンテーションで使用するソースまたは制約ファイルセットを Vivado ツールで評価された順序で並べられたリストを取得するには、`-compile_order` と `-used_in` オプションを共に使用する必要があります。`-compile_order` および `-used_in` オプションを使用すると、複雑なファイル順の規則が使用され、ヘッダー ファイル、インクルード ファイル、またはその他の言語オプションによって異なることがあります。`get_files` コマンドで返された結果を、`-used_in` オプションではなく、`-filter` オプションを使用して `USED_IN` プロパティに基づいてフィルターすることもできます。

`report_compile_order` コマンドを使用すると、合成、シミュレーション、およびインプリメンテーションで使用するソースまたは制約ファイルセットのすべてのファイルがセクション別にリストされたレポートが生成されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_files` で返されたオブジェクトのリストに、ファイルのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ファイル オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`FILE_TYPE`、`IS_ENABLED` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したファイル、ファイルセット、IP、またはリコンフィギュラブル モジュール (reconfig_module) オブジェクトに関連付けられているファイルを取得します。デフォルトでは、すべてのファイルセットが検索されます。`-compile_order` および `-used_in` オプションを指定した場合、`-of_objects` オプションで指定できるのは 1 つのファイルセット、または IP コア、ブロック デザイン、DSP デザインなどの 1 つのサブデザインのみです。サブデザインは、複合ファイルとも呼ばれます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-compile_order [sources | constraints]` (オプション): ソース デザイン ファイルまたは制約ファイルを、Vivado Design Suite によるコンパイル順で返します。

注記: このオプションは `-used_in` オプションと共に使用する必要があります。これらのオプションを指定した場合、`-of_objects` オプションで指定できるのは 1 つのファイルセットまたはサブデザインのみです。

`-used_in <arg>` (オプション): ファイルの `USED_IN` プロパティの値の 1 つを指定し、その値に一致するファイルを返します。指定可能な値は、`synthesis`、`simulation`、`implementation` などです。`USED_IN` プロパティおよびサポートされる値は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) を参照してください。

注記: このオプションは `-compile_order` オプションと共に使用する必要があります。これらのオプションを指定した場合、`-of_objects` オプションで指定できるのは 1 つのファイルセットまたはサブデザインのみです。

`-all` (オプション): IP およびその他のオブジェクトをサポートする内部ファイルも含め、デザインのすべてのファイルを返します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ファイルを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべてのファイルが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてファイルを検索できます。

例

次の例では、シミュレーションで使用される VHDL ファイルが返されます。

```
get_files -filter {FILE_TYPE == VHDL && USED_IN =~ simulation*}
```

次の例では、シミュレーションで使用されるデザイン ソース ファイルが返されます。

```
get_files -compile_order sources -used_in simulation
```

次の例では、合成で使用される `sources_1` ファイルセットから、指定の IP コア複合ファイル (`ip.xci`) に関連付けられているファイルのリストを取得しています。

```
get_files -compile_order sources -used_in synthesis -of [get_files ip.xci]
```

次の例では、sources_1 および constrs_1 ファイルセットに含まれるファイルのリストを取得しています。

```
get_files -of [get_filesets {sources_1 constrs_1}]
```

注記: 検索パターンに一致するファイルがない場合は、警告メッセージが表示されます。

関連項目

- [report_compile_order](#)
- [report_property](#)

get_filesets

現在のプロジェクトのファイルセットのリストを取得します。

構文

```
get_filesets [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
              [-quiet] [-verbose] [<patterns>]
```

戻り値

ファイルセット オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定したリコンフィギャラブル モジュール (<code>reconfig_module</code>) のファイルセット オブジェクトを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	ファイルセット名を検索するパターンを指定します。デフォルトは <code>*</code> です。

カテゴリ

Object (オブジェクト)、Project (プロジェクト)

説明

現在のプロジェクトに含まれるファイルセットで、検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトのファイルセットすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に `['*]` を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_filesets` で返されたオブジェクトのリストに、ファイルセットのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ファイルセット オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DESIGN_MODE`、`FILESET_TYPE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND 「`&&`」および OR 「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したリコンフィギュラブル モジュール (`reconfig_module`) オブジェクトに関連付けられているファイルセットを取得します。デフォルトでは、すべてのファイルセットが検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ファイルセットを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、すべてのファイルセットが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてファイルセットを検索できます。

例

次の例では、ソース セットのソース ファイルが返されます。

```
get_files -of_objects [get_filesets sources_1]
```

次の例では、`project_2` をアクティブ プロジェクトに指定した後、`s` または `r` で始まるファイルセットのリストを取得します。

```
current_project project_2
get_filesets s* r* -quiet
```

注記: 検索パターンに一致するファイルセットがない場合、通常は警告メッセージが表示されますが、上記の例では `-quiet` オプションが指定されているので、警告メッセージは表示されません。

次の例では、大文字/小文字を区別せずに、`C` で始まるファイルセットを取得しています。

```
get_filesets C.* -regexp -nocase
```

この例では、`constrs_1` および `constrs_2` 制約セットが現在のプロジェクトで定義されていれば返されます。

関連項目

- [get_files](#)
- [report_property](#)

get_generated_clocks

現在のデザインに含まれる生成クロックのリストを取得します。

構文

```
get_generated_clocks [-regex] [-nocase] [-filter <arg>]
                    [-of_objects <args>] [-match_style <arg>] [-quiet] [-verbose]
                    [<patterns>]
```

戻り値

クロックのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピン、ポート、またはネットの生成クロックを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	生成クロック名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

XDC、Object (オブジェクト)

説明

現在のプロジェクトに含まれる生成クロックで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトに含まれる生成クロックすべてのリストが取得されます。

生成クロックは、create_generated_clock を使用してデザインに追加するか、MMCM の出力でなど、ツールで自動的に生成することもできます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>`: 結果のリストに指定した式のフィルターを適用します。 `-filter` オプションを使用すると、`get_generated_clocks` で返されたオブジェクトのリストに、生成クロックのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。生成クロック オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DUTY_CYCLE`、`MASTER_CLOCK` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したポート、ピン、またはネット オブジェクトに接続されている生成クロックを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-match_style` (オプション): 検索パターンが UCF 制約または SDC 制約に一致することを示します。デフォルトは SDC です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 生成クロックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべての生成クロックが取得されます。

例

次の例は、現在のプロジェクトに含まれる生成クロックすべてのリストを取得します。

```
get_generated_clocks
```

関連項目

- [create_generated_clock](#)
- [get_clocks](#)
- [report_property](#)

get_gui_custom_command_args

カスタム コマンド引数を取得します。

構文

```
get_gui_custom_command_args -command_name <arg> [-regexp] [-nocase]  
[-quiet] [-verbose] [<patterns>...]
```

戻り値

カスタム コマンド引数名のリスト

使用法

名前	説明
-command_name	引数を表示するカスタム コマンドの名前を指定します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	カスタム コマンド引数名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

特定のカスタム GUI コマンドのカスタム GUI コマンド引数を返します。

`get_gui_custom_commands` を使用すると、定義済みのカスタム コマンドを取得できます。

引数

-command_name: 引数を取得するカスタム GUI コマンドの名前を指定します。

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): カスタム GUI コマンド引数を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのカスタム GUI コマンド引数が返されます。

例

次の例は、カスタム GUI コマンド `abc` のコマンド引数のリストを返します。

```
get_gui_custom_command_args -command_name abc
```

関連項目

- [create_gui_custom_command](#)
- [create_gui_custom_command_arg](#)
- [get_gui_custom_commands](#)

get_gui_custom_commands

カスタム コマンドを取得します。

構文

```
get_gui_custom_commands [-regexp] [-nocase] [-quiet] [-verbose]  
[<patterns>...]
```

戻り値

カスタム コマンド名のリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	カスタム コマンド名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

GUIControl (GUI 制御)

説明

カスタム GUI コマンドのリストを返します。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp オプションを使用した場合にのみ適用されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): カスタム GUI コマンドを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在定義されているすべてのカスタム GUI コマンドが返されます。

例

次の例は、名前が p で始まるカスタム GUI コマンドのリストを返します。

```
get_gui_custom_commands p*
```

関連項目

- [create_gui_custom_command](#)
- [create_gui_custom_command_arg](#)
- [get_gui_custom_command_args](#)

get_hierarchy_separator

階層区切り文字を取得します。

構文

```
get_hierarchy_separator [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[SDC](#)、[XDC](#)

説明

デザインで現在設定されている階層区切り文字を取得します。階層区切り文字を設定するには、`set_hierarchy_separator` コマンドを使用します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在定義されている階層区切り文字を取得しています。

```
get_hierarchy_separator
```

関連項目

- [set_hierarchy_separator](#)

get_highlighted_objects

ハイライトされたオブジェクトを取得します。

構文

```
get_highlighted_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
                        [-quiet] [-verbose]
```

戻り値

ハイライトされたオブジェクトのリスト

使用法

名前	説明
<code>[-color_index]</code>	色を色インデックスで指定します。
<code>[-rgb]</code>	色を RGB で指定します。
<code>[-color]</code>	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)、GUIControl (GUI 制御)

説明

Vivado IDE で開いている現在のデザインでハイライトされているオブジェクトを取得します。オブジェクトをハイライトを解除するには、`highlight_objects` コマンドを使用します。

デザインでハイライトされているオブジェクトをすべて取得するか、指定した名前、インデックス、または RGB 値の色でハイライトされているオブジェクトを取得できます。

注記: この Tcl コマンドは、Vivado を GUI モードで実行している場合にのみ機能します。

引数

`-color_index <arg>` (オプション): 有効な値は 1 ~ 19 の整数で、オブジェクトのハイライト色を色インデックスで指定します。色インデックスは、[Tools] → [Settings] をクリックし、[Settings] ダイアログ ボックスの [Colors] → [Highlight] ページで定義します。テーマの設定方法の詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

`-rgb <args>` (オプション): オブジェクトのハイライト色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定します。

`-color <arg>` (オプション): オブジェクトのハイライト色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

注記: 白は `select_objects` コマンドで指定したオブジェクトを表示するのに使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインでハイライトされているオブジェクトすべてを取得しています。

```
get_highlighted_objects
```

次の例では、現在のデザインに含まれるオブジェクトで、指定の色でハイライトされているものを取得しています。

```
get_highlighted_objects -color cyan
```

関連項目

- [get_marked_objects](#)
- [get_selected_objects](#)
- [highlight_objects](#)
- [mark_objects](#)
- [select_objects](#)

get_hw_axi_txns

ハードウェア AXI トランザクションのリストを取得します。

構文

```
get_hw_axi_txns [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                [-quiet] [-verbose] [<patterns>]
```

戻り値

hw_axi_txn オブジェクト

使用法

名前	説明
[-of_objects]	指定した hw_axi の hw_axi_txn を取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_axi_txn を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

指定した JTAG to AXI Master コア (hw_axi オブジェクト) の読み出しまたは書き込みトランザクションを取得します。

JTAG to AXI Master コアは、AXI トランザクションを駆動し、ハードウェア デバイス内の AXI 信号を駆動する AXI マスターとして機能するカスタマイズ可能な IP コアです。この IP は、Vivado® IP インテグレーターで使用するか、Vivado プロジェクトの HDL にインスタンスエートできます。

JTAG to AXI Master コアは、AXI4-Stream を除くメモリ マップ AXI インターフェイスをすべてサポートし、AXI4-Lite プロトコルをサポートします。AXI インターフェイスをコアのプロパティとして選択できます。AXI データ バスの幅はカスタマイズ可能です。この IP は AXI4-Lite またはメモリ マップ スレーブを直接駆動でき、また AXI マスターとしてインターコネクに接続することもできます。このコアにランタイムにアクセスするには、Vivado ロジック解析機能を使用する必要があります。JTAG to AXI Master コアの詳細は、『LogiCORE IP JTAG to AXI Master 製品ガイド』(PG174) を参照してください。この使用法は、『Vivado Design Suite チュートリアル: プログラムおよびデバッグ』(UG936) を参照してください。

JTAG to AXI Master コアは、ザイリンクス IP カタログから RTL コードにインスタンス化する必要があります。AXI トランザクションは、AXI マスターとさまざまなスレーブとの間の完全な読み出しまたは書き込みトランザクションとして定義されます。

このコマンドを実行すると、ハードウェア デバイス (hw_device) 上のハードウェア AXI トランザクション (hw_axi_txn) オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア オブジェクトの AXI マスター コアを取得します。デバイスは、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_axi_txns` で返されたオブジェクトのリストに、AXI トランザクションのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。hw_axi_txn オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、TYPE があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_axi` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_axi` のリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_axi_txn -filter {TYPE==WRITE} [get_hw_axis]
```

関連項目

- [delete_hw_axi_txn](#)
- [get_hw_axis](#)
- [refresh_hw_axi](#)
- [reset_hw_axi](#)

get_hw_axis

ハードウェア AXI オブジェクトのリストを取得します。

構文

```
get_hw_axis [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
            [-quiet] [-verbose] [<patterns>]
```

戻り値

hw_axi オブジェクト

使用法

名前	説明
[-of_objects]	指定した hw_device の hw_axi オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_axi オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

次の例では、現在のハードウェア デバイスで定義されている JTAG to AXI Master コア オブジェクト (hw_axi オブジェクト) を取得します。

JTAG to AXI Master コアは、AXI トランザクションを駆動し、ハードウェア デバイス内の AXI 信号を駆動する AXI マスターとして機能するカスタマイズ可能な IP コアです。この IP は、Vivado® IP インテグレーターで使用するか、サイリンクス® IP カタログから Vivado プロジェクトの HDL にインスタンス化できます。このコアにランタイムにアクセスするには、Vivado ロジック解析機能を使用する必要があります。

JTAG to AXI Master コアは、AXI4-Stream を除くメモリ マップ AXI インターフェイスをすべてサポートし、AXI4-Lite プロトコルをサポートします。AXI インターフェイスをコアのプロパティとして定義できます。AXI データ バスの幅はカスタマイズ可能です。この IP は AXI4-Lite またはメモリ マップ スレーブを直接駆動でき、また AXI マスターとしてインターコネクに接続することもできます。JTAG to AXI Master コアの詳細は、『LogiCORE IP JTAG to AXI Master 製品ガイド』 (PG174) を参照してください。この使用法は、『Vivado Design Suite チュートリアル: プログラムおよびデバッグ』 (UG936) を参照してください。

AXI コアは、`get_debug_cores` コマンドを使用して合成済みまたはインプリメント済みデザイン ネットリスト デザインで検索できます。これにより、`hw_axi` オブジェクトに関連するデザインに含まれるデバッグ コアのインスタンスと、ハードウェア デバイス (`hw_device`) 上のその他のデバッグ コアが返されます。

このコマンドを実行すると、`hw_device` 上の AXI マスター コア オブジェクト (`hw_axi`) のリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア オブジェクトの AXI マスター コアを取得します。デバイスは、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_axis` で返されたオブジェクトのリストに、AXI マスター コアのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ハードウェア AXI オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、PROTOCOL があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_axi` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_axi` のリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_axis -of_objects [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [program_hw_devices](#)
- [set_property](#)

get_hw_cfgmems

ハードウェア コンフィギュレーション メモリのリストを取得します。

構文

```
get_hw_cfgmems [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]  
[<patterns>]
```

戻り値

ハードウェア コンフィギュレーション メモリ

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_cfgmem オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス (hw_device) に作成されたハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを取得します。

ザイリンクス FPGA をコンフィギュレーションするには、ハードウェア デバイス (hw_device) の内部メモリにデザイン特定のコンフィギュレーション データをビットストリーム ファイルの形で読み込みます。hw_cfgmem は、ザイリンクス FPGA デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。create_hw_cfgmem オブジェクトを作成し、ハードウェア デバイスに関連付けたら、program_hw_cfgmem コマンドを使用してコンフィギュレーション メモリをビットストリームおよびその他のデータでプログラムできます。

hw_cfgmem オブジェクトは、デバイス オブジェクトの PROGRAM.HW_CFGMEM プロパティを使用して指定のハードウェア デバイス オブジェクトに関連付けられます。get_hw_cfgmems コマンドを使用すると、hw_cfgmem オブジェクトを操作できます。

このコマンドを実行すると、指定の検索条件に一致する hw_cfgmem オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_cfgmems` で返されたオブジェクトのリストに、`hw_cfgmem` オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_cfgmem` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在定義されている `hw_cfgmem` すべてが取得されます。

例

次の例では、現在の hw_device に関連付けられている hw_cfgmem オブジェクトを取得しています。

```
get_hw_cfgmems
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [delete_hw_cfgmem](#)
- [get_parts](#)
- [get_property](#)
- [program_hw_cfgmem](#)
- [set_property](#)
- [write_cfgmem](#)

get_hw_ddrmcs

Versal 統合およびソフト DDRMC コアのリストを取得します。

構文

```
get_hw_ddrmcs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
               [-quiet] [-verbose] [<patterns>]
```

戻り値

統合およびソフト DDRMC コア

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device の hw_ddrmc オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_ddrmc オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

ザイリンクス Versal デバイスには、統合 DDR メモリ コントローラー (DDRMC) とファブリック ベースのメモリ コントローラーが含まれます。統合コントローラー コアの詳細は、『Versal ACAP Programmable Network on Chip および Integrated Memory Controller LogiCORE IP 製品ガイド』 (PG313) を参照してください。ファブリック ベースのメモリ コントローラーの詳細は、『Versal ACAP Soft DDR4 SDRAM Memory Controller LogiCORE IP 製品ガイド』 (PG353) を参照してください。メモリ コントローラーには、ザイリンクス Vivado ハードウェア マネージャーからアクセス可能なメモリ設定およびメモリ キャリブレーション データが含まれています。キャリブレーションおよびマージン データは、解析しやすいようにグラフィカルに表示されます。

get_hw_ddrmc コマンドは、現在のハードウェア デバイス上のメモリ コアのリストを返します。

引数

-of_objects <arg> (オプション): 指定したハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、またはハードウェア デバイス (hw_device) オブジェクトの Versal メモリ コントローラー (DDRMC) オブジェクトを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_ddrmcs` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_ddrmc` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`MEM_TYPE`、`BITS_PER_BYTE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_ddrmc` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_ddrmc` オブジェクトのリストが取得されます。

例

次の例は、指定した DISPLAY_NAME の HBM を返します。

```
get_hw_hbms -filter {DISPLAY_NAME == HBM_2}
```

次の例は、指定した DISPLAY_NAME の DDRMC を返します。

```
get_hw_ddrmcs -filter {DISPLAY_NAME == DDRMC_1}
```

関連項目

- [current_hw_device](#)
- [current_hw_target](#)
- [refresh_hw_ddrmc](#)
- [report_hw_ddrmc](#)

get_hw_devices

ハードウェア デバイスのリストを取得します。

構文

```
get_hw_devices [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
[-of_objects]	指定した hw_target の hw_device オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_device オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

接続されたハードウェア サーバーで開いているターゲット上のハードウェア デバイスを返します。

各ハードウェア ターゲットには、プログラムまたはデバッグ目的で使用するサイリンクス デバイスを 1 つまたは複数含めることができます。current_hw_device コマンドは、現在のデバイスを指定するか、返します。

このコマンドを実行すると、使用可能なハードウェア デバイス オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-of_objects <arg> (オプション): 指定したハードウェア ターゲットのハードウェア デバイスを取得します。ターゲットは、get_hw_targets または current_hw_target コマンドを使用してオブジェクトとして指定する必要があります。

注記: -of_objects オプションでは、オブジェクトを名前で指定するのではなく、get_* コマンド (get_cells、get_pins など) を使用して指定する必要があります。また、-of_objects を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_devices` で返されたオブジェクトのリストに、ハードウェア デバイス オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ハードウェア デバイス (`hw_device`) オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、PART があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_device` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、接続されているハードウェア サーバーで使用可能なすべての `hw_device` のリストが取得されます。

例

次の例では、現在のハードウェア ターゲットで使用可能なハードウェア デバイス オブジェクトのリストを取得しています。

```
get_hw_devices -of_objects [current_hw_target]
```

関連項目

- [connect_hw_server](#)
- [create_hw_device](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_targets](#)
- [open_hw_target](#)

get_hw_hbms

ハードウェア HBM コアのリストを取得します。

構文

```
get_hw_hbms [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
             [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア HBM コア

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device の hw_hbm オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_hbm オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

一部のザイリンクス UltraScale+デバイスには、HBM (High Bandwidth Memory) コントローラーが含まれています。このコアの詳細は、『LogiCORE IP 製品ガイド: AXI High Bandwidth Memory Controller』(PG276) を参照してください。HBM コントローラーおよびメモリ スタックにはパフォーマンス カウンターと温度センサーの両方が含まれ、ザイリンクス Vivado ハードウェア マネージャー内から HBM アクティビティ モニター (amon) を使用してアクセスできます。HBM アクティビティ モニターは、物理的なデバイス上のパフォーマンス モニターおよび温度センサーにリアルタイムにアクセスするために使用できます。Vivado ハードウェア マネージャーは、読み出し、書き込み、全体的なデータのスループット、デバイス温度を表示します。アクティビティ モニターを実行すると、データを表示、収集、CSV ファイルにエクスポートできます。

get_hw_hbms は、現在のハードウェア ターゲット上の HBM コアのリストが返されます。

引数

-of_objects <arg> (オプション): 指定したハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、またはハードウェア デバイス (hw_device) オブジェクトの HBM コントローラー (hbm) を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_hbms` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_hbm` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DISPLAY_NAME`、`HBM_STACK_NUM` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_hbm` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェアデバイスで使用可能なすべての `hw_hbm` オブジェクトのリストが取得されます。

例

次の例では、指定した DISPLAY_NAME の HBM が返されます。

```
get_hw_hbms -filter {DISPLAY_NAME == HBM_2}
```

次の例では、現在のデバイスに含まれるすべての HBM オブジェクトを返しています。

```
get_hw_hbms
```

関連項目

- [add_hw_hbm_pc](#)
- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

get_hw_ila_datas

ハードウェア ILA データ オブジェクトのリストを取得します。

構文

```
get_hw_ila_datas [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア ILA データ

使用法

名前	説明
[-of_objects]	指定した hw_ila、hw_device の hw_ila_data オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_ila_data オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

Vivado ロジック解析機能に ILA データ オブジェクトを返します。

upload_hw_ila_data コマンドでは、FPGA デバイス (hw_device) 上の ILA デバッグ コア (hw_ila) からキャプチャされたデータを移動する際に hw_ila_data オブジェクトが作成されます。また、read_hw_ila_data コマンドでディスクから ILA データ ファイルを読み込んだときにも作成されます。

hw_ila_data オブジェクトは、display_hw_ila_data コマンドを使用して Vivado ロジック解析機能の波形ウィンドウに表示でき、write_hw_ila_data コマンドを使用してディスクに保存できます。

このコマンドを実行すると、使用可能なハードウェア ILA データ オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定した `hw_ila` デバッグ オブジェクトの `hw_ila_data` オブジェクトを取得します。`hw_ila` コアは、`get_hw_ilas` または `current_hw_ila` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_ila_datas` で返されるオブジェクトのリストに、ハードウェア ILA データ オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_ila_data` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`TIMESTAMP` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (`*`) を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を `AND` (`&&`) および `OR` (`||`) で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_ila_data` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、Vivado Design Suite で現在使用可能なすべての `hw_ila_data` オブジェクトのリストが取得されます。

例

次の例では、現在のデバイス上にあるすべてのハードウェア ILA デバッグ コア のデータ オブジェクトを取得しています。

```
get_hw_ila_datas -of_objects [get_hw_ilas]
```

関連項目

- [current_hw_device](#)
- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [display_hw_ila_data](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [read_hw_ila_data](#)
- [upload_hw_ila_data](#)
- [write_hw_ila_data](#)

get_hw_ilas

ハードウェア ILA のリストを取得します。

構文

```
get_hw_ilas [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
            [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア ILA を指定します。

使用法

名前	説明
[-of_objects]	指定した hw_device の hw_ila オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_ila オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

現在のハードウェア デバイスで定義されている ILA デバッグ コア オブジェクトを返します。

ILA (Integrated Logic Analyzer) デバッグ コアを使用すると、サイリンクス FPGA にプログラムされているインプリメント済みデザイン (デザイン ビットストリーム) のインシステム デバッグを実行できます。ILA コアには、ブルートリガー論理式、エッジ遷移トリガーなど、現代のロジック アナライザーで提供される多数のアドバンス機能が含まれています。ILA コアを使用すると、デザインの特定の信号のプロープ、プログラムされたハードウェア イベントでのトリガー、サイリンクス FPGA からのリアルタイムでのデータ キャプチャを実行できます。ILA コアの詳細は、『LogiCORE IP Integrated Logic Analyzer 製品ガイド』 (PG172) を参照してください。

ILA デバッグ コアは、デザインの RTL ソース ファイルに追加するか、create_debug_core コマンドを使用して合成済みネットリストに追加できます。デザインへのデバッグ コアおよび信号プロープの追加に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。get_debug_core コマンドを使用して合成済みネットリストのデバッグ コアを取得できますが、デザインのデバッグ コアは Vivado Design Suite のハードウェア マネージャー機能で検索されるハードウェア デバッグ コアと関連はあるものの、別のものであることに注意してください。

デバッグ コアとプローブは `write_debug_probes` コマンドを使用してプローブ ファイル(.ltx) に記述され、ハードウェア デバイス (hw_device) オブジェクトの PROBES.FILE および PROGRAM.FILE プロパティを使用してビットストリーム ファイル(.bit) と共にハードウェア デバイスに関連付けます。ハードウェア デバイスにこの情報をプログラムするには、`program_hw_devices` コマンドを使用します。

ILA デバッグ コアを使用したハードウェアでのデザインのデバッグ手順は、次のとおりです。

1. `connect_hw_server` および `open_hw_target` コマンドを使用して、ハードウェア サーバーとターゲットを接続します。
2. `program_hw_devices` コマンドを使用して、FPGA をビットストリーム ファイル(.bit) およびプローブ ファイル(.ltx) でプログラムします。
3. `set_property` コマンドを使用して ILA のプロパティを設定することにより、ILA デバッグ コアのトリガーおよびキャプチャ条件を設定します。
4. `run_hw_ila` コマンドを使用して、ILA デバッグ コアをトリガー待機状態にします。
5. `display_hw_ila_data` コマンドを使用して、Vivado ロジック解析機能の波形ウィンドウで ILA デバッグ コアからキャプチャされたデータを表示します。
6. オプションで、VIO デバッグ コアを使用して制御信号を駆動し、デザインのステータス信号を確認します。詳細は、`get_hw_vios` コマンドを参照してください。
7. オプションで、JTAG-to-AXI Master デバッグ コアを使用してデザイン内のさまざまな AXI スレーブ コアにアクセスするトランザクションを実行します。詳細は、`get_hw_axis` コマンドを参照してください。

このコマンドを実行すると、デバイス上の ILA デバッグ コアのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア オブジェクトの ILA デバッグ コアを取得します。デバイスは、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_ilas` で返されたオブジェクトのリストに、ILA デバッグ コア オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ハードウェア ILA (hw_ila) オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、CONTROL.DATA_DEPTH があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「.」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): hw_ila を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェアデバイスで使用可能なすべての hw_ila のリストが取得されます。

例

次の例では、現在のハードウェアデバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_ilas -of_objects [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [create_debug_core](#)
- [current_hw_device](#)
- [current_hw_ila](#)
- [display_hw_ila_data](#)
- [get_hw_axis](#)
- [get_hw_devices](#)
- [get_hw_ila_datas](#)
- [get_hw_vios](#)
- [open_hw_target](#)
- [program_hw_devices](#)

- [run_hw_ila](#)
- [set_property](#)
- [write_debug_probes](#)

get_hw_migs

ハードウェア MIG コアのリストを取得します。

構文

```
get_hw_migs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
            [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア MIG コア

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device の hw_mig オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_mig オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

現在のハードウェア デバイスのハードウェア メモリ IP (hw_mig) オブジェクトを返します。

ザイリンクス® IP カタログに含まれるメモリ IP は、ザイリンクス デバイス用のメモリ コントローラーおよびインターフェイスを生成するために使用されます。メモリ IP には、指定されたデバイス アーキテクチャおよびメモリ インターフェイスに応じて、ザイリンクス IP カタログからの異なる IP コアが含まれます。使用可能なメモリ IP の詳細は、『Zynq-7000 SoC および 7 シリーズ デバイス メモリ インターフェイス ソリューション ユーザー ガイド』(UG586) または『LogiCORE IP UltraScale アーキテクチャ FPGA メモリ インターフェイス ソリューション製品ガイド』(PG150) を参照してください。

メモリ IP をデザインに追加したら、デザインにインプリメントする必要があります。これは、デザインの合成後、インプリメンテーションのデザイン最適化 (opt_design) 中に実行されます。または、implement_mig_cores コマンドを使用して手動で実行することもできます。

ザイリンクス IP カタログからデザインに追加する際に、メモリ コントローラーでデバッグをイネーブルにできます。Vivado ロジック解析または Vivado Lab Edition では、デザインにインプリメントされているメモリ コントローラーは、デバッグがイネーブルになっているメモリ コントローラーごとに 1 つの hw_mig オブジェクトに関連付けられます。hw_mig オブジェクトには、キャリブレーション ステータスを取得し、ビットごとのアイ マージン表示を描画するために必要なプロパティがすべて含まれます。

Vivado ロジック解析または Vivado Lab Edition では、hw_mig オブジェクトを使用して、メモリ インターフェイス デザインのキャリブレーションおよび読み出しウィンドウ マージンを検証できます。ハードウェア マネージャー GUI を使用して、キャリブレーション ステータスを確認し、クロックまたは DQS の立ち上がりエッジおよび立ち下がりエッジの読み出しマージンを検証できます。また、ILA コアを使用して、読み出しおよび書き込みのデータ インテグリティを検証できます。

このコマンドを実行すると、現在のハードウェア デバイス上の hw_mig オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、またはハードウェア デバイス (hw_device) のメモリ IP オブジェクトを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_migs` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。hw_mig オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DISPLAY_NAME`、`BYTES`、`NIBBLES` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): hw_mig オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての hw_mig オブジェクトのリストが取得されます。

例

次の例では、現在のハードウェア ターゲットで定義されているメモリ IP を取得しています。

```
get_hw_migs -of_objects [current_hw_target]
```

関連項目

- [commit_hw_mig](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [implement_mig_cores](#)
- [refresh_hw_mig](#)
- [report_hw_mig](#)
- [set_property](#)

get_hw_probes

ハードウェア プロブのリストを取得します。

構文

```
get_hw_probes [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア プロブ

使用法

名前	説明
[-of_objects]	指定した hw_interface、hw_ila、hw_vio の hw_probe オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ハードウェア プロブ オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

デザインの信号に定義されている、または指定の ILA または VIO デバッグ コアに割り当てられているハードウェア プロブ (hw_probe) オブジェクトをハードウェア マネージャーに返します。

ILA および VIO デバッグ コアは、IP カタログからコアをカスタマイズすることによりデザインの RTL ソース ファイルに追加するか、ILA デバッグ コアは create_debug_core コマンドを使用して合成済みネットリストに追加できます。

デザインの信号をプロブし、信号値をモニターして FPGA デバイス上のハードウェア イベントを監視できます。デバッグ プロブは、create_debug_port コマンドを使用して合成済みネットリスト デザインの ILA コアに追加し、connect_debug_port コマンドを使用してデザインの信号に接続します。VIO デバッグ コアにプロブを追加できるのは、IP カタログから IP コアをカスタマイズまたは再カスタマイズするときのみで、RTL デザインで信号をそのコアに接続します。デザインへの ILA および VIO デバッグ コアと信号プロブの追加に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

デバッグ コアとプローブは `write_debug_probes` コマンドを使用してプローブ ファイル(.ltx) に記述し、ハードウェア デバイス オブジェクトの `PROBES.FILE` および `PROGRAM.FILE` プロパティを使用してビットストリーム ファイルと共にハードウェア デバイスに関連付けます。ハードウェア デバイスにこの情報をプログラムするには、`program_hw_devices` コマンドを使用します。

このコマンドを実行すると、デバイス上のデバッグ プローブ オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定した ILA または VIO デバッグコアに割り当てられているデバッグ プローブを取得します。ILA または VIO コアは、`get_hw_ilas`、`current_hw_ila`、`get_hw_vios`、または `current_hw_vio` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_probes` で返されたオブジェクトのリストに、デバッグ プローブ オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_probe` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`SIGNAL` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_probe` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のデバイスに割り当てられてるすべての `hw_probe` のリストが取得されます。

例

次の例では、現在の ILA デバッグ コアに割り当てられているプローブを取得しています。

```
get_hw_probes -of_object [current_hw_ila]
```

関連項目

- [connect_debug_port](#)
- [create_debug_core](#)
- [create_debug_port](#)
- [current_hw_device](#)
- [current_hw_ila](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_vios](#)

get_hw_servers

ハードウェア サーバーのリストを取得します。

構文

```
get_hw_servers [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
               [<patterns>]
```

戻り値

ハードウェア サーバー

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_server オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明



重要: このコマンドを使用する前に、open_hw コマンドを使用して Vivado Design Suite でハードウェア マネージャーを開く必要があります。

connect_hw_server コマンドを使用して Vivado Design Suite の現在のインスタンスに接続されているハードウェア サーバー オブジェクトを返します。

ハードウェア サーバーは、リモートまたはローカル マシンで実行されているザイリンクス ハードウェア サーバー (hw_server) アプリケーションのインスタンスです。ハードウェア サーバーは、FPGA デザインをプログラムおよびデバッグするために使用する 1 つまたは複数のザイリンクス デバイスで構成された JTAG チェーンを含むハードウェア ボードハードウェア ターゲットへの接続を制御します。

このコマンドを実行すると、接続されているハードウェア サーバー オブジェクトのリストが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_servers` で返されたオブジェクトのリストに、ハードウェア サーバー オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_server` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、HOST、NAME、PORT などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_server` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在接続されている `hw_server` すべてのリストが取得されます。

例

次の例では、Vivado Design Suite に接続されているすべての hw_server が返されます。

```
get_hw_servers
```

関連項目

- [connect_hw_server](#)
- [current_hw_server](#)
- [disconnect_hw_server](#)
- [refresh_hw_server](#)

get_hw_sio_commons

ハードウェア SIO GT common のリストを取得します。

構文

```
get_hw_sio_commons [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO GT common

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgroup、hw_sio_pll の hw_sio_common オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_common オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアに定義されている QPLL オブジェクト (hw_sio_common) を返します。

各シリアル トランシーバー チャンネルには、チャンネル PLL (CPLL) というリング PLL があります。ザイリンクス UltraScale および 7 シリーズ FPGA では、高速トランシーバーの各区画にクワッド PLL (QPLL) と呼ばれる追加の共有 PLL があります。この QPLL は、高速、高パフォーマンス、低消費電力のマルチレーン アプリケーションをサポートするための共有 LC PLL です。

デバイス上では、GTXE2_CHANNEL コンポーネントにシリアル トランシーバーと CPLL ユニットが含まれ、GTXE2_COMMON に QPLL ユニットが含まれます。

このコマンドを実行すると、デバイス上の QPLL オブジェクトのリストが hw_sio_common オブジェクトのリストとして返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、またはハードウェア SIO PLL (`hw_sio_pll`) の QPLL オブジェクトを取得します。オブジェクトは、対応する `get_hw_*` コマンドを使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_commons` で返されたオブジェクトのリストに、`hw_sio_common` オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_common` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`DISPLAY_NAME` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_common` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_common` のリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_sio_commons -of [get_hw_sio_iberts]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_gtgroups

ハードウェア SIO GT グループのリストを取得します。

構文

```
get_hw_sio_gtgroups [-of_objects <args>] [-regexp] [-nocase]
                    [-filter <arg>] [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO GT グループ。

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_common、hw_sio_pll、hw_sio_gt、hw_sio_tx、hw_sio_rx の hw_sio_gtgroup オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_gtgroup オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアで使用されている区画に関連付けられている GT グループ (hw_sio_gtgroup) を返します。

GT グループはハードウェア デバイス上の I/O バンクに関連しているので、使用可能なバンクおよび GT ピンはターゲット デバイスによって決まります。たとえば、Kintex-7 xc7k325 パーツには 4 つの区画があり、それぞれに 4 つの差動 GT ピンと 2 つの差動 REFCLK ピンが含まれます。各 GT ピンにはそれ専用のトランスミッター (TX) とレシーバー (RX) があります。各区画には、共有 PLL (クワッド PLL) も 1 つ含まれます。区画は IBERT デバッグ コアで定義されており、IBERT を RTL デザインに追加するときに多数の設定を使用してカスタマイズできます。詳細は、『LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 製品ガイド』 (PG132)、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』 (UG476)、または『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』 (UG576) を参照してください。

このコマンドを実行すると、IBERT デバッグ コア上の GT グループ オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO common (`hw_sio_common`)、ハードウェア SIO PLL (`hw_sio_pll`)、ハードウェア SIO GT (`hw_sio_gt`)、ハードウェア SIO TX (`hw_sio_tx`)、またはハードウェア SIO RX (`hw_sio_rx`) オブジェクトの `hw_sio_gtgroup` オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_gtgroups` で返されたオブジェクトのリストに、GT グループ オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_gtgroup` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DISPLAY_NAME`、`GT_TYPE` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_sio_gtgroup` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_gtgroup` のリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_sio_gtgroups -of [get_hw_sio_gts *MGT_X0Y9]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_gts

ハードウェア SIO GT のリストを取得します。

構文

```
get_hw_sio_gts [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO GT

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgroup、hw_sio_pll、hw_sio_tx、hw_sio_rx、hw_sio_link の hw_sio_gt オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_gt オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアで使用されるギガビット トランシーバー (GT) オブジェクト (hw_sio_gt) を返します。

カスタマイズ可能なザイリンクス® FPGA 用の LogiCORE™ IP Integrated Bit Error Ratio Tester (IBERT) コアは、ギガビット トランシーバー (GT) を評価および監視するために設計されています。IBERT コアはインシステム シリアル I/O の検証およびデバッグをイネーブルにし、FPGA ベース システムの高速シリアル I/O リンクを計測および最適化できるようにします。詳細は、『LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 製品ガイド』(PG132)、『7 シリーズ FPGA GTX/GTH トランシーバー ユーザー ガイド』(UG476)、または『UltraScale アーキテクチャ GTH トランシーバー ユーザー ガイド』(UG576) を参照してください。

IBERT デバッグ コアを使用すると、GTX トランシーバーのダイナミック リコンフィギュレーション ポート (DRP) ポートを介して GT トランスミッターとレシーバーを設定および調整できます。これにより、GT のプロパティ 設定を変更したり、ポート上の値を制御するレジスタを変更できます。

このコマンドを実行すると、IBERT デバッグ コア上の GT オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO PLL (`hw_sio_pll`)、ハードウェア SIO TX (`hw_sio_tx`)、ハードウェア SIO RX (`hw_sio_rx`)、またはハードウェア SIO リンク (`hw_sio_link`) オブジェクトの GT オブジェクトを取得します。オブジェクトは、対応する `get_hw_*` コマンドを使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_gts` で返されたオブジェクトのリストに、GT オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_gt` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`GT_TYPE` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_sio_gt` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のデバッグ コアで使用可能なすべての `hw_sio_gt` のリストが取得されます。

例

次の例では、IBERT デバッグ コアで定義されている通信リンクに関連付けられている GT を取得しています。

```
get_hw_sio_gts -of_objects [get_hw_sio_links]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_iberts

ハードウェア SIO IBERT コアのリストを取得します。

構文

```
get_hw_sio_iberts [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO IBERT コア。

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_gtgroup、hw_sio_gt、hw_sio_common、hw_sio_pll、hw_sio_tx、hw_sio_rx、hw_sio_link の hw_sio_ibert オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_ibert オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

現在のハードウェア デバイスで定義されている IBERT (Integrated Bit Error Ratio Tester) デバッグ コア オブジェクト (hw_sio_ibert) を返します。

カスタマイズ可能なザイリンクス® FPGA 用の LogiCORE™ IP Integrated Bit Error Ratio Tester (IBERT) コアは、ギガビット トランシーバー (GT) を評価および監視するために設計されています。IBERT コアはインシステム シリアル I/O の検証およびデバッグをイネーブルにし、FPGA ベース システムの高速シリアル I/O リンクを計測および最適化できるようにします。詳細は、『LogiCORE IP Integrated Bit Error Ratio Tester (IBERT) for 7 Series GTX Transceivers v3.0 製品ガイド』(PG132) を参照してください。

IBERT デバッグ コアを使用すると、次のようなデバイスの主な GT 機能を設定および制御できます。

- TX プリエンファシスおよびポストエンファシス
- TX 差動振幅

- RX イコライゼーション
- 判定帰還等化 (DFE)
- 位相ロック ループ (PLL) の分周設定

IBERT コアは、単純なクロックや接続の問題から複雑なマージン解析およびチャネル最適化の問題まで、さまざまなインシステム デバッグおよび検証の問題を解決するために使用できます。

このコマンドを実行すると、デバイス上の IBERT デバッグ コアのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、ハードウェア デバイス (hw_device)、ハードウェア SIO GT (hw_sio_gt)、ハードウェア SIO common (hw_sio_common)、ハードウェア SIO PLL (hw_sio_pll)、ハードウェア SIO TX (hw_sio_tx)、ハードウェア SIO RX (hw_sio_rx)、またはハードウェア SIO リンク (hw_sio_link) オブジェクトの IBERT コア オブジェクトを取得します。オブジェクトは、対応する `get_hw_*` コマンドを使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_iberts` で返されたオブジェクトのリストに、IBERT デバッグ コア オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。hw_sio_ibert オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、CORE_REFRESH_RATE_MS があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_ibert` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_ibert` オブジェクトのリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_sio_iberts -of_objects [current_hw_device]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_linkgroups

ハードウェア SIO リンク グループのリストを取得します。

構文

```
get_hw_sio_linkgroups [-of_objects <args>] [-regexp] [-nocase]  
                      [-filter <arg>] [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO リンク グループ

使用法

名前	説明
[-of_objects]	指定した hw_sio_link の hw_sio_linkgroup オブジェクトを追加します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_linkgroup オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上にインプリメントされている IBERT デバッグ コアの GT 上の TX オブジェクトと RX オブジェクト間の通信リンクに関連付けられているグループを返します。

Vivado シリアル I/O 解析機能はリンク ベースです。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。リンク グループ (hw_sio_linkgroup) オブジェクトを使用すると、複数のリンクにまとめてプロパティを設定したりスキャンを実行するため、リンクをグループとして関連付けることができます。

このコマンドを実行すると、IBERT デバッグ コア上のリンク グループのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-of_objects <arg> (オプション): 指定したハードウェア SIO リンク (hw_sio_link) オブジェクトのリンク グループを取得します。リンクは、get_hw_sio_links command コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_linkgroups` で返されたオブジェクトのリストに、グループ オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_linkgroup` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`DESCRIPTION` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_sio_linkgroup` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_linkgroup` のリストが取得されます。

例

次の例では、現在のハードウェア デバイス上の IBERT デバッグ コアに定義されているすべてのリンク グループを取得しています。

```
get_hw_sio_linkgroups
```

関連項目

- [create_hw_sio_link](#)
- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_links

ハードウェア SIO リンクのリストを取得します。

構文

```
get_hw_sio_links [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO リンク

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgrou、hw_sio_gt、hw_sio_tx、hw_sio_rx、hw_sio_linkgroup の hw_sio_link オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_link オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイスで定義されている IBERT デバッグ コアの GT 上の TX オブジェクトと RX オブジェクト間の通信リンクを返します。

Vivado シリアル I/O 解析はリンク ベースの解析機能であり、IBERT デザイン内の任意のトランスミッターとレシーバーをリンクできます。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。リンクを設定するには、set_property コマンドを使用してリンク オブジェクトのプロパティ値を指定します。

このコマンドを実行すると、IBERT デバッグ コアのリンク オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO GT (`hw_sio_gt`)、ハードウェア SIO TX (`hw_sio_tx`)、ハードウェア SIO RX (`hw_sio_rx`)、またはハードウェア SIO リンク グループ (`hw_sio_linkgroup`) オブジェクトのハードウェア SIO リンク (`hw_sio_link`) オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_links` で返されたオブジェクトのリストに、リンク オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_link` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DESCRIPTION`、`TX_ENDPOINT`、`RX_ENDPOINT` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_link` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_link` オブジェクトのリストが取得されます。

例

次の例では、リンクを作成したときに指定可能な `DESCRIPTION` プロパティに基づいてリンクを取得しています。

```
get_hw_sio_links -filter {DESCRIPTION == "Link4" || DESCRIPTION == "Link2"}
```

関連項目

- [create_hw_sio_link](#)
- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_linkgroups](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_plls

ハードウェア SIO PLL のリストを取得します。

構文

```
get_hw_sio_plls [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO PLL

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgroup、hw_sio_gt、hw_sio_common の hw_sio_pll オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_pll オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアに定義されている PLL オブジェクト (hw_sio_pll) を返します。

各シリアル トランシーバー チャンネルには、チャンネル PLL (CPLL) というリング PLL があります。ザイリンクス UltraScale および 7 シリーズ FPGA では、GTX の各区画にクワッド PLL (QPLL) と呼ばれる追加の共有 PLL があります。この QPLL は、高速、高パフォーマンス、低消費電力のマルチレーン アプリケーションをサポートするための共有 LC PLL です。

デバイス上では、GTXE2_CHANNEL コンポーネントにシリアル トランシーバーと CPLL ユニットが含まれ、GTXE2_COMMON に QPLL ユニットが含まれます。

このコマンドを実行すると、デバイス上のすべての PLL オブジェクト (CPLL および QPLL の両方) のリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO GT (`hw_sio_gt`)、またはハードウェア SIO common (`hw_sio_common`) オブジェクトの PLL オブジェクトを取得します。オブジェクトは、対応する `get_hw_*` コマンドを使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_plls` で返されたオブジェクトのリストに、PLL のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_pll` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`PARENT` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_pll` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_pll` のリストが取得されます。

例

次の例では、IBERT デバッグ コア上の PLL オブジェクトのリストが返されます。

```
join [get_hw_sio_plls -of [get_hw_sio_iberts] ] \n
```

次の例では、IBERT デバッグ コアの `hw_sio_common` 上の PLL オブジェクトが返されます。

```
join [get_hw_sio_plls -of [get_hw_sio_commons ]] \n
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_rxs](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_rxs

ハードウェア SIO RX のリストを取得します。

構文

```
get_hw_sio_rxs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO RX

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgroup、hw_sio_gt、hw_sio_link の hw_sio_rx オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_rx オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアで使用するギガビット トランシーバー (GT) のレシーバー オブジェクト (hw_sio_rx) を返します。

ハードウェア デバイス上では、各 GT に PCS および PMA で構成される独立したレシーバーが含まれます。高速シリアル データは、ボード上のトレースから GTX/GTH トランシーバー RX の PMA、PCS、そして最後に FPGA ロジックに送信されます。

このコマンドを実行すると、デバイス上のレシーバー オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO GT (`hw_sio_gt`)、またはハードウェア SIO リンク (`hw_sio_link`) のレシーバー オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_rxs` で返されたオブジェクトのリストに、レシーバーのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_rx` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DISPLAY_NAME`、`RX_DATA_WIDTH` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_rx` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスにあるすべての `hw_sio_rx` のリストが取得されます。

例

次の例では、IBERT デバッグ コアで定義されている通信リンクに関連付けられているレシーバー オブジェクトを取得しています。

```
get_hw_sio_rxs -of [get_hw_sio_links]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sio_scans

ハードウェア SIO スキャンのリストを取得します。

構文

```
get_hw_sio_scans [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO スキャン

使用法

名前	説明
[-of_objects]	指定した hw_sio_rx、hw_sio_link、hw_sio_sweep の hw_sio_scan オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_scan オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

IBERT デバッグ コアのシリアル I/O 解析スキャン オブジェクトを返します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用のアイ スキャン ハードウェアを使用してリンクのスキャンを実行すると有益です。Vivado シリアル I/O 解析機能では、リンク スキャンを作成、実行、および保存できます。

このコマンドを実行すると、1 つ以上のハードウェア SIO スキャン (hw_sio_scan) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-of_objects <arg> (オプション): 指定したハードウェア SIO RX (hw_sio_rx)、ハードウェア SIO リンク (hw_sio_link)、またはハードウェア SIO スイープ (hw_sio_sweep) の hw_sio_scan オブジェクトを取得します。オブジェクトは、対応する get_hw_* コマンドを使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_scans` で返されたオブジェクトのリストに、スキャンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_scan` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`DESCRIPTION` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_sio_scan` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、IBERT デバッグ コアのすべての `hw_sio_scan` のリストが取得されます。

例

次の例では、シリアル I/O 解析スキャンを取得しています。

```
get_hw_sio_scans
```

関連項目

- [create_hw_sio_scan](#)
- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_scan](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_scan](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_scan](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_scan](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_scan](#)
- [write_hw_sio_sweep](#)

get_hw_sio_sweeps

ハードウェア SIO スイープのリストを取得します。

構文

```
get_hw_sio_sweeps [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO スイープ

使用法

名前	説明
[-of_objects]	指定した hw_sio_link、hw_sio_scan の hw_sio_sweep オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_sweep オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

IBERT デバッグ コアで定義されているシリアル I/O 解析リンク スイープ オブジェクトを返します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用の機能を使用してリンクのスキャンを実行すると有益です。また、リンクに対して GT の異なる設定を使用して複数のスキャンを実行するのも有益です。デザインにどの設定が最適かを判断するのに役立ちます。Vivado シリアル I/O 解析機能では、リンク スイープ (ある値の範囲で実行するリンク スキャンのグループ) を定義、実行、および保存できます。

このコマンドは、run_hw_sio_sweep コマンドで指定のリンクまたは GT レシーバーに対して解析を実行するのに使用可能なリンク スイープ オブジェクトを返します。write_hw_sio_sweep コマンドを使用して、スイープをディスクに保存することもできます。

作成したスイープ オブジェクトを削除するには、remove_hw_sio_sweep コマンドを使用します。

このコマンドを実行すると、1 つ以上のハードウェア SIO スイープ (hw_sio_sweep) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア SIO リンク (`hw_sio_link`) またはハードウェア SIO スキャン (`hw_sio_scan`) のシリアル I/O 解析スイープ オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_sweeps` で返されたオブジェクトのリストに、`hw_sio_sweep` オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_sweep` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`DESCRIPTION` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_sweep` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_sio_sweep` のリストが取得されます。

例

次の例では、IBERT デバッグ コアのスイープ スキャンを取得しています。

```
get_hw_sio_sweeps
```

関連項目

- [create_hw_sio_scan](#)
- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_scan](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_scan](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_scan](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_scan](#)
- [write_hw_sio_sweep](#)

get_hw_sio_txs

ハードウェア SIO TX のリストを取得します。

構文

```
get_hw_sio_txs [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア SIO TX

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device、hw_sio_ibert、hw_sio_gtgroup、hw_sio_gt、hw_sio_link の hw_sio_tx オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sio_tx オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス上の IBERT デバッグ コアで使用されるギガビット トランシーバー (GT) のトランスミッター オブジェクト (hw_sio_tx) を返します。

ハードウェア デバイス上では、各 GT に PCS および PMA で構成される独立したトランスミッターが含まれます。パラレル データは、デバイス ロジックから FPGA TX インターフェイスに送信され、PCS および PMA を介して TX ドライバーから高速シリアル データとして出力されます。

このコマンドを実行すると、デバイス上のトランスミッター オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、ハードウェア デバイス (`hw_device`)、ハードウェア SIO IBERT (`hw_sio_ibert`)、ハードウェア SIO GT (`hw_sio_gt`)、またはハードウェア SIO リンク (`hw_sio_link`) のトランスミッター オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sio_txs` で返されたオブジェクトのリストに、トランスミッターのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sio_tx` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DISPLAY_NAME`、`TXUSRCLK_FREQ` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_sio_tx` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスにあるすべての `hw_sio_tx` のリストが取得されます。

例

次の例では、現在のデザインにあるトランスミッターのリストが返されます。

```
join [get_hw_sio_txs] \n
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_rxs](#)
- [get_hw_targets](#)
- [report_property](#)

get_hw_sysmon_reg

システム モニターのレジスタ値を取得します。

構文

```
get_hw_sysmon_reg [-quiet] [-verbose] <hw_sysmon> <hexaddress>
```

戻り値

レジスタ値 (16 進数)。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sysmon></code>	hw_sysmon オブジェクトを指定します。
<code><hexaddress></code>	読み出す 16 進数アドレスを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア システム モニター (hw_sysmon) オブジェクトの指定のアドレスで定義されているシステム モニター レジスタの 16 進数値を取得します。

システム モニター (SYSMON) Analog-to-Digital Converter (ADC) は、ハードウェア デバイス (hw_device) のダイの温度および電圧を測定するために使用されます。システム モニターは、オンチップ温度および電源センサーを使用して物理環境を監視します。ADC は最大で 17 の外部アナログ入力チャンネルにアクセスできます。

システム モニターのデータは、専用レジスタであるステータス レジスタおよび制御レジスタに保存され、ハードウェア システム モニター レジスタ (hw_sysmon_reg) オブジェクトでアクセスできます。システム モニター レジスタのアドレスの詳細は、『UltraScale アークテクチャ システム モニター ユーザー ガイド』(UG580) の「SYSMON のレジスタ インターフェイス」の章または『7 シリーズ FPGA および Zynq-7000 SoC XADC デュアル 12 ビット 1MSPS アナログ - デジタル コンバーター』(UG480) を参照してください。

get_hw_sysmon_reg コマンドを使用すると、hw_sysmon オブジェクトのレジスタに保存されている値に直接アクセスできますが、レジスタの値を hw_sysmon オブジェクトのフォーマットされたプロパティとして取得することをお勧めします。たとえば、システム モニター レジスタの 16 進数にアクセスするのではなく、次のコードを使用して、システム モニターの TEMPERATURE を hw_sysmon オブジェクトのフォーマットされたプロパティとして取得します。

```
set opTemp [get_property TEMPERATURE [get_hw_sysmons]
```

get_property コマンドは、TEMPERATURE をフォーマットされた値として、hw_sysmon オブジェクトの TEMPERATURE_SCALE プロパティで指定された単位 (摂氏、華氏、またはケルビン) で返します。



ヒント: `get_property` コマンドの前に `refresh_hw_sysmon` コマンドを使用すると、オブジェクトのプロパティ値が現在のものであることが確実にになります。

`get_hw_sysmon_reg` コマンドを実行すると、指定の `hw_sysmon` の指定のアドレスにある `hw_sysmon_reg` オブジェクトのフォーマットされていない 16 進数値が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sysmon>` (必須): レジスタにアクセスする `hw_sysmon` オブジェクトを指定します。`hw_sysmon` は、`get_hw_sysmons` コマンドを使用してオブジェクトとして指定する必要があります。

`<hexaddress> arg` (必須): システム モニター上の値を返すレジスタの 16 進アドレスを指定します。アドレスは 16 進数として指定し、指定のアドレスの値は 16 進数として返されます。

例

次の例では、`hw_sysmon_reg` のアドレス 00 の値を取得しています。これは、現在のハードウェア デバイス (`hw_device`) の動作温度を保存する XADC レジスタに関連しています。

```
set opTemp [get_hw_sysmon_reg [lindex [get_hw_sysmons] 0 ] 00 ]
```

注記: 動作温度は 16 進数値として返されます。

関連項目

- [commit_hw_sysmon](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_sysmons](#)
- [open_hw_target](#)
- [refresh_hw_sysmon](#)
- [set_hw_sysmon_reg](#)
- [set_property](#)

get_hw_sysmons

ハードウェア システム モニターのリストを取得します。

構文

```
get_hw_sysmons [-of_objects <args>] [-regex] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア システム モニター

使用法

名前	説明
[-of_objects]	指定した hw_server、hw_target、hw_device の hw_sysmon' オブジェクトを取得します。
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_sysmon オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

現在のハードウェア デバイスで定義されている SYSMON デバッグ コア オブジェクトを返します。

システム モニター (SYSMON) Analog-to-Digital Converter (ADC) は、ハードウェア デバイス (hw_device) のダイの温度および電圧を測定するために使用されます。システム モニターは、オンチップ温度および電源センサーを使用して物理環境を監視します。ADC では、広範囲のアプリケーションに高精度のアナログ インターフェイスが提供されます。ADC は最大で 17 の外部アナログ入力チャネルにアクセスできます。特定のデバイス アーキテクチャの詳細は、『UltraScale アーキテクチャ システム モニター ユーザー ガイド』(UG580) または『7 シリーズ FPGA および Zynq-7000 SoC XADC デュアル 12 ビット 1MSPS アナログ - デジタル コンバーター』(UG480) を参照してください。

システム モニター (hw_sysmon) のデータは、ハードウェア システム モニター レジスタ (hw_sysmon_reg) オブジェクトを使用してアクセス可能なステータス レジスタと呼ばれる専用レジスタに保存されます。システム モニター レジスタの値は、get_hw_sysmon_reg コマンドで取得できます。

システム モニターをサポートするデバイスでは、`refresh_hw_device` コマンドを実行すると、1 つまたは複数の `hw_sysmon` オブジェクトが自動的に作成されます。`hw_sysmon` オブジェクトが作成されると、すべての温度と電圧レジスタおよび制御レジスタに対して、1 つのプロパティが割り当てられます。`hw_sysmon` オブジェクトでは、温度および電圧レジスタに割り当てられた値は既に摂氏/華氏および電圧に変換されています。

特定のレジスタの値は `get_hw_sysmon_reg` コマンドを使用して System Monitor のレジスタに格納された 16 進数値にアクセスして取得することもできますが、`hw_sysmon` オブジェクトのフォーマットされたプロパティとして表示することもできます。たとえば、レジスタの 16 進数に直接アクセスするのではなく、次のコードを使用して、指定の `hw_sysmon` オブジェクトの `TEMPERATURE` プロパティを取得できます。

```
set opTemp [get_property TEMPERATURE [get_hw_sysmons]
```

このコマンドを実行すると、現在または指定の `hw_device` 上の `hw_sysmon` オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア サーバー (`hw_server`)、ハードウェア ターゲット (`hw_target`)、または `hw_device` の `hw_sio_scan` オブジェクトを取得します。オブジェクトは、対応する `get_hw_* command` . を使用して指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_sysmons` で返されたオブジェクトのリストに、システム モニター コアのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_sysmon` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`VCCINT`、`VCCAUX` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (`*`) を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): hw_sysmon を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスにあるすべての hw_sysmon のリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている hw_sysmon オブジェクトを取得しています。

```
get_hw_sysmons -of_objects [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_sysmon_reg](#)
- [open_hw_target](#)
- [program_hw_devices](#)
- [set_property](#)

get_hw_targets

ハードウェア ターゲットのリストを取得します。

構文

```
get_hw_targets [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア ターゲット

使用法

名前	説明
[-of_objects]	指定した hw_server の hw_target オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_target オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

接続されたハードウェア サーバーで使用可能なハードウェア ターゲットを返します。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のザイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、connect_hw_server コマンドを使用して開き、ザイリンクス ハードウェア サーバー アプリケーションで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908)を参照してください。

使用可能なハードウェア ターゲットのいずれかへの接続を開くには、open_hw_target コマンドを使用します。開いたターゲットは、自動的に現在のハードウェア ターゲットとなります。現在のハードウェア ターゲットを current_hw_target コマンドを使用して定義し、そのターゲットへの接続を開くこともできます。プログラムおよびデバッグ コマンドは、ハードウェア サーバー接続を介して開いているターゲットに対して実行されます。

このコマンドを実行すると、接続されているすべてのハードウェア サーバーで使用可能なハードウェア ターゲットが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア ターゲットのハードウェア ターゲットを取得します。ハードウェア サーバー (`hw_server`) は、`get_hw_servers` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_targets` で返されたオブジェクトのリストに、ターゲットのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_target` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`IS_OPENED` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): `hw_target` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、接続されているハードウェア サーバーで使用可能なすべての `hw_target` のリストが取得されます。

例

次の例では、現在接続されているハードウェア サーバーで使用可能なハードウェア ターゲットが返されます。

```
get_hw_targets
```

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [get_hw_servers](#)
- [open_hw_target](#)

get_hw_vios

ハードウェア VIO のリストを取得します。

構文

```
get_hw_vios [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
            [-quiet] [-verbose] [<patterns>]
```

戻り値

ハードウェア VIO

使用法

名前	説明
[-of_objects]	指定した hw_device の hw_vio オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	hw_vio オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア デバイス (hw_device) で定義されている VIO (Virtual Input/Output) デバッグ コア オブジェクトを返します。

VIO (Virtual Input/Output) デバッグ コアは、サイリンクス FPGA にプログラムされている内部信号をリアルタイムで監視および駆動できます。ターゲット ハードウェアへの物理的なアクセスがない場合は、このデバッグ機能を使用して、物理デバイス上の信号を駆動および監視できます。

VIO コアは、ハードウェア プロブ (hw_probe) オブジェクトを使用してデザインの特定の信号を監視および駆動します。入力プロブは、VIO コアへの入力として信号を監視します。出力プロブは、VIO コアから信号を指定の値に駆動します。デバッグ コアの値は、commit_hw_vio コマンドを使用してプロブで信号に駆動します。

VIO デバッグ コアは、RTL コードにインスタンスエートする必要があるため、デザインをデバッグする前にどのネットを監視および駆動するのか決めておく必要があります。VIO コアは、IP カタログの [Debug & Verification] → [Debug] カテゴリに含まれます。VIO コア IP の詳細は、『LogiCORE IP Virtual Input/Output 製品ガイド』(PG159) を参照してください。

このコマンドを実行すると、デバイス上の VIO デバッグ コアのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (オプション): 指定したハードウェア オブジェクトの VIO デバッグ コアを取得します。デバイスは、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_hw_vios` で返されたオブジェクトのリストに、VIO デバッグ コアのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`hw_vio` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`INSTANCE_NAME` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `hw_vio` オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のハードウェア デバイスで使用可能なすべての `hw_vio` オブジェクトのリストが取得されます。

例

次の例では、現在のハードウェア デバイスで定義されている ILA デバッグ コアを取得しています。

```
get_hw_vios -of_objects [current_hw_device]
```

関連項目

- [commit_hw_vio](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_probes](#)
- [program_hw_devices](#)
- [refresh_hw_vio](#)
- [reset_hw_vio_activity](#)
- [reset_hw_vio_outputs](#)
- [set_property](#)

get_interfaces

現在のデザインの I/O ポート インターフェイスのリストを取得します。

構文

```
get_interfaces [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
               [-quiet] [-verbose] [<patterns>]
```

戻り値

インターフェイス オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピンまたはネットのインターフェイスを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	I/O ポート インターフェイスを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

現在のプロジェクトに含まれる I/O インターフェイスで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトに含まれる I/O インターフェイスすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_interfaces` で返されたオブジェクトのリストに、インターフェイスのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (`*`) を使用すると、定義値が `""` のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): インターフェイスが割り当てられているピンまたはネットを 1 つ以上指定します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>`: インターフェイスを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (`*`) で、プロジェクトに含まれるすべてのインターフェイスのリストが取得されます。

例

次の例では、プロジェクトに含まれるインターフェイスすべてのリストを取得しています。

```
get_interfaces
```

関連項目

- [create_interface](#)
- [delete_interface](#)

get_io_standards

I/O 規格のリストを表示します。

構文

```
get_io_standards [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

I/O 規格

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定した BEL、サイト、package_pin、io_bank、ポートの I/O 規格を取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	I/O 規格を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

ターゲット デバイスで使用可能な I/O 規格 (IOSTANDARD) のリストを取得します。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_io_standards` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IOSTANDARD オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、IS_DCI、IS_DIFFERENTIAL などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND 「`&&`」および OR 「`||`」で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したパッケージ ピン、サイト、BEL、I/O バンク、またはポート オブジェクトの IOSTANDARD を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): IOSTANDARD を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、ターゲット パーツで使用可能なすべての IOSTANDARD が取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて IOSTANDARD を検索できます。

注記: 複数の検索パターンは「`{ }`」で囲み、1 つのエレメントとして指定します。

例

次の例では、ターゲット デバイスで使用可能な差動 IOSTANDARD のリストを取得しています。

```
get_io_standards -filter {IS_DIFFERENTIAL}
```

注記: 検索パターンに一致するオブジェクトがない場合は、警告メッセージが表示されます。

関連項目

- [list_property](#)
- [report_property](#)

get_iobanks

I/O バンク (iobank) のリストを取得します。

構文

```
get_iobanks [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]  
            [-quiet] [-verbose] [<patterns>]
```

戻り値

I/O バンクのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した package_pin、ポート、クロック領域、SLR、またはサイトの iobank を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	I/O バンクを検索するパターンを指定します。デフォルトは * です。

カテゴリ

XDC、Object (オブジェクト)

説明

現在のプロジェクトのターゲット デバイスの I/O バンクで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、ターゲット デバイスの I/O バンクすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_iobanks` で返されたオブジェクトのリストに、I/O バンクのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。I/O バンク オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`BANK_TYPE`、`DCI_CASCADE`、`INTERNAL_VREF` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したオブジェクトに接続されている I/O バンクのリストを取得します。指定可能なオブジェクト タイプは、クロック領域 (`clock_region`)、パッケージ ピン (`package_pin`)、ポート、SLR、およびサイトです。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン「`<pattern>`」と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): I/O バンクを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、デザインの I/O バンクすべてのリストが取得されます。

例

次の例では、指定したパッケージ ピンの I/O バンクが返されます。

```
get_iobanks -of_objects [get_package_pins H4]
```

次の例では、デバイスにある GT バンクが返されます。

```
get_iobanks -filter {BANK_TYPE == BT_MGT}
```

関連項目

- [get_package_pins](#)
- [get_ports](#)
- [get_sites](#)
- [list_property](#)
- [place_ports](#)
- [report_property](#)

get_ip_upgrade_results

現在のプロジェクトでの IP アップグレードの結果のリストを取得します。

構文

```
get_ip_upgrade_results [-srcset <arg>] [-quiet] [-verbose] [<objects>...]
```

戻り値

IP アップグレード結果のリスト

使用法

名前	説明
[-srcset]	アップグレードされた IP を含むソース ファイルセットを指定します。デフォルトは、現在のソース ファイルセットです。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	アップグレードする IP を指定します。デザイン内の IP インスタンスを <code>get_ips <instance name></code> または <code>get_bd_cells <cell name></code> コマンドを使用して指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

指定の IP の upgrade_log ファイルの名前を取得します。

このコマンドは、Vivado IDE で [IP Status] ウィンドウに upgrade_log ファイルからの情報を表示するために使用されます。このコマンドを使用すると、upgrade_log ファイルの名前をすばやく取得し、適切なファイル コマンドを使用してその内容を読み出しまたは表示できます。

このコマンドを実行すると、指定した IP オブジェクトの upgrade_log ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-srcset <arg> (オプション): 指定の IP ファイルを調べるソース ファイルセットを指定します。このオプションを使用すると、デフォルトの sources_1 ファイルセットとは異なるファイルセットに変更できます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): アップブレード結果をレポートする IP オブジェクトを指定します。IP オブジェクトは `get_ips` コマンドを使用して指定します。

例

次の例では、現在のファイルセットに含まれるすべての IP の `upgrade_log` ファイルの名前を取得しています。

```
get_ip_upgrade_results [get_ips]
```

次の例では、アップグレードされた IP のアップグレード ログを Tcl コンソールに表示しています。

```
set ipDefs [get_ip_upgrade_results [get_ips]]
for {set x 0} {$x<[llength $ipDefs]} {incr x} {
    set ipRoot [file rootname [lindex $ipDefs $x]]
    puts "Upgrade Log for $ipRoot"
    set ipDir [get_property IP_DIR [get_ips $ipRoot]]
    set ipLog [lindex $ipDefs $x]
    set catLog [concat $ipDir/$ipLog]

    # Check for file existence, and read file contents
    if {[file isfile $catLog]} {
        # Open the file for read, save the File Handle
        set FH [open $catLog r]
        #puts "Open $FH"
        set content [read $FH]
        foreach line [split $content \n] {
            # The current line is saved inside $line variable
            puts $line
        }
        close $FH
        #puts "Close $FH"
    }
}
```

関連項目

- [current_fileset](#)
- [get_ips](#)
- [import_ip](#)
- [report_ip_status](#)
- [upgrade_ip](#)

get_ipdefs

現在の IP カタログから IP のリストを取得します。

構文

```
get_ipdefs [-name] [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
           [-all] [-quiet] [-verbose] [<patterns>...]
```

戻り値

カタログ IP オブジェクトのリスト

使用法

名前	説明
[-name]	検索パターンを VLNV ではなく IP 表示名と比較します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した IP インスタンスまたは XCI ファイルの IP 定義を取得します。
[-all]	隠し IP を返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

現在のプロジェクトの IP カタログで定義されている IP コアで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、Vivado ツールの IP カタログで定義された IP コアすべてが返されます。

デフォルトでは、カタログに含まれるすべての IP コアが VLNV プロパティに基づいて検索されます。-name オプションを使用すると、IP コアの表示名で検索できます。

引数

-name (オプション): 検索パターン (<pattern>) を IP の VLNV プロパティではなく DISPLAY_NAME プロパティと比較して検索します。



ヒント: 検索可能なのは 1 つの文字列のみなので、Multiply Adder のように表示名が複数の単語で構成されている場合は、「*Multiply*」または「*Adder*」を使用して IP コアを検索する必要があります。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>`: 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_ipdefs` で返されたオブジェクトのリストに、オブジェクトのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`ipdefs` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、VLNV、NAME、IS_AXI などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定した IP インスタンスまたは IP ファイル (.xci) オブジェクトの IP 定義を取得します。オブジェクトは、`get_ips` または `get_files` コマンドを使用して指定する必要があります。

`-all` (オプション): 定義されたすべての IP カタログおよびリポジトリから IP 定義を取得します。デフォルトでは、`get_ipdefs` コマンドでスタンドアロンの IP カタログからの IP コアが返されます。このオプションを使用すると、Vivado ツールの標準 IP カタログと IP インテグレーターの IP カタログの両方からの IP が返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): IP カタログで IP コアを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、カタログに含まれるすべての IP コアのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてコア定義を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエLEMENTとして指定します。

例

次の例では、NAME プロパティが指定の検索パターンに一致する IP コアのリストが返されます。

```
get_ipdefs -filter {NAME=~*agilent*}
```



ヒント: フィルター演算子「=~」により、指定の検索パターンにおおよそ一致する IP コアが検索されます。

次の例では、AXI 準拠の IP コアのリストが返されます。

```
get_ipdefs -filter {IS_AXI==1}
```

次の例では、Vivado ツールの標準 IP カタログと IP インテグレーターの IP カタログからの IP が返されます。

```
get_ipdefs -all *axi_interconnect*
```

次の例では、上記の結果を DESIGN_TOOL_CONTEXTS プロパティでフィルターし、IP インテグレーター IP のみを取得しています。

```
get_ipdefs -all *axi_interconnect* -filter {DESIGN_TOOL_CONTEXTS =~*IPI*}
```



ヒント: DESIGN_TOOL_CONTEXTS プロパティでフィルターすると、Vivado ツールの標準 IP カタログではなく IP インテグレーターからの IP を特定できます。

IP の複数のバージョンが返される場合、UPGRADE_VERSIONS プロパティを使用すると、次の例に示すように特定のバージョンまたは最新バージョンを取得できます。

```
get_ipdefs -all *axi_interconnect* -filter {UPGRADE_VERSIONS == ""}
```



ヒント: {UPGRADE_VERSIONS == ""} フィルターにより、アップグレードのない (最新バージョンの) IP が返されます。

関連項目

- [create_ip](#)
- [generate_target](#)
- [get_ips](#)
- [import_ip](#)
- [report_property](#)
- [update_ip_catalog](#)

get_ips

現在のデザインの IP のリストを取得します。

構文

```
get_ips [-regex] [-nocase] [-all] [-filter <arg>] [-exclude_bd_ips]
        [-of_objects <args>] [-quiet] [-verbose] [<patterns>...]
```

戻り値

IP オブジェクトのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。
[-all]	検索にサブコア IP を含めます。
[-filter]	式を使用してリストをフィルター処理します。
[-exclude_bd_ips]	ブロック デザインに含まれる IP を除外します。
[-of_objects]	指定した IP、ファイルの IP オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	IP 名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regex を指定している場合は「.*」です。

カテゴリ

[Object \(オブジェクト\)](#)、[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

現在のプロジェクトに含まれる IP コアで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトの IP すべてのリストが返されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-all` (オプション): IP オブジェクトのサブモジュールを返します。サブモジュールは、親 IP コア生成の一部として作成される IP インスタンスです。デフォルトでは、現在のプロジェクトまたはデザインで使用される親 IP オブジェクトのみが返され、それらの親 IP オブジェクト内で使用されている IP コアは返されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_ips` で返されたオブジェクトのリストに、IP コアのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。IP オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、DELIVERED_TARGETS などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-exclude_bd_ips` (オプション): 現在のデザインでブロック デザインに含まれる IP を除外します。

`-of_objects <arg>` (オプション): 指定した IP またはファイル オブジェクトの IP コアを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): デザイン内の IP コアを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべての IP コアのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてコアを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、名前が「EDK」という文字列で始まる IP コアのリストが返されます。

```
get_ips EDK*
```

関連項目

- [create_ip](#)
- [generate_target](#)
- [get_ipdefs](#)
- [import_ip](#)
- [update_ip_catalog](#)

get_lib_cells

ライブラリ セルのリストを取得します。

構文

```
get_lib_cells [-regexp] [-filter <arg>] [-nocase] [-include_unsupported]
               [-of_objects <args>] [-quiet] [-verbose] <patterns>
```

戻り値

ライブラリ セルのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp を使用した場合にのみ適用されます。
[-include_unsupported]	テストのみのライブラリ セルを含めます。
[-of_objects]	指定したオブジェクトのライブラリ セルを取得します。指定可能なオブジェクトは、セルまたはインスタンス (get_cells)、セルピン (get_pins)、およびライブラリ ピン (get_lib_pins) です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<patterns>	ライブラリ セル名を検索するパターンを指定します。

カテゴリ

Object (オブジェクト)

説明

現在のデザインのターゲット パーツのライブラリに含まれるセルのリストを取得します。このコマンドを使用すると、特定のライブラリ セルやセル タイプを検索して、そのセルのプロパティを取得できます。

このコマンドを実行するには、ライブラリ名とセル名 (lib_name/cell_name) を含む階層名が必要です。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_lib_cells` で返されたオブジェクトのリストに、セルのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定のセル、ピン、またはライブラリ ピンのライブラリ セルのリストを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (必須): ライブラリ セルを検索するパターンを指定します。検索パターンには、ライブラリ名とセル名の両方を指定する必要があります。

例

次の例では、現在のデザインのターゲット パーツのライブラリのセル数を取得し、そのライブラリの AND タイプのセル数を取得しています。

```
llength [get_lib_cells [get_libs]/*]  
795  
llength [get_lib_cells [get_libs]/AND*]  
18
```

次の例では、指定したセル オブジェクトのライブラリ セルを取得しています。

```
get_lib_cells -of_objects [lindex [get_cells] 1]
```

関連項目

- [get_cells](#)
- [get_libs](#)
- [get_lib_pins](#)
- [list_property](#)
- [report_property](#)

get_lib_pins

ライブラリ セル ピンのリストを取得します。

構文

```
get_lib_pins [-regexp] [-filter <arg>] [-nocase] [-of_objects <args>]  
             [-quiet] [-verbose] <patterns>
```

戻り値

ライブラリ セル ピンのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します。このオプションは、 <code>-regexp</code> を使用した場合にのみ適用されます。
<code>[-of_objects]</code>	指定したオブジェクトのライブラリ セル ピンを取得します。指定可能なオブジェクトは、セルまたはインスタンス (<code>get_cells</code>)、セル ピン (<code>get_pins</code>)、およびライブラリ セル (<code>get_lib_cells</code>) です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><patterns></code>	ライブラリ セル ピン名を検索するパターンを、 <code><ライブラリ セル パターン>/<ライブラリ セル ピン パターン></code> の形式で指定します。

カテゴリ

[Object \(オブジェクト\)](#)

説明

現在のデザインのターゲット パーツのセル ライブラリから、指定したセルのピンのリストを取得します。

注記: このコマンドを実行するには、ライブラリ名、セル名、ピンを含む階層名が必要です (`lib_name/cell_name/pins`)。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` <args> (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_lib_pins` で返されたオブジェクトのリストに、ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <, >, <=, および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects` <arg> (オプション): 指定したセル、ライブラリ セル、またはピン オブジェクトのライブラリ セル ピンのリストを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (必須): ピンを検索するパターンを指定します。検索パターンには、ライブラリ名、セル名、およびピンを指定する必要があります。

例

次の例では、ライブラリ セル ピンすべてのリストを取得しています。

```
get_lib_pins xt_virtex6/AND2/*
```

次の例では、ターゲット デバイスのセル ライブラリに含まれるすべてのセルの、ピンすべてのリストを取得しています。

```
get_lib_pins [get_libs]/*/*
```

関連項目

- [get_libs](#)
- [get_lib_cells](#)
- [list_property](#)
- [report_property](#)

get_libs

ライブラリのリストを取得します。

構文

```
get_libs [-regexp] [-filter <arg>] [-nocase] [-quiet] [-verbose]
[<patterns>]
```

戻り値

ライブラリのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp を使用した場合にのみ適用されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ライブラリ名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

現在のデザインのターゲット デバイスのセル ライブラリを取得します。デバイス ファミリによって使用できるプリミティブは異なるので、デバイス ファミリごとに 1 つのライブラリがあります。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_libs` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ライブラリを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべてのライブラリのリストが取得されます。

例

次の例では、ターゲット パーツのセル ライブラリを取得しています。

```
get_libs
```

関連項目

- [get_lib_cells](#)
- [get_lib_pins](#)
- [list_property](#)
- [report_property](#)

get_macros

現在のデザインに含まれるマクロのリストを取得します。

構文

```
get_macros [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
           [-quiet] [-verbose] [<patterns>]
```

戻り値

マクロ オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセルのマクロを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	マクロ名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

XDC、Object (オブジェクト)

説明

現在のデザインに含まれるマクロで、検索パターンに一致するもののリストを取得します。デフォルトでは、デザインのすべてのマクロのリストが返されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_macros` で返されたオブジェクトのリストに、マクロのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。マクロ オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、ABSOLUTE_GRID、RLOCS などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したセル オブジェクトのマクロを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): マクロを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、プロジェクトに含まれるすべてのマクロのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてマクロを検索できます。

注記: 複数の検索パターンは波かっこ「`{ }`」またはダブルクォーテーション「`''`」で囲み、1 つのエレメントとして指定します。

例

次の例では、指定の検索パターンに一致するマクロに現在割り当てられているプロパティが返されます。

```
report_property [get_macro *Macro1]
```

注記: 検索パターンに一致するマクロがない場合は、警告メッセージが表示されます。

関連項目

- [create_macro](#)
- [list_property](#)
- [report_property](#)
- [update_macro](#)

get_marked_objects

マークされたオブジェクトを取得します。

構文

```
get_marked_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose]
```

戻り値

マークされたオブジェクトのリスト

使用法

名前	説明
[-rgb]	色を RGB で指定します。
[-color]	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[GUIControl \(GUI 制御\)](#)

説明

Vivado IDE で開いている現在のデザインでマークされているオブジェクトを取得します。オブジェクトをマークするには、`mark_objects` コマンドを使用します。

デザインでマークされているオブジェクトをすべて取得するか、指定した名前または RGB 値の色でマークされているオブジェクトを取得できます。

注記: この Tcl コマンドは、Vivado を GUI モードで実行している場合にのみ機能します。

引数

`-rgb <args>` (オプション): オブジェクトをマークする色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定し、{0 255 0} は緑を指定します。

`-color <arg>` (オプション): オブジェクトをマークする色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインでマークされているオブジェクトすべてを取得しています。

```
get_marked_objects
```

次の例では、現在のデザインに含まれるオブジェクトで、指定の色でマークされているものを取得しています。

```
get_marked_objects -color yellow
```

関連項目

- [get_highlighted_objects](#)
- [highlight_objects](#)
- [mark_objects](#)
- [select_objects](#)

get_methodology_checks

設計手法ルール チェック オブジェクトのリストを取得します。

構文

```
get_methodology_checks [-regexp] [-nocase] [-filter <arg>] [-abbrev <arg>]
                        [-quiet] [-verbose] [<patterns>]
```

戻り値

設計手法ルール チェック オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-abbrev]</code>	指定した略称のルールで ID が最大のものを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	<code>rule_check</code> オブジェクトを検索するパターンを指定します。デフォルトは <code>*</code> です。

カテゴリ

Methodology (設計手法)、Object (オブジェクト)

説明

現在定義済みの設計手法チェックのリストを取得します。このリストには、デフォルトで定義されているプロセスおよびタイミングの設計手法チェックも含まれます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_methodology_checks` で返される設計手法チェックのリストに、プロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-abbrev <arg>` (オプション): 指定した名前または略称の設計手法チェックを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 設計手法チェックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、すべての設計手法チェックが取得されます。

例

次の例では、すべての合成設計手法チェックのリストを取得しています。

```
get_methodology_checks SYNTH*
```

関連項目

- [get_methodology_violations](#)

- [list_property](#)
- [report_methodology](#)
- [report_property](#)

get_methodology_violations

前回の report_methodology コマンド実行からの設計手法違反のリストを取得します。

構文

```
get_methodology_violations [-name <arg>] [-regexp] [-filter <arg>]  
[-nocase] [-quiet] [-verbose] [<patterns>]
```

戻り値

設計手法違反オブジェクトのリスト

使用法

名前	説明
[-name]	指定した名前の結果を取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	設計手法違反を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

Methodology (設計手法)、Object (オブジェクト)

説明

report_methodology コマンドを実行したときにデザインで検出された違反オブジェクトのリストを取得します。個々の違反オブジェクトの詳細は、report_property または list_property コマンドを使用して取得します。

違反オブジェクトは、現在のデザインのセル、ネット、ピン、ポート、または現在のデバイス上のサイトに関連付けられています。設計手法違反オブジェクトに関連付けられているデザイン オブジェクトは、該当する get_* コマンド (get_cells、get_nets など) で -of_objects オプションを使用して取得できます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-name <arg> (オプション): 指定した名前の設計手法結果セットの違反を取得します。このオプションを使用する場合、report_methodology コマンドを -name オプションを使用して実行する必要があります。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_methodology_violations` で返されたオブジェクトのリストに、違反のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 違反を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべての違反が取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて違反を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、現在のデザインの設計手法違反をレポートし、これらの違反のリストを取得しています。

```
report_methodology  
get_methodology_violations
```

次の例では、指定した名前の設計手法レポートの違反をリストし、違反に関連付けられているピンを取得しています。

```
report_methodology -name method_1  
get_pins -of_objects [get_methodology_violations -name method_1]
```

関連項目

- [get_methodology_checks](#)
- [list_property](#)
- [report_methodology](#)
- [report_property](#)

get_msg_config

現在のメッセージ カウント、最大表示回数、または set_msg_config コマンドで定義されたメッセージの制御ルールを返します。

構文

```
get_msg_config [-id <arg>] [-severity <arg>] [-rules] [-limit] [-count]
               [-quiet] [-verbose]
```

使用法

名前	説明
[-id]	メッセージ ID を指定します。-limit または -count と共に使用する必要があります。デフォルトは空です。
[-severity]	メッセージの重要度を指定します。-limit または -count と共に使用する必要があります。デフォルトは空です。
[-rules]	現在のプロジェクトに設定されているメッセージに制御ルールを表形式で表示します。
[-limit]	-id または -severity に一致するメッセージの最大表示回数を表示します。
[-count]	-id または -severity に一致するメッセージのこれまでに生成された回数を表示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

指定のメッセージ ID または重要度の現在のメッセージ カウントまたは最大表示回数、あるいは現在のプロジェクトで定義されているすべてのメッセージの制御ルールを返します。メッセージの制御ルールは、set_msg_config コマンドで定義します。

-count オプションを使用すると、指定の ID または重要度のメッセージがこれまでに生成された回数が表示されます。



重要: get_msg_config コマンドは、Vivado を起動した元の CPU プロセスのメッセージ数をレポートします。launch_runs コマンドで合成 run およびインプリメンテーション run を実行するために使用されるサブプロセスなど、Vivado Design Suite で実行されるサブプロセスは、このメッセージ数には含まれません。これが、get_msg_config -count コマンドで返されるメッセージ数と Vivado IDE に表示されるものまたは予測されるものと異なる場合に、混乱の元となることがあります。そのため、-count オプションは非プロジェクトベースのデザインで使用するのが適しています。

-limit オプションを使用すると、指定の ID または重要度のメッセージに現在設定されている最大表示回数が表示されます。

`-rules` オプションを使用すると、現在定義されているすべてのメッセージの制御ルールが表形式で表示されます。

注記: 1 つの `get_msg_config` コマンドでは、最大表示回数、現在のカウント、ルールのいずれかのみを返すことができます。複数指定すると、エラーが返されます。

引数

`-id <arg>` (オプション): メッセージ ID を指定します。各メッセージには、独自の ID が含まれます。たとえば、「Common 17-54」、「Netlist 29-28」です。

`-severity <value>` (オプション): メッセージの重要度を指定します。次の 5 つの重要度があります。

- **ERROR:** デザインの結果が使用不可となったり、ユーザーの操作がないと回避できないような問題が発生していることを示すエラー メッセージです。
- **{CRITICAL WARNING}:** 入力や制約に適用されないものがあったり、FPGA ファミリには適していないものがあることを示すクリティカル警告メッセージです。修正することが強く推奨されます。

注記: これは 2 単語の値なので、`{ }` で囲む必要があります。

- **WARNING:** 制約または仕様が指定どおりに適用されず、デザインが最適な結果にならない可能性を示す警告メッセージです。修正するかどうかは、ユーザーが判断します。
- **INFO:** STATUS メッセージと同じですが、重要度とメッセージ ID タグが含まれる点が異なります。INFO メッセージには、必要に応じてアンサー データベースで確認できるようにメッセージ ID が含まれます。
- **STATUS:** デザインの処理に関するプロセスおよびフィードバックのステータスを示すステータス メッセージです。STATUS メッセージには、メッセージ ID は含まれません。

`-rules` (オプション): 現在のプロジェクトで `set_msg_config` コマンドを使用して定義されているメッセージの制御ルールを返します。

注記: `-rule` を指定すると、`-id` または `-severity` の指定にかかわらず、現在のプロジェクトのルールがすべて返されます。

`-limit` (オプション): 指定したメッセージ ID または重要度のメッセージの現在の最大表示回数を返します。

`-count` (オプション): 指定したメッセージ ID または重要度のメッセージの現在のカウントを返します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定した情報メッセージの現在のカウントを取得しています。

```
get_msg_config -id "Common 17-81" -count
```

次の例では、現在のプロジェクトで定義されているメッセージの制御ルールが返されます。

```
get_msg_config -rules
```

次の例では、指定したメッセージ ID のメッセージの重要度を変更し、現在のメッセージ制御ルールを取得して、特定のルールをリセットする 2 つのコマンドを示しています。

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
-----
Message control rules currently in effect are:
Rule Name   Rule                                           Current
Message Count
1  set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
-----
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```



ヒント: 上記の例でメッセージをリセットするのに必要なのは、いずれか 1 つの `reset_msg_config` コマンドのみです。

関連項目

- [reset_msg_config](#)
- [set_msg_config](#)

get_net_delays

ドライバーから各ロード ピンまでのネットの配線遅延または見積もり遅延 (ps) を取得します。

構文

```
get_net_delays -of_objects <args> [-regexp] [-nocase] [-patterns <arg>]
               [-filter <arg>] [-to <args>] [-interconnect_only] [-quiet] [-verbose]
```

戻り値

ネット遅延

使用法

名前	説明
-of_objects	指定した net の net_delay オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-patterns]	net_delay オブジェクトを検索するパターンを指定します。デフォルトは * です。
[-filter]	式を使用してリストをフィルター処理します。
[-to]	指定したターミナル、またはポートまでのネットの遅延を取得します。
[-interconnect_only]	インターコネクト遅延のみを含めます。デフォルトでは、サイト間遅延が含まれます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)、Netlist (ネットリスト)、Object (オブジェクト)

説明

現在のデザインで指定した、ドライバーから各ロード ピン、または指定のピンを介する指定のロード ピンまでのネットの遅延オブジェクトを取得します。

遅延オブジェクトには、ファーストおよびスロー プロセス コーナーの最大および最小遅延を定義するプロパティが含まれます。get_property コマンドを使用すると、遅延オブジェクトから必要なプロパティを取得できます。遅延オブジェクトの遅延プロパティ値は、ピコ秒で指定します。



ヒント: Vivado ツールでは、ほとんどの場合、遅延値はナノ秒で指定されますが、遅延オブジェクトではピコ秒が使用されます。

ネットが配線されているかどうかによって、算出された値または見積もり値が返されます。遅延値は、配線済みの実際の遅延、未配線のネットの見積もりネット遅延になります。ネット遅延には、ロジック エlementまたはデバイスサイトを介する遅延を含めるか、またはインターコネクト遅延のみを含めることができます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

`get_net_delays` コマンドを実行すると、指定のネットの遅延オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-of_objects <arg>` (必須): 指定したネット オブジェクトのネット遅延を取得します。このオプションは、`get_net_delays` コマンドで返されるデータ量を削減するために使用できます。

注記: `-of_objects` オプションは、`get_*` コマンド (`get_net_delays` では `get_nets`) を使用してオブジェクトとして指定する必要があります。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-patterns <arg>` (オプション): ネット遅延を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべてのネット遅延のリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1つのElementとして指定します。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_net_delays` で返されたオブジェクトのリストに、遅延オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。遅延オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NET`、`FAST_MAX`、`FAST_MIN` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-to <args> (オプション): 遅延を算出するネットの終点を指定します。ピン、ポート、または PIP オブジェクトを終点として指定できます。

-interconnect_only (オプション): 配線によるネットの遅延の算出にインターコネクト遅延のみを含めます。デフォルトでは、遅延の算出にサイト間遅延も含まれます。



ヒント: route_design コマンドの -min_delay または -max_delay オプションの遅延範囲を get_net_delays コマンドを使用して定義する場合、-interconnect_only オプションを使用して指定したネット遅延にインターコネクトのみが含まれるようにしてください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、指定のネットのインターコネクト遅延のみを遅延オブジェクトとして取得しています。

```
report_property -all [lindex [get_net_delays -interconnect_only \
-of_objects [get_nets control_reg[*]]] 16 ]
```



ヒント: 遅延オブジェクトの FAST_MAX、FAST_MIN、SLOW_MAX、および SLOW_MIN プロパティ値は、ピコ秒でレポートされます。

関連項目

- [get_pins](#)
- [get_pips](#)
- [get_ports](#)
- [get_property](#)
- [list_property](#)
- [report_property](#)

get_nets

現在のデザインのネットのリストを取得します。

構文

```
get_nets [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-filter <arg>]
         [-of_objects <args>] [-match_style <arg>]
         [-top_net_of_hierarchical_group] [-segments] [-boundary_type <arg>]
         [-quiet] [-verbose] [<patterns>]
```

戻り値

ネット オブジェクトのリスト

使用法

名前	説明
[-hsc]	階層区切り文字を指定します。デフォルトはスラッシュ (/) です。
[-hierarchical]	すべての階層レベルで検索します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したピン、ポート、セル、タイミング パス、クロック、または DRC 違反のネットを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-top_net_of_hierarchical_group]	階層ネットの上位に属するネット セグメントを返します。
[-segments]	階層全体のネットのセグメントをすべて返します。
[-boundary_type]	ピンと同じレベル (upper)、下のレベル (lower)、またはその両方 (both) にある階層ピンに接続されているネット セグメントを返します。有効な値は upper、lower、both で、デフォルトは upper です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ネット名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

SDC、XDC、Object (オブジェクト)

説明

現在のデザインに含まれるネットで、検索パターンに一致するもののリストを取得します。デフォルトでは、開いているデザインの `current_instance` コマンドで指定されている現在のインスタンスに含まれるすべてのネットのリストが返されます。

`-hierarchical` オプションを使用すると、現在のデザインのすべての階層からネットを取得できます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-hsc <arg>` (オプション): デフォルトの階層区切り文字はスラッシュ (/) です。それ以外の階層区切り文字を指定する場合は、このオプションを使用します。

`-hierarchical` (オプション): 現在のインスタンスから開始し、デザイン階層のすべてのレベルからネットを取得します。このオプションを指定しない場合、`current_instance` コマンドで設定された現在のインスタンスのネットのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層ネット名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「U1/*」を指定すると、名前に「U1/」を含むネットが階層の各レベルで検索され、意図した結果が得られない可能性があります。`-hierarchical` を使用した検索の例は、「`get_cells`」を参照してください。

注記: `-regex` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「U1/*」を指定した場合に意図した結果が得られます。

`-regex` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regex` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regex.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regex` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_nets` で返されたオブジェクトのリストに、ネットのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ネット オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`PARENT`、`TYPE`、`MARK_DEBUG` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価(==)、不等価(!=)、一致(=~)、不一致(!~)です。数値比較演算子<、>、<=、および>=も使用できます。複数のフィルター式をAND(&&)およびOR(||)で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型(boolean)プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects <arg> (オプション): 指定したセル、ピン、ポート、またはクロックに接続されているネット、または指定のDRC違反オブジェクトに関連付けられているネットのリストを取得します。

注記: -of_objects オプションでは、オブジェクトを名前で指定するのではなく、get_* コマンド(get_cells、get_pins など)を使用して指定する必要があります。また、-of_objects を検索パターン(<pattern>)と共に使用することはできません。

-match_style [sdc | ucf] (オプション): 検索パターンがUCF制約またはSDC制約に一致することを示します。デフォルトはSDCです。

-top_net_of_hierarchical_group (オプション): 階層ネットの最上位ネットセグメントまたはすべてのネットの最上位ネットセグメントを取得します。このオプションを-segmentと共に使用すると、階層ネットのすべてのセグメントから最上位ネットセグメントが返されます。

-segments (オプション): 階層ネットのすべての階層レベルにあるすべてのセグメントを取得します。このオプションは、指定のネットだけではなく、指定のネットのすべてのセグメントを返す点が-hierarchicalオプションとは異なります。



重要: -segments オプションは、-filter オプションのフィルターパターンに一致しないネットが除去された後に適用されます。そのため、フィルター処理された結果のネットセグメントが取得されます。この動作を変更するには、-filter オプションを使用する代わりに、get_nets コマンドに-segments オプションを適用した後、返されたセグメントにfilterコマンドを使用します。下の例を参照してください。

-boundary_type [upper | lower | both] (オプション): 指定の階層ピンのレベル(upper)、ピンまたはポートの下レベル(lower)、またはその両方のネットセグメントを取得します。有効な値は、upper、lower、bothです。デフォルト値はupperです。

注記: このオプションを使用する場合、-of_objects オプションを使用して階層ピンを指定する必要があります。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OKが返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): 指定した検索パターンと一致するネットを検索します。デフォルトの検索パターンはワイルドカード(*)で、プロジェクトに含まれるすべてのネットのリストが返されます。複数の検索パターンを指定して、異なる検索条件に基づいてネットを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、現在のデザインに対して `report_drc` コマンドを実行し、指定の DRC レポートに違反のリストを取得して、NDRV (ドライバーなしのネット ルール) に関連するネットを取得しています。

```
report_drc -name drc_1
get_drc_violations -name drc_1
get_nets -of_objects [get_drc_violations -name drc_1 NDRV*]
```

次の例では、`connect_debug_port` コマンドでデバッグ用にマークされたネットのリストが返されます。

```
get_nets -hier -filter {MARK_DEBUG==1}
```

次の例では、指定の階層ピン オブジェクトに接続されているネットを返し、ピン オブジェクトに接続されているネット セグメントを返して、最後にピン オブジェクトに接続されている最上位ネット セグメントを返しています。

```
get_nets \
  -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]

get_nets -segments \
  -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]

get_nets -top -segments \
  -of [get_pins cpuEngine/or1200_cpu/or1200_sprs/esr_reg[9]_i_3/I0]
```

次の例では、最初のコマンドでは `-filter` が適用されて `IS_INTERNAL` プロパティが設定されたネットが検索された後、`-segment` オプションが適用されてそれらのネットのセグメントが返されます。このコマンドで返されるネット セグメントの数は 0 で、警告が表示されます。2 番目のコマンドでは、すべてのネットのセグメントが返され、その結果がフィルター処理されて `IS_INTERNAL` プロパティが設定されたセグメントが検索されます。検索されたセグメント数は 448 です。

```
llength [get_nets -segments -filter {IS_INTERNAL}]
WARNING: [Vivado 12-1023] No nets matched for command 'get_nets -segments
-filter IS_INTERNAL'.
0
llength [filter [get_nets -segments] {IS_INTERNAL}]
448
```

関連項目

- [connect_debug_port](#)
- [current_instance](#)
- [get_cells](#)
- [get_clocks](#)
- [get_drc_violations](#)
- [get_pins](#)
- [get_ports](#)
- [list_property](#)
- [report_drc](#)

- [report_property](#)

get_nodes

デバイスのノードのリストを取得します。

構文

```
get_nodes [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
          [-uphill] [-downhill] [-flyover] [-from <args>] [-to <args>] [-quiet]
          [-verbose] [<patterns>]
```

戻り値

ノード

使用法

名前	説明
[-of_objects]	指定したネット、タイル、ノード、bel_pin、site_pin、ワイヤ、PIP、speed_model のノード オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-uphill]	-of_objects で指定したサイト ピン、PIP、ノード、またはタイルからドライバー側のノードを取得します。
[-downhill]	-of_objects で指定したサイト ピン、PIP、ノード、またはタイルからロード側のノードを取得します。
[-flyover]	指定したタイルを飛び越すノードを取得します。
[-from]	指定した PIP またはサイト ピンから開始するノードを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-to]	指定したワイヤまたはサイト ピンで終了するノードを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ノード オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

開いているデザインのデバイスのノードで、指定した検索パターンに一致するものをリストします。

デフォルトでは、デバイスのノードすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-of_objects <args>` (オプション): 指定したネット、タイル、ノード、BEL ピン (`bel_pin`)、サイト ピン (`site_pin`)、ワイヤ、スピード モデル (`speed_model`)、または PIP オブジェクトのノードを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>`: 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_nodes` で返されたオブジェクトのリストに、ノードのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ノード オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IS_INPUT_PIN`、`IS_BEL_PIN`、`NUM_WIRES` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-uphill` (オプション): `-of_objects` オプションで指定したオブジェクトのドライバー側 (ロジック ネットワークの前方) にあるノードを返します。

`-downhill` (オプション): `-of_objects` オプションで指定したオブジェクトのロード側 (ロジック ネットワークの後方) にあるノードを返します。

-flyover (オプション): 指定のタイルを通過 (フライオーバー) するノードを返します。

-from <args> (オプション): site_pin または PIP オブジェクトから取得するノードの始点を指定します。このオプションは、-uphill オプションと共に使用できます。デフォルトでは、始点のロード側にあるノードが返されます。デフォルトでは、すべてのノードが返されます。

-to <args> (オプション): site_pin または PIP オブジェクトから取得するノードの終点を指定します。このオプションは、-uphill オプションと共に使用できます。デフォルトでは、指定の終点への始点のロード側にあるノードが返されます。デフォルトでは、すべてのノードが返されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): 指定した検索パターンと一致するノードを検索します。デフォルトの検索パターンはワイルドカード (*) で、デバイスのノードすべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてノードを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定のタイルに関連するノードが返されます。

```
get_nodes -of_objects [get_tiles CLBLM_R_X11Y158]
```

次の例では、指定のノードのロード側にあるノードが返されます。

```
get_nodes -downhill -of_objects [get_nodes LIOB33_SING_X0Y199/IOB_PADOUT0]
```

関連項目

- [get_site_pins](#)
- [get_tiles](#)
- [get_wires](#)
- [list_property](#)
- [report_property](#)

get_objects

1 つまたは複数の HDL スコープで、指定した検索パターンに一致する HDL オブジェクトのリストを取得します。

構文

```
get_objects [-filter <arg>] [-r] [-local] [-regexp] [-nocase] [-quiet]
            [-verbose] [<patterns>...]
```

戻り値

指定した検索パターンに一致するすべてのオブジェクト

使用法

名前	説明
[-filter]	結果のリストに指定した式 (<patterns>) のフィルターを適用します。
[-r]	サブスコープでも検索を実行します。
[-local]	現在のスコープに選択されたサブプログラム フレームのオブジェクトを検索します。
[-regexp]	検索パターンが正規表現で記述されていることを指定します。デザイン オブジェクトを供給するアプリケーションで、検索の実行方法が決定されます。文字列を指定する必要があります。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	検索パターンを指定します。デフォルトは * で、すべての HDL オブジェクトが返されます。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

1 つまたは複数の HDL スコープで、指定した検索パターンに一致する HDL オブジェクトのリストを取得します。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event などがあります。HDL 定数には、Verilog のパラメーターおよび localparam、VHDL ジェネリックおよび定数が含まれます。

HDL スコープは、Verilog のモジュール、関数、タスク、プロセス、begin-end ブロックなど、HDL コードの宣言部分で定義されます。VHDL スコープには、エンティティ/アーキテクチャ定義、関数、プロシージャ、およびプロセスブロックが含まれます。

引数

`-recursive` | `-r` (オプション): 現在のスコープおよびそのサブスコープすべてにコマンドを適用します。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_objects` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。HDL オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、SCOPE、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): HDL オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、現在のスコープのすべての HDL オブジェクトが返されます。検索パターンは、次の 2 つの方法で定義できます。

- `<patterns>`: 取得するオブジェクトの検索パターンのみを指定します。この方法では、現在のスコープおよび `-recursive` を使用している場合はそのサブスコープに含まれるオブジェクトが返されます。
- `<scope>/<pattern>`: スコープ (現在のスコープを基準に指定) と検索パターンを指定します。この場合、現在のスコープから指定の `<scope>` および `-recursive` を使用している場合はそのサブスコープが特定され、検索パターン (`<pattern>`) に一致するオブジェクトが返されます。

例

次の例では、現在のスコープを指定し、そのスコープに含まれるすべての HDL オブジェクトを取得しています。

```
current_scope ./cpuEngine
get_objects
```

次の例では、現在のスコープに含まれるオブジェクト数を取得し、現在のスコープとそのサブスコープに含まれるすべてのオブジェクトの数を取得しています。

```
llength [get_objects]
182
llength [get_objects -recursive ]
2182
```

次の例では、`<scope>/<pattern>` 検索パターンを指定しています。cpuEngine スコープおよびそのサブスコープが特定され、検索パターン `c1*` に一致するオブジェクトが返されます。

```
get_objects -recursive -filter {type == internal_signal} cpuEngine/c1*
/top/cpuEngine/clk_i
/top/cpuEngine/iwb_biu/clk
/top/cpuEngine/iwb_biu/clmode
/top/cpuEngine/or1200_cpu/clk
...
/top/cpuEngine/or1200_immu_top/or1200_immu_tlb/itlb_mr_ram/clk
```

次の例では、現在のスコープとそのサブスコープで `c1` または `ma` で開始する内部信号を取得しています。

```
get_objects -recursive -filter {type == internal_signal} ma* c1*
```

関連項目

- [current_scope](#)
- [list_property](#)
- [report_objects](#)
- [report_property](#)

get_package_pins

パッケージ ピンのリストを取得します。

構文

```
get_package_pins [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
                 [-quiet] [-verbose] [<patterns>]
```

戻り値

パッケージ ピン オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定したサイト、BEL、I/O バンク、パッケージ ピン バイト グループ、パッケージ ピン ニブル、またはポートのパッケージ ピン オブジェクトのリストを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	パッケージ ピンを検索するパターンを指定します。デフォルトは <code>*</code> です。

カテゴリ

XDC、Object (オブジェクト)

説明

ターゲット デバイスに対して選択したパッケージのピンのリストを取得します。デフォルトでは、パッケージのすべてのピンのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_package_pins` で返されたオブジェクトのリストに、ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ピン オブジェクトの場合、結果をフィルターできるプロパティには `IS_CLK_CAPABLE`、`IS_VREF`、`IS_GLOBAL_CLK` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND 「`&&`」および OR 「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したオブジェクトに接続されているパッケージ ピンを取得します。指定可能なオブジェクトはサイト、BEL、I/O バンク、パッケージ ピン バイト グループ、パッケージ ピン ニブル、またはポートです。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン「`<pattern>`」と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ピンを検索するパターンをしていします。デフォルトの検索パターンはワイルドカード「`*`」で、パッケージのすべてのピンが返されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

例

次の例では、UltraScale デバイスの指定したバイト グループに関連付けられているパッケージ ピンのリストを取得しています。

```
get_package_pins -of [get_pkgpin_bytegroups BANK44_BYTE0]
```

次の例では、パッケージのクロック兼用 (CC) ピンの数を取得しています。

```
llength [get_package_pins -filter {IS_CLK_CAPABLE==1}]
```

注記: 検索パターンに一致するピンがない場合は、警告メッセージが表示されます。

関連項目

- [get_pkgpin_bytegroups](#)
- [get_ports](#)
- [get_sites](#)
- [list_property](#)
- [place_ports](#)
- [report_property](#)

get_param

パラメーター値を取得します。

構文

```
get_param [-quiet] [-verbose] <name>
```

戻り値

パラメーター値

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	パラメーター名を指定します。

カテゴリ

[PropertyAndParameter](#) (プロパティおよびパラメーター)

説明

指定したツール パラメーターに現在定義されている値を取得します。これらのパラメーターは、ツールのさまざまな動作を制御するためのユーザー定義可能な環境設定です。各パラメーターが何を設定または制御するかは、`report_param` を参照してください。

引数

`<name>` (必須): 値を取得するパラメーターの名前を指定します。ユーザー定義可能なパラメーターのリストは、`list_param` を実行すると確認できます。このコマンドでは、パラメーターの完全な名前を指定する必要があります。パターン一致は実行されず、1 つのパラメーターのみ指定可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、マルチスレッド プロセスで使用される MaxThreads パラメーターの現在の値が返されます。

```
get_param general.MaxThreads
```

関連項目

- [list_param](#)
- [report_param](#)
- [reset_param](#)
- [set_param](#)

get_partition_defs

PartitionDef のリストを取得します。

構文

```
get_partition_defs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
[<patterns>]
```

戻り値

PartitionDef オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	パーティション定義名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Partition (パーティション)

説明



重要: まず、プロジェクトの PR_FLOW プロパティを TRUE に設定するか、[Tools]→[Enable Partial Reconfiguration] コマンドを使用して、プロジェクトをパースシャル リコンフィギュレーション (PR) プロジェクトとして定義する必要があります。

現在のデザインに含まれるすべてのパーティション 定義 (partitionDef) オブジェクト、または検索パターンに一致する partitionDef のリストを取得します。

パースシャル リコンフィギュレーション フローでは、デザインの階層セルからパーティション定義 (partitionDef) を作成でき、これらの partitionDef にリコンフィギュラブル モジュール (RM) を割り当てることにより、コア デザインと RM の組み合わせに基づいて独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパースシャル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

このコマンドを実行すると、partitionDef オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_partition_defs` で返されたオブジェクトのリストに、パーティション定義のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`partitionDef` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DEFAULT_RM`、`LIBRARY` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ " *RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `partitionDef` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべての `partitionDef` のリストが取得されます。

例

次の例では、現在のプロジェクトに含まれる partitionDef オブジェクトすべてを取得しています。

```
get_partition_defs
```

関連項目

- [create_partition_def](#)
- [delete_partition_defs](#)
- [set_property](#)

get_parts

ツールで使用可能なパーツのリストを取得します。

構文

```
get_parts [-regexp] [-filter <arg>] [-of_objects <args>] [-quiet]
          [-verbose] [<patterns>]
```

戻り値

パーツ オブジェクトのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定したプロジェクト、デザイン、または run オブジェクトのパーツを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	パーツ名を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または <code>-regexp</code> を指定している場合は「.*」です。

カテゴリ

Object (オブジェクト)

説明

指定の検索パターンに一致するパーツのリストを取得します。デフォルトでは、すべてのパーツのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_parts` で返されたオブジェクトのリストに、パーツのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。パーツ オブジェクトの場合、結果をフィルター処理できるプロパティには `DEVICE`、`FAMILY`、`SPEED` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したプロジェクト、デザイン、または run オブジェクトのパーツを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): パーツを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのパーツのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてパーツを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、7vx485t パーツでスピード グレード -1 のもののリストを取得しています。

```
get_parts -filter {DEVICE =~ xc7vx485t* && speed == -1}
```

次の例では、7 シリーズおよび Virtex-6 パーツの数を取得しています。

```
llength [get_parts -regexp {xc7v.* xc6V.*} -nocase]
```

注記: 検索パターンに一致するパーツがない場合は、警告メッセージが表示されます。

関連項目

- [list_property](#)
- [report_property](#)

get_path_groups

現在のデザインのパス グループのリストを取得します。

構文

```
get_path_groups [-regexp] [-nocase] [-quiet] [-verbose] [<patterns>]
```

戻り値

パス グループのリスト

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	パス グループ名を検索するパターンを指定します。デフォルトは <code>*</code> です。

カテゴリ

XDC、Object (オブジェクト)

説明

現在のプロジェクトのタイミング パス グループで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、デザインのすべてのパス グループのリストが取得されます。

デザインで新しいクロックを作成すると、そのクロック ドメイン内のすべてのパスを含むパス グループが自動的に作成されます。パス グループは、`group_path` コマンドを使用して手動で作成することもできます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): パス グループを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべてのパス グループが取得されます。

例

次の例では、デザインのすべてのパス グループのリストを取得しています。

```
get_path_groups
```

次の例では、名前に Clk という文字列を含むパス グループを取得しています。

```
get_path_groups *Clk*
```

注記: 検索パターンに一致するパス グループがない場合は、警告メッセージが表示されます。

関連項目

- [group_path](#)

get_pblocks

現在のデザインの Pblock のリストを取得します。

構文

```
get_pblocks [-regex] [-nocase] [-filter <arg>] [-of_objects <args>]  
            [-include_nested_pblock] [-quiet] [-verbose] [<patterns>]
```

戻り値

Pblock オブジェクトのリスト

使用法

名前	説明
[-regex]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regex を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセルの Pblock を取得します。
[-include_nested_pblock]	入れ子の Pblock のリストを表示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	Pblock 名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Floorplan (フロアプラン)、XDC

説明

現在のプロジェクトで定義されている Pblock で、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトの Pblock すべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regex (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pblocks` で返されたオブジェクトのリストに、Pblock のプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。Pblock オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、GRID_RANGES などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (`*`) を使用すると、定義値が `""` のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (`==`)、不等価 (`!=`)、一致 (`=~`)、不一致 (`!~`) です。数値比較演算子 `<`、`>`、`<=`、および `>=` も使用できます。複数のフィルター式を AND (`&&`) および OR (`||`) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定のセルが割り当てられている Pblock を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-include_nested_pblocks` (オプション): 指定した検索で返された Pblock に含まれている Pblock も取得します。Pblock に含まれている Pblock は、このコマンドで返される Pblock のリストに追加されます。入れ子の Pblock の例は、`create_pblock` コマンドを参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): Pblock を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (`*`) で、プロジェクトのすべての Pblock が返されます。

例

次の例では、現在のプロジェクトの Pblock すべてのリストを取得しています。

```
get_pblocks
```

次の例では、スライス範囲が指定されていない Pblock のリストを取得しています。

```
get_pblocks -filter {GRIDTYPES !~ SLICE}
```

次の例では、指定したセルの Pblock の割り当てを取得しています。

```
get_pblocks -of [get_cells CORE/BR_TOP/RLD67_MUX/REG_PMBIST_C1]
```

次の例では、指定した Pblock と、その Pblock に含まれる Pblock を取得しています。

```
get_pblocks -include_nested_pblocks usbTop
```

関連項目

- [add_cells_to_pblock](#)
- [create_pblock](#)
- [get_cells](#)
- [place_pblocks](#)
- [resize_pblock](#)

get_pins

現在のデザインのピンのリストを取得します。

構文

```
get_pins [-hsc <arg>] [-hierarchical] [-regexp] [-nocase] [-leaf]
         [-filter <arg>] [-of_objects <args>] [-match_style <arg>]
         [-include_replicated_objects] [-quiet] [-verbose] [<patterns>]
```

戻り値

ピン オブジェクトのリスト

使用法

名前	説明
[-hsc]	階層区切り文字を指定します。デフォルトはスラッシュ (/) です。
[-hierarchical]	すべての階層レベルで検索します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-leaf]	-of_objects で指定したネットの下位/グローバル ピンを取得します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したセル、ネット、タイミング パス、クロック、DRC 違反のピンを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-include_replicated_objects]	パターンを検索する際に複製されたオブジェクトを含めます。このオプションは、patterns を指定した場合にのみ有効です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ピン名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

SDC、XDC、Object (オブジェクト)

説明

現在のデザインに含まれるピン オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、開いているデザインの `current_instance` コマンドで指定されている現在のインスタンスに含まれるすべてのピンのリストが返されます。-hierarchical オプションを使用すると、現在のデザインのすべての階層からピンを取得できます。



重要: デザインには多数のピンがあるので、`get_pins` コマンドを使用するとパフォーマンスの問題が発生し、デザイン制約の処理時間が大幅に長くなる可能性があります。多くの場合、`get_pins` コマンドを使用して記述されたデザイン制約は、例に示すように `get_cells` コマンドを使用して記述し直して、Vivado ツールによる制約の処理およびパフォーマンスを大きく向上することが可能です。

`get_pins` コマンドには、物理最適化 (`phys_opt_design`) 中にデザインに追加された複製されたピンも取得するオプションがあります。`-include_replicated_objects` オプションを使用すると、オリジナルのセルのピンと複製されたピンが返されます。このオプションを使用して、セルのピンに適用された制約またはプロパティが複製されたセルのピンにも適用されていることを確認できます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-hsc <arg>` (オプション): デフォルトの階層区切り文字はスラッシュ (/) です。それ以外の階層区切り文字を指定する場合は、このオプションを使用します。

`-hierarchical` (オプション): 現在のインスタンスの階層レベルから開始し、デザイン階層のすべてのレベルからピンを取得します。このオプションを指定しない場合、現在のインスタンスの階層からのピンのみが取得されます。`-hierarchical` を使用する場合、検索パターンは完全な階層セル名ではなく階層の各レベルに適用されるので、検索パターンには階層区切り文字を含めないでください。たとえば、検索パターンとして「U1/*」を指定すると、名前に「U1/」を含むピンが階層の各レベルで検索され、意図した結果が得られない可能性があります。`-hierarchical` を使用した検索の例は、「`get_cells`」を参照してください。

注記: `-regex` と共に使用する場合、検索パターンは完全な階層名と比較され、検索パターンとして「U1/*」を指定した場合に意図した結果が得られます。

`-regex` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regex` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regex.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regex` オプションを使用した場合にのみ適用されます。

`-leaf` (オプション): `-of_object` オプションで指定したオブジェクトに対して、プリミティブまたはブラック ボックス セルからの下位ピンを含めます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pins` で返されたオブジェクトのリストに、ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ピン オブジェクトのリストをフィルター処理するのに使用できるプロパティには、PARENT、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

-of_objects <arg> (オプション): 指定したセル、クロック、タイミング パス、またはネットに接続されているピン、あるいは指定の DRC 違反オブジェクトに関連付けられているピンのリストを取得します。

注記: -of_objects オプションでは、オブジェクトを名前で指定するのではなく、get_* コマンド (get_cells、get_pins など) を使用して指定する必要があります。また、-of_objects を検索パターン (<pattern>) と共に使用することはできません。

-match_style [sdc | ucf] (オプション): 検索パターンが UCF 制約または SDC 制約に一致することを示します。デフォルトは SDC です。

-include_replicated_objects (オプション): 最適化でのセル インスタンスの複製により追加されたピンも含めます。このオプションは、<patterns> を指定している場合にのみ有効で、複製されたセル オブジェクトからのパターンに一致するピンを返します。デフォルトでは、get_pins コマンドでは複製されたセルのピンは返されません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<patterns> (オプション): ピンを検索するパターンをしていします。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべてのピンのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、指定したセルに接続されているピンのリストを取得しています。

```
get_pins -of_objects [get_cells usb*]
```

注記: 検索パターンに一致するピンがない場合は、警告メッセージが表示されます。

次の例では、get_pins コマンドのパフォーマンスを向上するために get_cells コマンドを使用しています。

```
[get_pins -hier * -filter {NAME=~xx*/yy*}]
```

上記のコマンドは、次のように記述し直すことができます。

```
[get_pins -filter {REF_PIN_NAME=~yy*} -of [get_cells -hier xx*]]
```

次の例では、set_max_delay 制約を記述し直すことでパフォーマンスの向上が可能であることを示しています。

```
set_max_delay 15 -from [get_pins -hier \
  -filter name=~*/aclk_dpram_reg*/*/CLK] \
  -to [get_cells -hier -filter name=~*/bclk_dout_reg*] -datapath_only
```

上記のコマンドは、次のように記述し直すことができます。

```
set_max_delay 15 -from [get_pins -of \
  [get_cells -hier -filter name=~*/aclk_dpram_reg*/*/] \
  -filter {REF_PIN_NAME == CLK}] \
  -to [get_pins -of [get_cells -hier -filter {name =~ */bclk_dout_reg*}]
  \
  -filter {REF_PIN_NAME == D}] -datapath_only
```



ヒント: 2 つ目のコマンド構文は 1 つ目のコマンド構文よりも複雑ですが、パフォーマンスは大幅に向上します。

次の例では、現在のデザインに対して report_drc コマンドを実行し、DRC 違反に関連するピンを取得しています。

```
report_drc -name drc_1
get_pins -of_objects [get_drc_violations]
```

関連項目

- [current_instance](#)
- [get_cells](#)
- [get_drc_violations](#)
- [list_property](#)
- [phys_opt_design](#)
- [report_drc](#)
- [report_property](#)
- [set_max_delay](#)

get_pips

現在のデバイスのプログラマブル インターコネクト ポイント (PIP) のリストを取得します。

構文

```
get_pips [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-uphill]
          [-downhill] [-from <args>] [-to <args>] [-quiet] [-verbose]
          [<patterns>]
```

戻り値

PIP

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したサイト、タイル、ワイヤ、ノード、PIP、またはネットの PIP を取得します。
[-uphill]	指定したワイヤまたは PIP からドライバー側の PIP を取得します。
[-downhill]	指定したワイヤまたは PIP からロード側の PIP を取得します。
[-from]	指定した PIP またはサイト ピンから開始する PIP の順序付きリストを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-to]	指定したワイヤまたはサイト ピンで終了する PIP の順序付きリストを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	PIP を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

プログラマブル インターコネクト ポイント (PIP) は、ロジック ネットワークを接続するのに使用する物理配線パスです。このコマンドは、デバイス上の PIP で、指定した検索パターンに一致するものをリストします。このコマンドを実行するには、デザインを開いておく必要があります。

デフォルトでは、デバイス上の PIP すべてのリストが取得されます。ただし、デバイス上には PIP が多数あるので、このコマンドをデフォルトで実行することはお勧めしません。-of_objects オプションを使用して返される PIP の数を制限してください。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pips` で返されたオブジェクトのリストに、PIP のプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。PIP オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IS_DIRECTIONAL`、`FROM_PIN` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (! =~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したサイト、タイル、ワイヤ、ノード、PIP、またはネット オブジェクトの PIP を取得します。`-of_objects` オプションを常に使用し、`get_pips` コマンドのランタイムとメモリ使用量を制限することをお勧めします。そうしないと、返される PIP の数が極端に大きくなる可能性があります。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-uphill` (オプション): 指定したワイヤまたは PIP のドライバー側 (ロジック ネットワークの前方) にある PIP を返します。

`-downhill` (オプション): 指定したワイヤまたは PIP のロード側 (ロジック ネットワークの後方) にある PIP を返します。

`-from <args>` (オプション): ワイヤまたは `site_pin` オブジェクトから取得する PIP の始点を指定します。このオプションは、`-uphill` オプションと共に使用できます。デフォルトでは、始点のロード側にある PIP が返されます。デフォルトでは、すべての PIP が返されます。

`-to <args>` (オプション): ワイヤまたは `site_pin` オブジェクトから取得する PIP の終点を指定します。このオプションは、`-uphill` オプションと共に使用できます。デフォルトでは、指定の終点への始点のロード側にある PIP が返されます。デフォルトでは、すべての PIP が返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): PIP を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、デバイス上の PIP すべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて PIP を検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、指定のタイルに関連する PIP が返されます。

```
get_pips -of_object [get_tiles DSP_R_X9Y75]
```

関連項目

- [get_sites](#)
- [get_tiles](#)
- [get_wires](#)
- [list_property](#)
- [report_property](#)

get_pkgpin_bytegroups

パッケージ ピン バイト グループのリストを取得します。

構文

```
get_pkgpin_bytegroups [-regexp] [-nocase] [-filter <arg>]  
                        [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

pin_group オブジェクト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した package_pin、iobank、サイト、またはポートの pin_group を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	pin_group を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[XDC、Object \(オブジェクト\)](#)

説明

現在のザイリンクス UltraScale デバイスの I/O バンクにあるバイトグループのリストを取得します。

7 シリーズ デバイスでは、I/O バンクの階層に I/O バンクとパッケージ ピンの 2 つのオブジェクト タイプが含まれます。ザイリンクス UltraScale FPGA デバイスでは、バイトグループとニブルが追加されています。

UltraScale デバイスでのこれらのオブジェクトの関係は、次のように定義されています。

- I/O バンク (iobank) には 2 ～ 4 個のバイトグループが含まれます。
- 各パッケージ ピン バイトグループ (pkgpin_bytegroup) には、2 つのニブル (上位または下位) が含まれ、13 本のパッケージ ピンがあります。
- 各パッケージ ピン ニブル (pkgpin_nibble) には 6 ～ 7 本のピンが含まれ、pkgpin_bytegroup の上位または下位ニブルです。
- パッケージ ピン (package_pin) は、I/O バンク (iobank)、pkgpin_bytegroup、または pkgpin_nibble の 1 つのピンです。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pkgpin_bytegroup` で返されたオブジェクトのリストに、オブジェクトのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`pkgpin_bytegroup` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、IOBANK があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定した iobank、サイト、ポート、または `package_pin` オブジェクトの `pkgpin_bytegroup` を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `pkgpin_bytegroup` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべての `pkgpin_bytegroup` が返されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

例

次の例では、ターゲット デバイスのピンすべてのリストを取得しています。

```
get_pkgpin_bytegroups -of [get_iobanks 44]
```

関連項目

- [get_iobanks](#)
- [get_package_pins](#)
- [list_property](#)
- [report_property](#)

get_pkgpin_nibbles

パッケージ ピン ニブルのリストを取得します。

構文

```
get_pkgpin_nibbles [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

ピン ニブル

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定した <code>package_pin</code> 、 <code>iobank</code> 、サイト、またはポートのピンニブルを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	<code>pin_nibble</code> を検索するパターンを指定します。デフォルトは <code>*</code> です。

カテゴリ

[XDC、Object \(オブジェクト\)](#)

説明

現在のザイリンクス UltraScale デバイスの I/O バンクにあるニブル (ハーフバイト) のリストを取得します。

7 シリーズ デバイスでは、I/O バンクの階層に I/O バンクとパッケージ ピンの 2 つのオブジェクト タイプが含まれます。ザイリンクス UltraScale FPGA デバイスでは、バイト グループとニブルが追加されています。

UltraScale デバイスでのこれらのオブジェクトの関係は、次のように定義されています。

- I/O バンク (`iobank`) には 2 ～ 4 個のバイト グループが含まれます。
- 各パッケージ ピン バイト グループ (`pkgpin_bytegroup`) には、2 つのニブル (上位または下位) が含まれ、13 本のパッケージ ピンがあります。
- 各パッケージ ピン ニブル (`pkgpin_nibble`) には 6 ～ 7 本のピンが含まれ、`pkgpin_bytegroup` の上位または下位ニブルです。
- パッケージ ピン (`package_pin`) は、I/O バンク (`iobank`)、`pkgpin_bytegroup`、または `pkgpin_nibble` の 1 つのピンです。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pkgpin_nibbles` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。`pkgpin_nibble` ニブル オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、IOBANK、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定した `package_pin`、`iobank`、サイト、またはポート オブジェクトの `pkgpin_nibble` オブジェクトを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `pkgpin_nibble` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトのすべての `pkgpin_nibble` オブジェクトが返されます。

例

次の例では、指定の I/O バンクに関連付けられている上位ニブルが返されます。

```
get_pkgpin_nibbles -of [get_iobanks 44] -filter {TYPE == U}
```

関連項目

- [get_iobanks](#)
- [get_package_pins](#)
- [list_property](#)
- [report_property](#)

get_ports

現在のデザインに含まれるポートのリストを取得します。

構文

```
get_ports [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
          [-match_style <arg>] [-scoped_to_current_instance] [-no_traverse]
          [-prop_thru_buffers] [-quiet] [-verbose] [<patterns>]
```

戻り値

ポート オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したネット、インスタンス、サイト、クロック、タイミングパス、I/O 規格、I/O バンク、パッケージピン、DRC 違反のポートを取得します。
[-match_style]	パターン一致のスタイルを指定します。有効な値は ucf、sdc で、デフォルトは sdc です。
[-scoped_to_current_instance]	現在のインスタンスとして指定されているインスタンスのピンに対してパターン検索を実行し、最上位の接続ポートを返します。
[-no_traverse]	接続されている最上位ターミナルを検索しません。
[-prop_thru_buffers]	検索範囲として指定されているインスタンスのピンに接続されている最上位ターミナルを検索する際に、検索がバッファを介して伝搬されるようにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ポート名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

SDC、XDC、Object (オブジェクト)

説明

現在のデザインに含まれるポート オブジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、デザインに含まれるすべてのポートのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_ports` で返されたオブジェクトのリストに、ポートのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。ポート オブジェクトのリストをフィルター処理するのに使用できるプロパティには、PARENT、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ " *RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したネット、BEL、サイト、クロック、タイミングパス、I/O 規格 (io_standard)、I/O バンク、またはパッケージ ピン (package_pin) オブジェクトに接続されているポート、あるいは指定の DRC 違反オブジェクトに関連付けられているポートを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-match_style [sdc | ucf]` (オプション): 検索パターンが UCF 制約または SDC 制約に一致することを示します。デフォルトは SDC です。

`-scoped_to_current_instance` (オプション): 現在のインスタンスの最上位ピンを返します。<patterns> を使用して現在のインスタンスのインスタンス ピンを検索する場合に適用されます。

`-prop_thru_buffers` (オプション): 現在のインスタンスのピンに接続されている最上位ポートを返します。このオプションは、`-scoped_to_current_instance` オプションと共に使用して、検索を現在のインスタンスのピンからバッファを介してデザインの最上位ポートに伝搬させることができます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): ポートを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべてのポートのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてポートを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つの要素として指定します。

例

次の例では、指定したセルに接続されているピンのリストが取得されます。

```
get_ports -of_objects [lindex [get_cells] 1]
```

注記: 検索パターンに一致するポートがない場合は、警告メッセージが表示されます。

次の例では、現在のデザインに対して `report_drc` コマンドを実行し、指定の DRC レポートに違反のリストを取得して、NSTD (未指定 I/O 規格ルール) に関連するポートを取得しています。

```
report_drc -name drc_1
get_drc_violations -name drc_1
get_ports -of_objects [get_drc_violations -name drc_1 NSTD*]
```

次の例では、現在のインスタンスとなるセルを指定し、現在のインスタンスのピンを返し、それらのピンに接続されている最上位ポートを返しています。

```
current_instance [get_cells dac_spi*]
get_ports -scoped_to_current_instance
get_ports -scoped_to_current_instance -prop_thru_buffers
```

関連項目

- [current_instance](#)
- [get_cells](#)
- [get_clocks](#)
- [get_drc_violations](#)
- [get_nets](#)
- [get_timing_paths](#)

- [list_property](#)
- [report_drc](#)
- [report_property](#)

get_pplocs

ピンまたはネット上の PPLOC をレポートします。

構文

```
get_pplocs -nets <args> -pins <args> [-count] [-unlocked] [-locked]
          [-level <arg>] [-quiet] [-verbose]
```

戻り値

PPLOC ノードまたは PPLOC の数

使用法

名前	説明
-nets	PPLOC をレポートするネットを 1 つ以上指定します。
-pins	PPLOC をレポートするピンを 1 つ以上指定します。
[-count]	PPLOC の数をレポートし、PPLOC またはノードの名前はレポートしません。
[-unlocked]	固定されていない PPLOC のみをレポートします。
[-locked]	固定されている PPLOC のみをレポートします。-level オプションを使用して固定レベルを指定してください。
[-level]	固定レベルを指定します。有効な値は placement および routing で、デフォルトは placement です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

get_pr_configurations

パーティション コンフィギュレーションのリストを取得します。

構文

```
get_pr_configurations [-regexp] [-nocase] [-filter <arg>] [-quiet]
                     [-verbose] [<patterns>]
```

戻り値

コンフィギュレーション オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	パーティション コンフィギュレーション名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Partition \(パーティション\)](#)

説明

現在のプロジェクトに含まれる PR コンフィギュレーション オブジェクトのリストを取得します。

パーシャル リコンフィギュレーション (PR) デザイン フローでは、PR コンフィギュレーションでパーティション定義 (partitionDef) の特定のインスタンスに割り当てるリコンフィギュラブル モジュール (RM) を指定できます。このフローでは、スタティック デザインと 1 つ以上の RM の組み合わせに基づいて、独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパーシャル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

このコマンドを実行すると、PR コンフィギュレーション オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_pr_configurations` で返されたオブジェクトのリストに、コンフィギュレーションのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): PR コンフィギュレーションを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべてのコンフィギュレーションのリストが取得されます。

例

次の例では、現在のプロジェクトに含まれる PR コンフィギュレーション オブジェクトすべてを取得しています。

```
get_pr_configurations
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)
- [create_reconfig_module](#)
- [delete_pr_configurations](#)
- [setup_pr_configurations](#)

get_primitives

パーツで使用可能な UNISIM プリミティブのリストを取得します。

構文

```
get_primitives [-regexp] [-nocase] [-filter <arg>] [-part <arg>]
               [-retarget] [-macro] [-hierarchy] [-quiet] [-verbose] [<patterns>]
```

戻り値

プリミティブ タイプ

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-part]	プリミティブを取得するパーツを指定します。
[-retarget]	現在の (または指定した) パーツに自動的にリターゲットされるプリミティブ タイプを含めます。
[-macro]	ロジック ゲートなど、より基本的でネイティブにサポートされるプリミティブに常に変換されるプリミティブを含めます。
[-hierarchy]	インプリメンテーション中に自動的に最下位セルの階層に展開されるプリミティブ タイプを含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	プリミティブ タイプを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)

説明

指定のデバイスでサポートされるすべてのプリミティブのリストを取得します。このコマンドは、エラボレート済みデザイン、合成済みデザイン、またはインプリメント済みデザインに対して使用でき、その場合は現在のデザインの PART で指定されているパーツでサポートされるプリミティブが返されます。-part オプションを使用して、プリミティブを取得するデバイスを指定することも可能です。

デフォルトでは、ターゲット パーツに変更せずに配置可能なネイティブ プリミティブが返されます。-retarget、-macro、および -hierarchy オプションを使用すると、該当するプリミティブがリストに追加されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter` <args> (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_primitives` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <, >, <=, および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-part` <arg> (オプション): 指定したパーツで使用可能なプリミティブのリストを取得します。

`-retarget` (オプション): 現在のパーツまたは指定したパーツに自動的にリターゲットされるプリミティブを含めます。

`-macro` (オプション): Include ロジック ゲートなどのより基本的なプリミティブに変換されるマクロ プリミティブを含めます。

`-hierarchy` (オプション): インプリメンテーション中に自動的に最下位セルの階層に展開されるプリミティブを含めます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (必須): プリミティブを検索するパターンを指定します。

例

次の例では、現在のパーツのネイティブ プリミティブとマクロ プリミティブを取得しています。

```
get_primitives -macro
```

関連項目

- [get_cells](#)
- [get_libs](#)
- [get_lib_pins](#)
- [get_lib_cells](#)
- [get_parts](#)
- [list_property](#)
- [report_property](#)

get_projects

プロジェクトのリストを取得します。

構文

```
get_projects [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
             [<patterns>]
```

戻り値

プロジェクト オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	プロジェクト名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Project (プロジェクト)

説明

現在開いているプロジェクトで、検索パターンに一致するもののリストを取得します。デフォルトでは、開いているプロジェクトすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_projects` で返されたオブジェクトのリストに、プロジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。プロジェクト オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NAME`、`DIRECTORY`、`TARGET_LANGUAGE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): プロジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのパーツのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてプロジェクトを検索できます。

例

次の例では、開いているプロジェクトすべてのリストを取得しています。

```
get_projects
```

次の例では、project_found という変数が get_projects コマンドで返されるプロジェクト リストの長さに設定され、そのプロジェクトが検出されたか、検出されなかったかが表示されます。

```
set project_found [llength [get_projects ISC*] ]
if {$project_found > 0} {puts "Project Found."} \
    else {puts "No Projects Found."}
```

注記: 検索パターンに一致するプロジェクトがない場合は、警告メッセージが表示されます。

関連項目

- [create_project](#)
- [current_project](#)
- [open_project](#)

get_property

オブジェクトのプロパティを取得します。

構文

```
get_property [-min] [-max] [-quiet] [-verbose] <name> <object>
```

戻り値

プロパティ 値

使用法

名前	説明
[-min]	最小値のみを返します。
[-max]	最大値のみを返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	値を取得するプロパティの名前を指定します。
<object>	プロパティを取得するオブジェクトを指定します。

カテゴリ

Object (オブジェクト)、PropertyAndParameter (プロパティおよびパラメーター)、XDC

説明

指定したオブジェクトの指定したプロパティの現在の値を取得します。複数のオブジェクトを指定した場合、値のリストが返されます。

オブジェクトに指定のプロパティが設定されていない場合、または値が設定されていない場合は、`get_property` コマンドは何も返しません (NULL 文字列)。複数のオブジェクトを指定した場合、NULL 文字列が返される値のリストに追加されます。

`get_property` コマンドで複数のオブジェクトを指定する際に `-min` または `-max` オプションを使用すると、指定したプロパティの最小値または最大値を返すことができます。この機能は、タイミング制約を設定する際に有益です。



推奨: 値が数値のプロパティでは、最小値/最大値は数値に基づきます。その他のプロパティでは、文字列の並べ替え順に基づきます。

このコマンドを実行すると、値または値のリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-min` (オプション): 複数のオブジェクト (<objects>) を指定した場合に、<name> で指定されたプロパティの値を調べ、オブジェクトのリストから最小値のみを返します。最小値/最大値は、値が数値のプロパティでは数値に基づき、その他のプロパティでは文字列の並べ替え順に基づいて決定されます。

-max (オプション): 複数のオブジェクト (<objects>) を指定した場合に、<name> で指定されたプロパティの値を調べ、オブジェクトのリストから最大値のみを返します。最小値/最大値は、値が数値のプロパティでは数値に基づき、その他のプロパティでは文字列の並べ替え順に基づいて決定されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<name> (必須): 値を取得するプロパティの名前を指定します。大文字/小文字は区別されません。

<objects> (必須): 指定のプロパティの値を取得する 1 つまたは複数のオブジェクトを指定します。

例

次の例では、指定したセルの NAME プロパティを取得しています。

```
get_property NAME [lindex [get_cells] 3]
```

次の例では、指定したクロック オブジェクトの PERIOD プロパティの最小値が返されます。

```
get_property -min PERIOD [get_clocks]
```

次の例では、指定したポートの SITE プロパティの文字列に基づく並べ替え順を示しています。

```
get_property SITE [get_ports]
IOB_X1Y75 IOB_X1Y76 IOB_X1Y98 IOB_X1Y125 IOB_X0Y94 IOB_X1Y95 IOB_X1Y96
IOB_X1Y93 IOB_X1Y94

get_property -min SITE [get_ports]
IOB_X0Y94

get_property -max SITE [get_ports]
IOB_X1Y98
```

注記: ポート オブジェクトでは IOB_X1Y125 がサイトの最大値ですが、プロパティ 値の並べ替え順のため、IOB_X1Y98 が返されます。

関連項目

- [create_property](#)
- [get_cells](#)
- [get_ports](#)
- [list_property](#)
- [list_property_value](#)
- [report_property](#)
- [reset_property](#)
- [set_property](#)

get_qor_suggestions

QoR 推奨項目のリストを取得します。

構文

```
get_qor_suggestions [-filter <arg>] [-quiet] [-verbose] [<IDs>]
```

戻り値

QoR 推奨項目オブジェクトのリスト

使用法

名前	説明
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<IDs>]	名前を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[Feasibility \(設計実現可能性\)](#)、[Timing \(タイミング\)](#)

説明

report_qor_suggestions コマンドで作成されたか read_qor_suggestions コマンドを使用してデザインに読み込まれた既存の QoR 推奨項目オブジェクトを取得します。選択した QoR 推奨項目オブジェクトは、write_qor_suggestions コマンドを使用してファイルに書き込むことができます。レポート コマンドで作成された推奨項目は、delete_qor_suggestions を実行するか、別のレポートで上書きされるまで、メモリに保持されます。

このコマンドを実行すると、開いているデザインの既存の QoR 推奨項目オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-filter <arg> (オプション): 返された推奨項目オブジェクトに指定した式のフィルターを適用します。-filter オプションを使用すると、コマンドで返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、report_property または list_property コマンドで確認できます。qor_suggestion オブジェクトのリストをフィルター処理するのに使用できるプロパティには、ENABLED、CATEGORY、および AUTO があります。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<IDs> (オプション): 選択する QoR 推奨項目 ID のリストを指定します。

例

個々の推奨項目を返すには、フル ID を指定します。

```
get_qor_suggestions RQS_CLOCK-5_1-1
```

推奨項目の命名規則により、ワイルドカードを直接使用することはできませんが、次のように `filter` コマンドで `-regexp` オプションを指定すると使用できます。

```
filter -regexp [get_qor_suggestions] {NAME =~ RQS_CLOCK.*}
```

特定のカテゴリの推奨項目オブジェクトのみを取得できます。カテゴリには、Timing (タイミング)、Congestion (密集)、Utilization (使用率)、XDC、Clocking (クロッキング)、および Strategy (ストラテジ) があります。次の例は、Timing カテゴリの推奨項目を取得します。

```
get_qor_suggestions -filter {CATEGORY==Timing}
```

関連項目

- [delete_qor_suggestions](#)
- [read_qor_suggestions](#)
- [report_qor_suggestions](#)
- [write_qor_suggestions](#)

get_reconfig_modules

リコンフィギャラブル モジュールのリストを取得します。

構文

```
get_reconfig_modules [-regexp] [-nocase] [-filter <arg>]
                    [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

リコンフィギャラブル モジュール オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した partition_def の reconfig_module オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	リコンフィギャラブル モジュール名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Partition (パーティション)

説明

現在のデザインに含まれるリコンフィギャラブル モジュール (RM) で、検索パターンに一致するもののリストを取得します。デフォルトでは、現在のプロジェクトに含まれるすべての RM のリストが返されます。

このコマンドを実行すると、RM オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_reconfig_modules` で返されたオブジェクトのリストに、RM のプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。RM オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IS_GATE_LEVEL`、`PARTITION_DEF` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (=~)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したパーティション定義 (partition_def) オブジェクトのリコンフィギュレーション モジュール (reconfig_module) を取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): リコンフィギュラブル モジュール (RM) を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべての RM のリストが取得されます。

例

次の例では、プロジェクトのすべてのゲート レベルまたはネットリスト ベースの RM を取得しています。

```
get_reconfig_modules -filter {IS_GATE_LEVEL}
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)

- [delete_reconfig_modules](#)
- [get_partition_defs](#)
- [set_property](#)

get_report_configs

設定可能なレポート オブジェクトのリストを取得します。

構文

```
get_report_configs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
                  [-quiet] [-verbose] [<patterns>]
```

戻り値

設定可能なレポート オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定した run のレポート オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	レポート名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Report (レポート)

説明

create_report_config コマンドで作成されたレポート オブジェクトを取得します。

検索パターンに一致するレポート オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_report_configs` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。レポート オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`REPORT_TYPE`、`RUN_STEP`、`STATE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): `get_runs` コマンドを使用して指定したデザイン run オブジェクトに関連付けられているレポートを取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): レポート オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべてのレポート オブジェクトが取得されます。

例

次の例では、現在のプロジェクトに含まれるレポート オブジェクトすべてを取得しています。

```
get_report_configs
```

次の例では、現在のプロジェクトに含まれるレポート オブジェクトで、`opt_design` に関連付けられているものを取得しています。

```
get_report_configs -filter {RUN_STEP == opt_design}
```

関連項目

- [create_report_config](#)
- [delete_report_configs](#)
- [generate_reports](#)
- [get_runs](#)

get_runs

run のリストを取得します。

構文

```
get_runs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
         [-verbose] [<patterns>]
```

戻り値

run オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したリコンフィギャラブル モジュール (reconfig_module) の run オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	run 名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、Project (プロジェクト)

説明

現在のプロジェクトの合成 run およびインプリメンテーション run で、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、プロジェクトで定義されている run すべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_runs` で返されたオブジェクトのリストに、`run` のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。`run` オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`CONSTRSET`、`IS_IMPLEMENTATION`、`IS_SYNTHESIS`、`FLOW` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定した `reconfig_module` オブジェクトのデザイン `run` を取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): `run` を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトで定義されているすべての `run` のリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて `run` を検索できます。

例

次の例では、現在のプロジェクトで完了していない `run` すべてのリストを取得しています。

```
get_runs -filter {PROGRESS < 100}
```

次の例では、指定の検索パターンに一致する run のリストを取得しています。

```
get_runs imp*
```

注記: 検索パターンに一致する run がない場合は、警告メッセージが表示されます。

関連項目

- [create_run](#)
- [current_run](#)
- [get_reconfig_modules](#)
- [report_property](#)

get_scopes

スコープの子 HDL スコープのリストを取得します。

構文

```
get_scopes [-filter <arg>] [-regexp] [-nocase] [-r] [-quiet] [-verbose]  
[<patterns>...]
```

戻り値

HDL スコープ オブジェクト

使用法

名前	説明
[-filter]	結果のリストに指定した式 (<patterns>) のフィルターを適用します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-r]	子スコープでも検索を実行します (glob または -regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	スコープを検索するパターン文字列を指定します。デフォルトは * で、すべての子スコープが取得されます。

カテゴリ

Simulation (シミュレーション)

説明

現在のスコープまたは指定したスコープの子 HDL スコープのリストを取得します。

このコマンドを実行すると、スコープ オブジェクトのリストまたはエラーが返されます。

引数

-recursive | -r (オプション): 子スコープでも検索を実行します。

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 返された結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_scopes` で返されたオブジェクトのリストに、スコープのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): スコープ オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、現在のスコープの子であるすべてのスコープのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてスコープ オブジェクトを検索できます。

注記: 複数の検索パターンは「`{ }`」で囲み、1 つのエレメントとして指定します。

例

次の例では、指定したスコープの子スコープを、子スコープでも検索を実行してすべて取得しています。

```
get_scopes -r /testbench/dut
```

関連項目

- [current_scope](#)
- [report_scopes](#)

get_selected_objects

選択したオブジェクトを取得します。

構文

```
get_selected_objects [-primary] [-quiet] [-verbose]
```

戻り値

選択したオブジェクトのリスト

使用法

名前	説明
<code>[-primary]</code>	選択規則に従って選択されたオブジェクトは含みません。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[GUIControl \(GUI 制御\)](#)

説明

現在 Vivado IDE で `select_objects` コマンドにより選択されているオブジェクトを取得します。プライマリ選択オブジェクトとセカンダリ選択オブジェクトの両方を取得できます。

注記: この Tcl コマンドは、Vivado を GUI モードで実行している場合にのみ機能します。

プライマリ オブジェクトとは直接選択されたオブジェクトで、セカンダリ オブジェクトとは [Tools]→[Settings] をクリックして開いた [Settings] ダイアログ ボックスの [Selection Rules] ページで現在定義されている選択規則に従って選択されるオブジェクトを指します。選択規則の設定方法は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

このコマンドを実行すると、オブジェクトが 1 つであっても、選択したオブジェクトの Tcl リストが返されます。これは、`current_instance` コマンドなど、オブジェクトのリストを使用できない Vivado ツール コマンドで問題となることがあります。その場合は、`lindex` を使用して、`get_selected_objects` コマンドで返されたリストから特定のオブジェクトを `current_instance` コマンドに渡します。

```
current_instance [lindex [get_selected_objects] 0]
```

引数

`-primary` (オプション): プライマリ選択オブジェクトのみを返します。セカンダリ オブジェクトは返しません。デフォルトでは、`get_selected_objects` コマンドで現在選択されているオブジェクトがすべて返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在選択されているすべてのオブジェクト (プライマリとセカンダリの両方) のプロパティがレポートされます。

```
report_property [get_selected_objects]
```

関連項目

- [get_highlighted_objects](#)
- [get_marked_objects](#)
- [highlight_objects](#)
- [mark_objects](#)
- [select_objects](#)

get_simulators

登録されているシミュレータのリストを取得します。

構文

```
get_simulators [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
               [<patterns>]
```

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	シミュレータ名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[ToolLaunch \(ツール起動\)](#)、[Simulation \(シミュレーション\)](#)

説明

Vivado Design Suite 統合シミュレーション環境で使用するよう登録されているシミュレータのリストを取得します。

Vivado Design Suite では、統合シミュレーション環境で使えるシミュレータがあらかじめ登録されています。
register_simulator コマンドを使用して、サードパーティ シミュレータを登録することもできます。

このコマンドを実行すると、登録されているシミュレータの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.*」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド regexp はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

-nocase (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、-regexp オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_simulators` で返されるオブジェクトのリストに、登録されたシミュレータのプロパティ値に基づいて適用できます。登録されているシミュレータに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。シミュレータ オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、DESCRIPTION、TCLPROC、BOOTSTRAP などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): シミュレータを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、登録されているすべてのシミュレータが取得されます。

例

次の例では、登録されているすべてのシミュレータが返されます。

```
get_simulators
```

関連項目

- [launch_simulation](#)

get_site_pins

サイト ピン (site_pin) のリストを取得します。

構文

```
get_site_pins [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]  
              [-quiet] [-verbose] [<patterns>]
```

戻り値

site_pin オブジェクト

使用法

名前	説明
[-of_objects]	指定したサイト、xdef_site、ノード、ピン、ネット、bel_pin の site_pin オブジェクトを取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	site_pin オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

開いているデザインに含まれる指定のサイト、ノード、論理セル ピン、またはネット オブジェクトのサイト ピンのリストを返します。

このコマンドでは、実行時間を手短縮し、計算リソースを削減するため、-of_objects オプションを使用することをお勧めします。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-of_objects <args> (オプション): 指定したサイト、ノード、ピン、またはネット オブジェクトのサイト ピンを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_site_pins` で返されたオブジェクトのリストに、サイト ピンのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。サイト ピン オブジェクトの場合、結果をフィルターできるプロパティには `IS_CLOCK`、`IS_DATA`、`IS_PART_OF_BUS` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): サイト ピンを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトのサイト ピンすべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてサイト ピンを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定したネットのサイト ピンが返されます。

```
get_site_pins -of_objects [get_nets *Clk]
IOB_X1Y24/I
```

次の例では、サイト SLICE_X21Y92 に関連する出力サイト ピンが返されます。

```
get_site_pins -of_objects [get_sites SLICE_X21Y92] -filter
{DIRECTION==OUT}
SLICE_X21Y92/A SLICE_X21Y92/AMUX SLICE_X21Y92/AQ
SLICE_X21Y92/B SLICE_X21Y92/BMUX SLICE_X21Y92/BQ
SLICE_X21Y92/C SLICE_X21Y92/CMUX SLICE_X21Y92/COUT
SLICE_X21Y92/CQ SLICE_X21Y92/D SLICE_X21Y92/DMUX
SLICE_X21Y92/DQ
```

関連項目

- [get_nets](#)
- [get_nodes](#)
- [get_pins](#)
- [get_sites](#)
- [list_property](#)
- [report_property](#)

get_site_pips

指定したオブジェクトのサイト PIP (site_pip) のリストを取得します。

構文

```
get_site_pips [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]  
              [-quiet] [-verbose] [<patterns>]
```

戻り値

site_pip オブジェクト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したサイトの site_pip を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	site_pip を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[XDC](#)

説明

プログラマブル インターコネクト ポイント (PIP) は、ロジック ネットワークを接続するのに使用する物理配線パスです。このコマンドは、指定のサイトの PIP で、指定した検索パターンに一致するものをリストします。このコマンドを実行するには、デザインを開いておく必要があります。

このコマンドでは、-of_objects オプションを使用して PIP を取得するサイトを指定する必要があります。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

索引

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_site_pips` で返されたオブジェクトのリストに、サイト PIP のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。PIP オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`IS_DIRECTIONAL`、`FROM_PIN` があります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): このオプションは `get_sites` コマンドと共に使用でき、指定したサイトの PIP を返します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): PIP を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、`-of_objects` オプションで指定したサイトの PIP すべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてピンを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエLEMENTとして指定します。

例

次の例では、デバイス上の指定したサイト範囲に関連する指定した BEL のピンが返されます。

```
get_site_pips -of_objects [get_sites SLICE_X21Y92]
```

関連項目

- [get_sites](#)
- [list_property](#)
- [report_property](#)

get_sites

サイトのリストを取得します。

構文

```
get_sites [-regexp] [-filter <arg>] [-nocase] [-range <args>]
          [-of_objects <args>] [-quiet] [-verbose] [<patterns>]
```

戻り値

サイト オブジェクトのリスト

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-range]	サイトを検索する範囲を指定します。範囲は、2 つのサイト名で定義します。
[-of_objects]	指定した SLR、タイル、BEL、ピン、package_pin、ポート、Pblock、ネット、site_type、io_bank、セル、clock_region、または drc_violation のサイトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	サイトを検索するパターンを指定します。ボンディングされたサイトも、パッケージ ピン名に一致します。デフォルトは * です。

カテゴリ

[XDC](#)、[Object \(オブジェクト\)](#)

説明

ターゲット デバイスのサイトで、指定した検索パターンに一致するもののリストを取得します。デフォルトでは、ターゲット デバイスのサイトすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_sites` で返されたオブジェクトのリストに、サイトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。サイト オブジェクトの場合、結果をフィルター処理できるプロパティには `SITE_TYPE`、`IS_USED`、`NUM_INPUTS`、`NUM_OUTPUTS` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-range <arg>` (オプション): 指定の範囲内のサイトを取得します。サイトの範囲は、同じ `SITE_TYPE` の 2 つのサイト値 (例: {SLICE_X2Y12 SLICE_X3Y15}) で指定します。サイトの `SITE_TYPE` は、`report_property` コマンドで確認できます。

注記: タイプの異なる 2 つのサイトで範囲を指定すると、エラーが発生します。

`-of_objects <arg>` (オプション): 指定したオブジェクトのサイトを取得します。指定可能なオブジェクトは、タイル、BEL、ピン、パッケージピン、ポート、Pblock、I/O バンク、セル、クロック領域、または指定の DRC 違反オブジェクトに関連付けられているサイトです。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): サイトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、ターゲット デバイスのサイトすべてのリストが取得されます。

例

次の例は、ターゲット デバイスのサイトすべてのリストを取得します。

```
get_sites
```

次の例は、デバイスで現在使用されていないサイト数を返します。これらのコマンドは、同じ結果を返します。2 番目のコマンドは、`IS_USED` ブール プロパティを直接評価します。

```
llength [get_sites -filter {IS_USED==0}]
-or-
llength [get_sites -filter !IS_USED]
```

注記: 検索パターンに一致するサイトがない場合は、警告メッセージが表示されます。

次の例は、デバイス上のすべてのサイトを取得し、固有の `SITE_TYPE` を返します。

```
set sites [get_sites]
set type {}
foreach x $sites {
    set prop [get_property SITE_TYPE $x]
    if { [lsearch -exact $type $prop] == -1 } {
        lappend type $prop
    }
}
foreach y $type {
    puts "SITE_TYPE: $y"
}
```

次の例は、3 つのサイト範囲の指定方法を示します。

```
get_sites -range {SLICE_X0Y0 SLICE_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
get_sites -range SLICE_X0Y0 -range SLICE_X1Y1
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
get_sites -range {SLICE_X0Y0:SLICE_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
```

関連項目

- [get_cells](#)
- [get_drc_violations](#)
- [get_property](#)
- [list_property](#)
- [report_property](#)

get_slrs

SLR のリストを取得します。

構文

```
get_slrs [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
         [-verbose] [<patterns>]
```

戻り値

slr

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したデバイス、タイル、サイト、BEL、サイト ピン、BEL ピン、クロック領域、ノード、PIP、ピン、パッケージ ピン、IO バンク、セルの SLR を取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	SLR を検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

ターゲット デバイスの SLR (Super Logic Region) のリストを取得します。複数の SLR が含まれないデバイスでは、SLR0 が返されます。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-regexp (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および -filter オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「.」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_slrs` で返されたオブジェクトのリストに、SLR のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。SLR オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`NUM_CHANNELS`、`NUM_SLLS`、`NUM_TILES` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したデバイス、タイル、サイト、BEL、サイト ピン (`site_pin`)、BEL ピン (`bel_pin`)、クロック領域 (`clock_region`)、ノード、PIP、ピン、パッケージ ピン (`package_pin`)、I/O バンク、またはセルに関連付けられている SLR を取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン「`<pattern>`」と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): SLR を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、現在のデザインに含まれるすべての SLR のリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいて SLR を検索できます。

注記: 複数の検索パターンは「`{ }`」で囲み、1 つのエレメントとして指定します。

例

次の例では、ターゲット デバイス上の各 SLR を異なる色でハイライトしています。

```
foreach x [get_slrs] {  
    incr i  
    highlight_objects -color_index $i $x  
}
```

注記: 検索パターンに一致するセルがない場合は、警告メッセージが表示されます。

次の例では、現在のデバイスの SLR (Super Logic Region) 間にある SLL (Super Long Line) の数が返されます。

```
get_property NUM_SLLS [get_slrs SLR0]
```

関連項目

- [get_property](#)
- [list_property](#)
- [report_property](#)

get_speed_models

デバイスのスピード モデル (speed_model) のリストを取得します。

構文

```
get_speed_models [-of_objects <args>] [-regexp] [-nocase] [-patterns <arg>]  
                 [-filter <arg>] [-speed_pattern <arg>] [-quiet] [-verbose]
```

戻り値

speed_model オブジェクト

使用法

名前	説明
[-of_objects]	指定したノード、BEL、PIP、セルの speed_model を取得します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-patterns]	speed_model オブジェクトを検索するパターンを指定します。デフォルトは * です。
[-filter]	式を使用してリストをフィルター処理します。
[-speed_pattern]	パターンに完全に一致する 1 つのモデルを返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Object \(オブジェクト\)](#)、[XDC](#)

説明

現在のデザインの UltraScale アーキテクチャ デバイスのスピード モデルを取得します。

スピード ファイルは、ザイリンクスにより各デバイスおよびスピード グレードに対して提供されています。スピード ファイルにスピード モデルが含まれています。デバイスのノード、PIP、BEL などのエレメントにスピード モデルがあります。セットアップ、ホールド、伝搬遅延、ジッターなどに対するスピード モデルがあります。

スピード モデルには、BEL、サイト、配線リソースなどのデバイス リソースに関連付けられている遅延 (ns) に関する情報が含まれます。スピード モデルは、Vivado タイミング エンジンでターゲット パーツにおける現在のデザインの解析を実行する際に使用されます。

返されるオブジェクトは、スピード ファイルから直接抽出された、PIP やワイヤなどの物理リソースに関連付けられているスピード モデルです。これらには、キャパシタンス値および抵抗値、バッファ モデルが含まれることもあります。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

このコマンドを実行すると、指定のスピード モデル オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_speed_models` で返されたオブジェクトのリストに、オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。スピード モデル オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、TYPE、DELAY などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したノード、BEL、PIP、またはセル オブジェクトのスピード モデルを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、A6LUT のスピード モデルのプロパティをレポートしています。

```
report_property -all [lindex [get_speed_models -of \
[get_bels SLICE_X0Y0/A6LUT]] 0]
Property      Type      Read-only  Value
CLASS         string   true      speed_model
DELAY         double  true      0.043
FAST_MAX      double  true      0.035
FAST_MIN      double  true      0.028
IS_INSTANCE_SPECIFIC bool    true      1
NAME          string   true      bel_d_lut6_a1_o6
NAME_LOGICAL   string   true      bel_d_lut6_a1_o6
SLOW_MAX      double  true      0.043
SLOW_MIN      double  true      0.036
SPEED_INDEX   int      true      65535
TYPE          string   true      bel_delay
```

次の例では、デバイスの特定の A6LUT を介する遅延 (ns) と、指定のオブジェクトの遅延名を返します。

```
set x [get_speed_models -of [get_bels SLICE_X0Y0/A6LUT]]
puts [format "%6.3f : %s" [get_property DELAY [lindex $x 0 ]] \
[get_property NAME [lindex $x 0 ]]]
```

関連項目

- [get_bels](#)
- [list_property](#)
- [report_property](#)

get_stacks

デザインのサブプログラム内で待機中のプロセスのリストを取得します。

構文

```
get_stacks [-of_instance <arg>] [-quiet] [-verbose]
```

戻り値

サブプログラム内で待機中のプロセスである HDL スコープ オブジェクト (指定したオプションから)

使用法

名前	説明
<code>[-of_instance]</code>	デフォルトは NULL です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

デザインのサブプログラム内で待機中のプロセスすべての HDL スコープを返します。-of_instance オプションを使用すると、出力を指定したインスタンスのプロセスに制限できます。

引数

-of_instance <HDL_Instance_scope> (オプション): サブプログラム内で待機中のプロセスで、指定した HDL インスタンスのもののみを取得します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例は、ターゲット デバイスのサイトすべてのリストを取得します。

```
module top;

    int i;

    function void f(input int in1);
        automatic int a;
        a = in1 + 7;
        $display($time, " in f :: a %d in1 %d ", a, in1);
    endfunction

    task automatic t(input int in2);
        int b;
        b = in2 + 10;
        $display($time, " in t :: in2 %d b %d ", in2, b);
        #5;
        f(b);        // Case C
        $display($time, " Back in t : after wait and f(%d) ", b);
    endtask

    initial begin                                // "/top/Initial18_0"
        $display($time, " in initial 1 ");
        i = 200;
        t(i);        // Case B
        $display($time, " Back in initial 1 after t(%d) ", i);
    end

    initial begin                                // "/top/Initial25_1"
        $display($time, " in initial 2 ");
        #2;
        f(50);        // Case A
        $display($time, " Back in initial 2 after f(50) ");
    end
end
endmodule
```

シミュレーションが Case A の呼び出しのために関数 f 内で停止すると、2つのプロセス /top/Initial18_0 および /top/Initial25_1 はそれぞれタスク t (Case B の呼び出し) および関数 f (Case A の呼び出し) 内で待機します。

```
1. > get_stacks
    /top/Initial18_0 /top/Initial25_1
```

関連項目

- [report_stacks](#)

get_template_bd_designs

IP インテグレーター サンプル デザインのリストを取得します。

構文

```
get_template_bd_designs [-quiet] [-verbose]
```

戻り値

IP インテグレーター デザイン オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado Design Suite の現在のリリースで使用可能なテンプレート ブロック デザインのリストを返します。正常に実行されなかった場合はエラーを返します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のリリースに含まれるブロック デザイン テンプレートのリストを取得しています。

```
get_template_bd_designs
```

get_tiles

タイルのリストを取得します。

構文

```
get_tiles [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>] [-quiet]
          [-verbose] [<patterns>]
```

戻り値

タイル

使用法

名前	説明
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-of_objects]</code>	指定した SLR、サイト、BEL、site_pin、bel_pin、ノード、ワイヤ、PIP、ネット、clock_region のタイルを取得します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<patterns>]</code>	タイルを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

開いているデザインのデバイス上にあるタイルのリストを返します。デフォルトでは、デバイス上のタイルすべてのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_tiles` で返されたオブジェクトのリストに、タイル オブジェクトのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。タイル オブジェクトの場合、結果をフィルター処理できるプロパティには `NUM_ARCS`、`NUM_SITES`、`IS_GT_SITE_TILE` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (`bool`) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したサイト、BEL、サイト ピン (`site_pin`)、BEL ピン (`bel_pin`)、ノード、ワイヤ、PIP、ネット、クロック領域 (`clock_region`)、または SLR に関連付けられているタイルを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<patterns> (オプション): タイルを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、デバイス上のタイルすべてのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてタイルを検索できます。

注記: 複数の検索パターンは波かっこ「`{ }`」またはダブルクォーテーション「`''`」で囲み、1 つのエレメントとして指定します。

例

次の例では、タイミング アークの数がそれぞれ 100 および 150 を超えるタイルの数が返されます。

```
llength [get_tiles -filter {NUM_ARCS>100} ]  
13468  
  
llength [get_tiles -filter {NUM_ARCS>150} ]  
11691
```

関連項目

- [get_bels](#)
- [get_nodes](#)
- [get_pips](#)
- [get_site_pins](#)
- [get_sites](#)
- [get_wires](#)
- [list_property](#)
- [report_property](#)

get_timing_arcs

タイミング アークのリストを取得します。

構文

```
get_timing_arcs [-from <args>] [-to <args>] [-filter <arg>]
                [-of_objects <args>] [-quiet] [-verbose]
```

戻り値

タイミング アーク オブジェクトのリスト

使用法

名前	説明
[-from]	タイミング アークの始点を指定します。
[-to]	タイミング アークの終点を指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したのセルのタイミング アークを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

XDC、Object (オブジェクト)、Timing (タイミング)

説明

指定したオブジェクトのタイミング アークのリストを取得します。指定のプロパティに基づいて、タイミング アークをフィルター処理できます。

タイミング アークはタイミング パスの一部で、2つのピン間のワイヤであったり、入力ピンと出力ピンの間にあるロジック インスタンスの内部パスであったりします。

注記: メモリおよびパフォーマンスを向上するため、get_* コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。lappend を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

-from <args> (オプション): タイミング アークの始点を指定します。ポート、ピン、またはネットを始点として指定できます。

-to <args> (オプション): タイミング アークの終点を指定します。ポート、ピン、またはネットを終点として指定できます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_timing_arcs` で返されたオブジェクトのリストに、タイミング アークのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。タイミング アーク オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`FROM_PIN`、`TO_PIN`、`LIB_CELL` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したセル オブジェクトからのタイミング アークを取得します。セルを指定すると、そのセルのすべてのセル アークが返されます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したバッファの出力ピンからのタイミング アークが返されます。

```
report_property -all [get_timing_arcs -of_objects [get_cells go_IBUF_inst]]
```

次の例では、指定したセルのタイミング アークが返されます。

```
get_timing_arcs -of_objects [get_cells count_reg[6]]
{count_reg[6]/C --> count_reg[6]/Q [Reg Clk to Q] }
{count_reg[6]/C --> count_reg[6]/D [setup] }
{count_reg[6]/C --> count_reg[6]/D [hold] }
{count_reg[6]/C --> count_reg[6]/CLR [recovery] }
{count_reg[6]/C --> count_reg[6]/CE [hold] }
{count_reg[6]/C --> count_reg[6]/CLR [removal] }
{count_reg[6]/C --> count_reg[6]/CE [setup] }
{count_reg[6]/CLR --> count_reg[6]/Q [Reg Set/Clr] }
```

関連項目

- [report_timing](#)

get_timing_paths

タイミング パスを取得します。

構文

```
get_timing_paths [-from <args>] [-rise_from <args>] [-fall_from <args>]
[-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
[-rise_through <args>] [-fall_through <args>] [-delay_type <arg>]
[-setup] [-hold] [-max_paths <arg>] [-nworst <arg>] [-unique_pins]
[-slack_lesser_than <arg>] [-slack_greater_than <arg>] [-group <args>]
[-no_report_unconstrained] [-user_ignored] [-routable_nets]
[-sort_by <arg>] [-filter <arg>] [-regexp] [-nocase] [-cell <args>]
[-quiet] [-verbose]
```

使用法

名前	説明
[-from]	タイミング パスの始点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_from]	ピン、ポート、セル、またはクロックの立ち上がりエッジを始点として指定します。
[-fall_from]	ピン、ポート、セル、またはクロックの立ち下がりエッジを始点として指定します。
[-to]	タイミング パスの終点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_to]	ピン、ポート、セル、またはクロックの立ち上がりエッジを終点として指定します。
[-fall_to]	ピン、ポート、セル、またはクロックの立ち下がりエッジを終点として指定します。
[-through]	タイミング パスの通過点 (ピン、ポート、セル、またはネット) を指定します。
[-rise_through]	ピン、ポート、セル、またはネットの立ち上がりエッジを通過点として指定します。
[-fall_through]	ピン、ポート、セル、またはネットの立ち下がりエッジを通過点として指定します。
[-delay_type]	パス遅延のタイプを指定します。有効な値は max、min、min_max、max_rise、max_fall、min_rise、min_fall で、デフォルトは max です。
[-setup]	最大遅延タイミング パスを取得します (-delay_type max と同じ)。
[-hold]	最小遅延タイミング パスを取得します (-delay_type min と同じ)。
[-max_paths]	出力するパスの最大数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
[-nworst]	終点までのワーストパスを N 個リストします。1 以上の値を指定します。デフォルト値は 1 です。
[-unique_pins]	ピンの各セットに対して、パス グループごとに 1 つのパスを表示します。
[-slack_lesser_than]	この値よりも小さいスラックのパスを含めます。デフォルトは 1e+30 です。

名前	説明
<code>[-slack_greater_than]</code>	この値よりも大きいスラックのパスを含めます。デフォルトは $-1e+30$ です。
<code>[-group]</code>	指定のグループのパスのみを返します。
<code>[-no_report_unconstrained]</code>	制約が適用されていないパスは取得しません。
<code>[-user_ignored]</code>	<code>set_false_path</code> または <code>set_clock_groups</code> タイミング制約のためにスラックが無限のパスのみをレポートします。
<code>[-routable_nets]</code>	配線可能なネット数をタイミング パスのプロパティとして保存します。
<code>[-sort_by]</code>	パスの並べ替え順を指定します。有効な値は <code>group</code> 、 <code>slack</code> で、デフォルトは <code>slack</code> です。
<code>[-filter]</code>	式を使用してリストをフィルター処理します。
<code>[-regexp]</code>	検索パターンを正規表現で指定します。
<code>[-nocase]</code>	検索パターンの大文字/小文字を区別せずに検索します (<code>-regexp</code> を指定した場合のみ有効)。
<code>[-cell]</code>	<code>get_timing_paths</code> を指定のセルに対して実行します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Object (オブジェクト)、Timing (タイミング)

説明

指定の条件を満たすタイミング パス オブジェクトを取得します。 `report_timing` に渡すタイミング パスを定義するためなどに使用できます。また、このコマンドを使用してカスタム レポートおよび解析を作成できます。

`get_timing_paths` コマンドは `report_timing` コマンドと似ていますが、`get_timing_paths` でがファイルまたは文字列を返すのに対し、`report_timing` はタイミング パス オブジェクトを返すので、そのプロパティを取得したり、ほかの Tcl コマンドに渡したりできます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは 1 つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナー リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-from <args>` (オプション): 解析するタイミング パスの始点を指定します。ポート、ピン、またはセルをタイミング パスの始点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが始点として指定されます。

`-rise_from <args>` (オプション): `-from` オプションと同様ですが、指定した始点からの信号の立ち上がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジで駆動されるパスのみが始点として考慮されます。

`-fall_from <args>` (オプション): `-from` オプションと同様ですが、指定した始点からの信号の立ち下がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジで駆動されるパスのみが始点として考慮されます。

`-to <args>` (オプション): 解析するタイミング パスの終点 (デスティネーション オブジェクト) を指定します。ポート、ピン、またはセル オブジェクトを終点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが終点として指定されます。

`-rise_to <args>` (オプション): `-to` オプションと同様ですが、指定した終点に到達する信号の立ち上がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジでキャプチャされるパスのみが終点として考慮されます。

`-fall_to <args>` (オプション): `-to` オプションと同様ですが、指定した終点に到達する信号の立ち下がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジでキャプチャされるパスのみが終点として考慮されます。

`-through <args>` (オプション): 指定したピン、セル インスタンス、またはネットを通過するパスのみをタイミング解析で考慮します。個別の `-through` (または `-rise_through` および `-fall_through`) 点を順次指定し、デザインを通過する特定のパスを定義できます。特定のパスを定義するには、通過点の指定順序が重要です。通過点を複数のオブジェクトで指定することもできます。この場合、指定の通過オブジェクトのいずれかを通過するタイミング パスが考慮されます。

`-rise_through <args>` (オプション): `-through` オプションと同様ですが、指定したオブジェクトで立ち上がるパスのみでタイミング解析を実行します。

`-fall_through <args>` (オプション): `-through` オプションと同様ですが、指定したオブジェクトで立ち下がるパスのみでタイミング解析を実行します。

`-delay_type <arg>` (オプション): タイミング レポートを実行する際に解析に使用する遅延のタイプを指定します。有効な値は `min`、`max`、`min_max`、`max_rise`、`max_fall`、`min_rise`、`min_fall` です。`-delay_type` オプションのデフォルト値は `max` です。

`-setup` (オプション): セットアップ違反がないかどうかをチェックします。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): ホールド違反がないかどうかをチェックします。これは `-delay_type min` を指定するのと同じです。

注記: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` を指定するのと同じになります。

`-max_paths <arg>` (オプション): `-sort_by` での指定の応じて、スラック順に並べた場合に出力するパスの最大数、またはグループごとに並べた場合にパス グループごとに出力するパスの最大数を指定します。1 以上の値を指定します。デフォルトは 1 で、ワースト タイミング パス 1 つ、またはグループごとにワースト パス 1 つがレポートされます。

`-nworst <arg>` (オプション): 各終点に対して表示するタイミング パス数を指定します。タイミング レポートには、指定した数のワースト パスが表示されます。1 以上の値を指定します。デフォルト値は 1 です。

`-unique_pins` (オプション): 各ピンまたはピンのグループごとに 1 つのタイミング パスのみを表示します。これはブール値のオプションであり、使用するとイネーブルになります。

`-slack_greater_than <arg>` (オプション): 算出されたスラック値が指定した値より大きいパスのタイミングをレポートします。`-slack_less_than` と共に使用すると、スラック値の範囲を指定できます。

`-slack_less_than <arg>` (オプション): 算出されたスラック値が指定した値より小さいパスのタイミングをレポートします。`-slack_greater_than` と共に使用すると、スラック値の範囲を指定できます。

`-group <args>` (オプション): 指定したパス グループのパスのタイミングをレポートします。

`-no_report_unconstrained` (オプション): 制約が適用されていないパスのタイミングはレポートしません。

`-user_ignored` (オプション): `set_false_path` または `set_clock_groups` タイミング制約が設定されているためタイミング解析で無視されたタイミング パスを表示します。これはブール値のオプションであり、使用するとイネーブルになります。

`-sort_by [slack | group]` (オプション): タイミング パスをパス グループごとまたはスラック値順にリストします。デフォルトではスラック順にリストされます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_timing_paths` で返されたオブジェクトのリストに、タイミング パスのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。タイミング パス オブジェクトのリストをフィルター処理するのに使用できるプロパティには、`DATAPATH_DELAY`、`ENDPOINT_PIN`、`ENDPOINT_CLOCK` などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`""`」のプロパティに一致します。

次の例では、100 個のクリティカル パス オブジェクトを取得し、パス グループ `clk_tx_clk_core_1` のもののみを取得しています。

```
get_timing_paths -max_paths 100 -filter {GROUP == clk_tx_clk_core_1}
```

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND「`&&`」および OR「`||`」で組み合わせることもできます。次の例では、名前に文字列「`RESET`」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン「`<patterns>`」および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-cell <arg>` (オプション): `get_timing_paths` コマンドを実行するセルを指定します。セルは、名前で指定するか、または `get_cells` コマンドを使用してオブジェクトとして指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定した終点からのワースト タイミング パスを 5 つを取得し、リストの 4 つ目のタイミング パスのプロパティをすべてレポートしています。

```
report_property -all [lindex [get_timing_paths -to [get_ports led_pins[*]]
\
-nworst 5] 3]
```

次の例では、`custom_report` というプロシージャを定義し、そのプロシージャを使用して `clk_tx_clk_core_1` パス グループの 100 個のワースト パスをレポートしています。

```
proc custom_report { listOfPaths } {
    puts [format {%40s %-40s %-20s %-20s %7s} "Startpoint" "Endpoint" \
        "Launch Clock" "Capture Clock" "Slack"]
    puts [string repeat "-" 140]
    foreach path $listOfPaths {
        set startpoint [get_property STARTPOINT_PIN $path]
        set startclock [get_property STARTPOINT_CLOCK $path]
        set endpoint [get_property ENDPOINT_PIN $path]
        set endclock [get_property ENDPOINT_CLOCK $path]
        set slack [get_property SLACK $path]
        puts [format {%40s %-40s %-20s %-20s %7s} $startpoint $endpoint \
            $startclock $endclock $slack]
    }
}
set paths [get_timing_paths -group clk_tx_clk_core_1 -max_paths 100]\
custom_report $paths
```

次の例では、タイミング パス オブジェクトを `report_timing` コマンドで使用方法を示しています。

```
set paths [get_timing_paths -group clk_tx_clk_core_1 -max_paths 100]
report_timing -of_objects $paths
```

これは、次のコマンドと同等です。

```
report_timing -group clk_tx_clk_core_1 -max_paths 100
```

次の例では、ロジック レベル数が指定のロジック レベル数よりも多いタイミング パスを取得しています。

```
get_timing_paths -max_paths 1000 -filter {LOGIC_LEVELS > 1}
```

関連項目

- [report_property](#)
- [report_timing](#)

get_value

選択した HDL オブジェクト (変数、信号、ワイヤ、レジスタ) の現在の値を取得します。

構文

```
get_value [-radix <arg>] [-quiet] [-verbose] <hdl_object>
```

戻り値

hdl_object の値

使用法

名前	説明
[-radix]	hdl_object の値の表示に使用する基数を指定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hdl_object>	現在の値を取得する hdl_object を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーション時間における 1 つの HDL オブジェクトの値を取得します。



ヒント: 複数の HDL オブジェクトの値を取得する場合は、`report_values` コマンドを使用してください。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event などがあります。

HDL 定数には、Verilog のパラメーターおよび localparam、VHDL ジェネリックおよび定数が含まれます。HDL スコープは、Verilog のモジュール、関数、タスク、プロセス、begin-end ブロックなど、HDL コードの宣言部分で定義されます。VHDL スコープには、エンティティ/アーキテクチャ定義、関数、プロシージャ、およびプロセス ブロックが含まれます。

引数

`-radix <arg>` (オプション): 指定したオブジェクトの値を表示するのに使用する基数を指定します。有効な値は default、dec、bin、oct、hex、unsigned、ascii、および smag です。

注記: dec は、符号付き 10 進数を示します。符号なしデータの場合は、unsigned を指定してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hdl_object> (必須): 値を取得する 1 つの HDL オブジェクトの名前を指定します。オブジェクトは、名前または `get_objects` コマンドで指定します。

例

次の例では、`sysClk` 信号の値を取得しています。

```
get_value sysClk
Z
```

次の例では、指定したバスから返された値の基数 `bin`、`dec`、および `unsigned` による違いを示しています。

```
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
10100101
get_value -radix unsigned /test/bench_VStatus_pad_0_i[7:0]
165
get_value -radix dec /test/bench_VStatus_pad_0_i[7:0]
-91
```

関連項目

- [current_time](#)
- [get_objects](#)
- [set_value](#)
- [report_values](#)

get_waivers

1 つまたは複数の DRC、METHODOLOGY、CDC メッセージ除外を取得します。

構文

```
get_waivers [-type <arg>] [-id <arg>] [-of_objects <args>] [-regexp]  
            [-filter <arg>] [-nocase] [-quiet] [-verbose] [<patterns>]
```

戻り値

除外オブジェクト

使用法

名前	説明
[-type]	除外のタイプを指定します。有効な値は、DRC、METHODOLOGY、CDC、ALL です。
[-id]	除外されている DRC、METHODOLOGY、CDC の ID を指定します。
[-of_objects]	DRC、METHODOLOGY、CDC 除外が設定されているオブジェクト (セル、ネット、ピン、サイトなど) を 1 つまたは複数指定します。
[-regexp]	検索パターンを正規表現で指定します。
[-filter]	式を使用してリストをフィルター処理します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	除外を検索するパターンを指定します。デフォルトの検索パターンは、ワイルドカード (*) または -regexp を指定している場合は「.*」です。

カテゴリ

Waiver (除外)、Object (オブジェクト)

説明

create_waiver コマンドを使用すると、デザインで無視しても問題ない DRC、設計手法、CDC の違反またはルール チェックを除外オブジェクトとして指定し、デザイン フローを先に進めることができます。get_waivers コマンドを使用すると、現在のデザインで定義されている除外オブジェクトを取得できます。

除外は、個々の DRC または設計手法違反、特定の DRC または設計手法チェック、または CDC パスに指定する必要があります。特定のオブジェクト、特定の違反 ID、または -from/-to オプションを使用してパスを指定します。get_waivers コマンドでは、特定のタイプの除外または特定のオブジェクトに関連付けられている除外を取得できます。

現在のデザインで定義されている除外をレポートするには `report_waivers` コマンドを使用し、除外を削除するには `delete_waivers` コマンドを使用します。

引数

`-type <arg>` (オプション): 取得する除外をタイプを指定します。有効な値は DRC、METHODOLOGY、および CDC です。

`-id <arg>` (オプション): 除外に関連付けられているチェックまたは違反の ID を指定します。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_waivers` で返されたオブジェクトのリストに、除外のプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。除外オブジェクトのリストをフィルター処理するのに使用できるプロパティには、OBJECT_COUNTS、TYPE などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <arg>` (オプション): 指定したオブジェクト (セル、ネット、ピン、サイトなど) に関連付けられている除外を取得します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 除外を検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、プロジェクトに含まれるすべての除外のリストが取得されます。

例

次の例では、現在のデザインで定義されているすべての除外をレポートしています。

```
get_waivers
```

次の例では、すべての DRC チェック除外を取得しています。

```
get_waivers -type DRC *
```

次の例では、指定したオブジェクトに関連付けられているすべての除外を取得しています。

```
get_waivers -of_objects [get_ports {src_in* dest_out*}]
```

関連項目

- [create_waiver](#)
- [delete_waivers](#)
- [report_waivers](#)

get_wave_configs

指定のオプションに一致する波形設定を取得します。

構文

```
get_wave_configs [-regexp] [-nocase] [-filter <arg>] [-quiet] [-verbose]
[<patterns>...]
```

戻り値

指定のオプションに一致する波形設定

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	結果のリストに指定した式 (<patterns>) のフィルターを適用します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	波形設定名を検索するパターン文字列を指定します。

カテゴリ

Waveform (波形)

説明

現在のシミュレーションで指定の検索オプションに一致する波形設定オブジェクトを取得します。

Vivado® シミュレータ GUI では、波形ウィンドウを使用してデザインを解析し、コードをデバッグできます。波形設定ファイルには、表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。波形設定には最上位 HDL オブジェクトが表示されますが、`add_wave` および `add_wave_divider` などのコマンドを使用してオブジェクトを追加できます。

このコマンドを実行すると、一致する波形設定オブジェクトが返されるか、検索パターンに一致するオブジェクトがない場合は何も返されません。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。ザイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_wave_configs` で返されたオブジェクトのリストに、波形設定オブジェクトのプロパティ値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。波形設定オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、NEEDS_SAVE、FILE_PATH などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`*`」を追加して、プロパティ値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ値に一致すると、そのオブジェクトが返されます。ワイルドカード「`*`」を使用すると、定義値が「`''`」のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価「`==`」、不等価「`!=`」、一致「`=~`」、不一致「`!~`」です。数値比較演算子「`<`」、「`>`」、「`<=`」、および「`>=`」も使用できます。複数のフィルター式を AND 「`&&`」および OR 「`||`」で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

ブール型 (boolean) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): 現在のシミュレーションで波形設定オブジェクトを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード「`*`」で、シミュレーションで開いている波形設定すべてのリストが取得されます。

例

次の例では、現在のシミュレーションで変更が保存されていない波形設定オブジェクトを取得しています。

```
get_wave_config -filter {NEEDS_SAVE}
```

関連項目

- [close_wave_config](#)

- [create_wave_config](#)
- [current_wave_config](#)
- [open_wave_config](#)
- [save_wave_config](#)

get_waves

波形設定から波形オブジェクトを取得します。

構文

```
get_waves [-of <args>] [-regexp] [-nocase] [-filter <arg>] [-recursive]
          [-r] [-long_name] [-short_name] [-quiet] [-verbose] <patterns>...
```

戻り値

検索された波形オブジェクト

使用法

名前	説明
[-of]	検索する波形設定、グループ、または仮想バスを指定します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	結果のリストに指定した式 (<patterns>) のフィルターを適用します。
[-recursive]	デザイン オブジェクトがスコープの場合に、そのスコープ内のデザイン オブジェクトすべての波形オブジェクトを作成します。
[-r]	デザイン オブジェクトがスコープの場合に、そのスコープ内のデザイン オブジェクトすべての波形オブジェクトを作成します。
[-long_name]	長い名前を使用して波形オブジェクトを検索します。
[-short_name]	短い名前を使用して波形オブジェクトを検索します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<patterns>	波形オブジェクトを作成する基になるデザイン オブジェクトを指定します。

カテゴリ

Waveform (波形)

get_wires

ワイヤのリストを取得します。

構文

```
get_wires [-regexp] [-nocase] [-filter <arg>] [-of_objects <args>]
          [-uphill] [-downhill] [-from <args>] [-to <args>] [-quiet] [-verbose]
          [<patterns>]
```

戻り値

ワイヤ

使用法

名前	説明
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
[-of_objects]	指定したタイル、ノード、PIP、またはネットのワイヤを取得します。
[-uphill]	指定した PIP からドライバー側のワイヤを取得します。
[-downhill]	指定した PIP からロード側のワイヤを取得します。
[-from]	指定した PIP またはサイト ピンから開始するワイヤの順序付きリストを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-to]	指定したワイヤまたはサイト ピンで終了するワイヤの順序付きリストを取得します。-uphill と共に使用できます。デフォルトでは -downhill が使用されます。-all が自動的に使用されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	ワイヤを検索するパターンを指定します。デフォルトは * です。

カテゴリ

Object (オブジェクト)、XDC

説明

開いているデザインに含まれるワイヤで、指定した検索パターンに一致するものをリストします。

デフォルトでは、デザインに含まれるすべてのワイヤのリストが取得されます。

注記: メモリおよびパフォーマンスを向上するため、`get_*` コマンドでは1つのタイプのオブジェクト (セル、ネット、ピン、ポートなど) のコンテナ リストが返されます。`lappend` を使用するなどしてリストにオブジェクトを追加できますが、現在リストに含まれるオブジェクトと同じタイプのオブジェクトしか追加できません。リストに異なるタイプのオブジェクトや文字列を追加しようとすると、Tcl エラーが返されます。

引数

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (<patterns>) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、`get_wires` で返されたオブジェクトのリストに、ワイヤのプロパティ 値に基づいてフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。フィルターには、任意のプロパティと値の組み合わせを使用できます。ワイヤ オブジェクトのリストをフィルター処理するのに使用できるプロパティには、NAME、NUM_DOWNHILL_PIPS、NUM Uphill_PIPS などがあります。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.*`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!=) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ ".*RESET.*"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-of_objects <args>` (オプション): 指定したノード、PIP、またはタイルのワイヤを取得します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (<pattern>) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ワイヤを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、デザインに含まれるすべてのワイヤのリストが取得されます。複数の検索パターンを指定して、異なる検索条件に基づいてワイヤを検索できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定のタイルに関連するワイヤが返されます。

```
get_wires -of_objects [get_tiles IO_INT_INTERFACE_L_X0Y198]
```

関連項目

- [get_nodes](#)
- [get_pips](#)
- [get_tiles](#)
- [list_property](#)
- [report_property](#)

group_bd_cells

階層セルを作成し、セルのグループをその階層セルに移動します。セル間の接続は保持されます。これらのセルとその他のセルの接続は、階層セルをまたぐことにより保持されます。

構文

```
group_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<target_cell_name>]  
[<cells>...]
```

戻り値

正しく実行された場合は 0。

使用法

名前	説明
[-prefix]	セルに追加する接頭辞を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<target_cell_name>]	ターゲット セルを指定します。
[<cells>]	セル名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP インテグレーター サブシステム デザインに新しい階層モジュールを作成し、指定したセルをそのモジュール内に移動します。

指定したセルのグループを階層モジュールに移動することもできます。指定したセル間の接続は保持されます。移動するセル間の接続は保持されます。移動するセルとその他の移動されないセルの間の接続は、IP インテグレーターで階層の境界をまたぐためにピンおよびポートを追加することにより自動的に保持されます。

階層モジュールを `create_bd_cells` コマンドを使用して作成し、`move_bd_cells` コマンドを使用してその階層モジュールにセルを移動することも可能です。

このコマンドを実行すると、作成された階層モジュールの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-prefix <arg>` (オプション): 階層モジュールに移動するセルに適用する接頭辞を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<target_cell_name>` (必須): 作成する階層モジュールの名前を指定します。

`<cells>` (必須): 現在の IP サブシステム デザインから階層モジュールに移動するセルを指定します。セルの指定には、`get_bd_cells` コマンドを使用します。

例

次の例では、現在の IP インテグレーター サブシステム デザインに新しい階層ブロックを作成し、指定した 3 つのセルに接頭辞を指定してブロック内に移動しています。

```
group_bd_cells -prefix M1_ module1 [get_bd_cells /microblaze_1_xlconcat] \
[get_bd_cells /microblaze_1_axi-intc] [get_bd_cells /proc-sys-reset-1]
```

関連項目

- [get_bd_cells](#)
- [move_bd_cells](#)

group_path

コスト ファンクション計算用にパスをグループ化します。

構文

```
group_path [-name <args>] [-weight <arg>] [-default] [-from <args>]
           [-to <args>] [-through <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-name]	グループの名前を指定します。複数の名前を指定できます。
[-weight]	コスト関数の重みを指定します。有効な値は 1、2 で、デフォルトは 1 です。
[-default]	パスをデフォルトのグループに戻します。
[-from]	指定した始点から開始するパスを含めます。
[-to]	指定した終点で終了するパスを含めます。
[-through]	ピン、セル、またはネットを通過するパスを含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC

説明

このコマンドは、主にタイミング解析で使用するコスト ファンクション計算用にパスをグループ化します。Vivado ツールでは、特別な処理用にクロック信号のパス グループが自動的に定義されます。ユーザー定義パスは、始点、終点、または通過点を指定してグループ化できます。パス グループを作成すると、`report_timing` コマンドを使用してそのパス グループに対してタイミング解析を実行できます。

クロックのパス グループに重みを指定し、配置、配線、および最適化でそれらのパスが最初に処理されるようにすることができます。

パス グループからパスを削除するには、`-default` オプションを使用してパス グループからパスを削除して標準のデフォルト パス グループに戻す必要があります。

このオプションには、次の制限があります。

- ツールでクロック パス グループに割り当てられているパスは、`-default` オプションでデフォルト パス グループに戻されるのではなく、クロック パス グループに戻されます。
- `group_path -default` と `reset_path` は、完全に独立したコマンドです。`reset_path` コマンドはパス グループには影響せず、`group_path` コマンドはタイミング例外には影響しません。

デザインで現在定義されているパス グループは、`get_path_groups` コマンドを使用して確認できます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-name <arg>` (オプション): パス グループの名前を指定します。指定したパス グループ名が既に存在する場合は、指定したパスが既存のグループに追加されます。

`-weight [1 | 2]` (オプション): クロック パスの優先度を指定します。1 は標準優先度、2 は高優先度です。高優先順位のパス グループは、配置、配線、物理最適化で先に処理されます。デフォルトは 1 (標準優先度) です。



重要: `-weight` オプションは、クロック パス グループでのみサポートされ、`-name` オプションを使用してパス グループを指定する必要があります。ユーザー定義パス グループでの使用、`-from`、`-through`、`-to` オプションとの使用はサポートされていません。

`-default` (オプション): パス グループを、ほかのグループに割り当てられていないデフォルトのパス グループに戻します。このオプションは、`-name` または `-weight` オプションと共に指定することはできません。パスは、`-from`、`-through`、`-to` オプションを使用して、最初にパス グループに割り当てたときに定義したように指定する必要があります。

`-from <args>` (オプション): 指定した始点から開始するパスを含めます。始点はピン、ポート、またはクロックで指定できます。

`-through <element_names>` (オプション): 指定したピン、セル、またはネットを通過するパスを含めます。

`-to <path_names>` (オプション): 指定した終点で終了するパスを含めます。終点はピン、ポート、またはクロックで指定できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、「*signal*reg/D」に一致するレジスタが終点となるパスを含む `signal_grp` という名前のパス グループを作成し、そのグループのタイミングをレポートしています。

```
group_path -to [get_pins *signal*reg/D -hierarchical] -name signal_grp
report_timing -group signal_grp
```

パス グループ `signal_grp` は、`get_path_groups` コマンドでも返されます。

```
get_path_groups signal_grp
```

次の例では、`signal_grp` からパスを削除し、デフォルトのパス グループに戻しています。

```
group_path -to [get_pins *signal*reg/D -hierarchical] -default
```

関連項目

- [get_path_groups](#)

- [report_timing](#)

help

1 つまたは複数の項目に対するヘルプを表示します。

構文

```
help [-category <arg>] [-args] [-syntax] [-long] [-prop <arg>]
      [-class <arg>] [-message <arg>] [-quiet] [-verbose]
      [<pattern_or_object>]
```

使用法

名前	説明
[-category]	指定のカテゴリでトピックを検索します。
[-args]	引数の説明を表示します。
[-syntax]	構文の説明を表示します。
[-long]	詳細な説明を表示します。
[-prop]	指定したプロパティのヘルプを表示します。デフォルトは * です。
[-class]	指定したオブジェクトのクラスのプロパティに関する情報を表示します。
[-message]	指定のメッセージに関する情報を表示します。ツールで表示される各メッセージには、アプリケーション サブシステムのコードとメッセージ識別子を含む固有のメッセージ ID が付けられています。例: -message {Common 17-8}
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<pattern_or_object>]	情報を表示するコマンド、またはリストするコマンドを検索するパターンを指定します。デフォルトは *、です。

カテゴリ

Project (プロジェクト)

説明

指定した Tcl コマンドの説明、サイリンクス Tcl コマンドのカテゴリのリスト、または指定した検索パターンに一致するコマンドのリストを返します。

引数を指定せずに `help` を使用すると、Tcl コマンドのカテゴリのリストが返されます。コマンド カテゴリは、File I/O などの特定のファンクションを実行するコマンド グループです。

`help` コマンドで使用するオプションによって、コマンド構文のみ、コマンド構文と各引数の簡単な説明、またはコマンドの詳細な説明と使用例を表示できます。

Vivado Design Suite のメモリ使用量を制限するため、ツールの一部の機能はその機能が使用されるときにのみメモリに読み込まれます。ある機能に関連するすべての Tcl コマンドのリストとヘルプ テキストを表示するには、`load_features` コマンドを使用してその機能をメモリに読み込む必要があります。

help コマンドでは、デザイン オブジェクトに設定可能なさまざまなプロパティに関する情報も表示できます。プロパティに関する情報を表示するには、`-prop` および `-class` オプションを使用してください。

このコマンドを実行すると、指定のヘルプ テキストまたはエラーが返されます。

引数

`-category <arg>` (オプション): 指定したカテゴリのコマンドのリストを取得します。

`-args` (オプション): 指定したコマンドの簡単な説明を取得します。デフォルトでは、指定したコマンドの詳細な説明が返されます。簡単な説明を表示する場合にこのオプションを使用します。

`-syntax` (オプション): コマンドの構文のみを返します。

`-long` (オプション): コマンドの構文、引数の簡単な説明、およびコマンドの詳細な説明と使用例を返します。これがデフォルトです。

`-prop <arg>` (オプション): オブジェクト クラスの指定のプロパティ、または現在のデザインに含まれる特定のオブジェクトに割り当てられているプロパティを返します。

注記: このオプションを使用する場合は、`-class` も使用するか、1 つのデザイン オブジェクトを指定する必要があります。

`-class <arg>` (オプション): 指定したオブジェクトのクラスに関する情報を返します。

`-message <arg>` (オプション): 指定したメッセージに関する情報を返します。メッセージには、「Common 17-24」や「{Common 17-24}」などのアプリケーション サブシステムのコードとメッセージ識別子を含む固有のメッセージ ID が付けられています。詳細は、`set_msg_config` コマンドを参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pattern_or_object>` (オプション): 情報を表示するコマンド、またはリストするコマンドを検索するパターンを指定します。

注記: `-class` および `-prop` を使用してプロパティの情報を返す場合は、セルやサイトなどの Vivado ファースト クラス オブジェクトを指定する必要があります。

例

次の例では、ザイリンクス Tcl コマンドのカテゴリのリストが返されます。

```
help
```

次の例では、Vivado Design Suite のシミュレータ機能を読み込み、Simulation および Waveform カテゴリの Tcl コマンドを取得しています。

```
load_features simulator
help -category simulation
help -category waveform
```

次の例では、指定した検索パターンに一致するコマンドのリストが返されます。

```
help *file*
```

このリストは、`remove_files` や `delete_files` などの特定の機能のコマンドを検索するのに使用できます。

次の例では、`remove_files` コマンドとその引数の詳細な説明が返されます。

```
help remove_files
```

注記: `-args` オプションを使用すると、そのコマンドの簡単な説明を表示できます。

次の例では、`short` というプロシージャを定義し、`-args` を使用して指定したコマンドの簡単な説明を取得しています。

```
proc short cmdName {help -args $cmdName}
```

注記: このプロシージャを `init.tcl` ファイルに追加すると、ツールを起動するたびにこのコマンドを読み込むことができます。`init.tcl` ファイルの詳細は、第 1 章「概要」を参照してください。

次の例では、デザイン オブジェクトまたはデザイン オブジェクトのクラスのプロパティに関する情報を表示する方法を示しています。

```
help -class cell -prop NAME
help -prop NAME [get_cells cpuEngine]
```

注記: この例では、最初のコマンドで `NAME` プロパティに関する一般的な情報を取得し、2 番目のコマンドで指定したデザイン オブジェクトの `NAME` プロパティの値を取得しています。

関連項目

- [get_cells](#)
- [list_features](#)
- [list_property](#)
- [load_features](#)
- [report_property](#)
- [set_msg_config](#)

highlight_objects

オブジェクトを指定の色でハイライトします。

構文

```
highlight_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
                  [-leaf_cells] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-color_index]</code>	色を色インデックスで指定します。
<code>[-rgb]</code>	色を RGB で指定します。
<code>[-color]</code>	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
<code>[-leaf_cells]</code>	最下位セルを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ハイライトするオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

指定したオブジェクトを色オプションのいずれかで指定した色でハイライトします。



ヒント: ハイライト色を指定する色オプションは、1 つのみ指定します。複数の色オプションを指定すると、`-rgb`、`-color_index`、`-color` の優先順位で色が決定されます。

オブジェクトを指定したハイライト色で表示するため、オブジェクトの選択は自動的に解除されます。オブジェクトのハイライトを解除するには、`unhighlight_objects` コマンドを使用します。

引数

`-color_index <arg>` (オプション): 有効な値は 1 ~ 20 の整数で、オブジェクトのハイライト色を色インデックスで指定します。色インデックスは、[Tools]→[Settings] をクリックして開く [Settings] ダイアログ ボックスの [Colors]→[Highlight] ページで表示および設定できます。色の設定方法の詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

`-rgb <args>` (オプション): オブジェクトをマークする色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定します。

`-color <arg>` (オプション): オブジェクトのハイライト色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

注記: 白は `select_objects` コマンドで指定したオブジェクトを表示するのに使用されます。

`-leaf_cells <arg>` (オプション): ハイライトするセルの名前またはセル オブジェクトを 1 つまたは複数指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): ハイライトするオブジェクトを指定します。

例

次の例は、現在選択されているオブジェクトを赤でハイライトします。

```
highlight_objects -color red [get_selected_objects]
```

次の例は、指定のセルを緑色でハイライトします。

```
highlight_objects -color green -leaf_cells [get_cells cpuEngine/*]
```

関連項目

- [get_highlighted_objects](#)
- [get_marked_objects](#)
- [get_selected_objects](#)
- [mark_objects](#)
- [select_objects](#)
- [unhighlight_objects](#)

implement_debug_core

デバッグ コアをインプリメントします。

構文

```
implement_debug_core [-quiet] [-verbose] [<cores>...]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<cores>]</code>	デバッグ コアを指定します。

カテゴリ

Debug (デバッグ)

説明

現在のプロジェクトに含まれる Vivado ロジック解析デバッグ コアをインプリメントします。ツールは指定した ILA デバッグ コアに対して 1 回実行され、すべてのコアが指定されると、デバッグ ハブ コアに対してもう 1 回実行されます。現在 `create_debug_core` コマンドでサポートされるコア タイプは、ILA コア (`labtools_ila_v3`) のみです。プロジェクトに ILA コアを含めてコンフィギュレーションするため、ツールによりデバッグ ハブ コア (`labtools_xsdbmasterlib_v2`) が自動的に追加されます。

デバッグ ハブ コアおよび ILA コアは、最初ブラック ボックスとして作成されます。これらのコアは、配置配線を実行する前にインプリメントしておく必要があります。`create_debug_core` でコアを作成した後、`create_debug_port` および `connect_debug_port` でポートを追加および接続しますが、デバッグ コアの内容はデザインでは定義されていません。

デバッグ コアは、`launch_runs` コマンドを使用してインプリメンテーション `run` を実行したとき、または `opt_design` コマンドで実行されるデザイン最適化中に自動的にインプリメントされますが、`implement_debug_core` コマンドを使用すると、デザイン全体をインプリメントせずに、デザインに含まれるコアをインプリメントできます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<cores> (オプション): インプリメントするデバッグ コアを 1 つ以上指定します。コアを指定しない場合、すべてのデバッグ コアがインプリメントされます。

例

次の例では、現在のプロジェクトに含まれるすべてのデバッグ コアをインプリメントします。

```
implement_debug_core [get_debug_cores]
```

関連項目

- [connect_debug_port](#)
- [create_debug_core](#)
- [create_debug_port](#)
- [get_debug_cores](#)
- [launch_runs](#)

implement_mig_cores

IP サービスを呼び出して IP を再生成し、現在のネットリストに統合します。

構文

```
implement_mig_cores [-outputdir <arg>] [-rtlonly] [-force] [-debug_output]
                    [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-outputdir]</code>	PHY IP の生成ファイルの保存先ディレクトリを指定します。
<code>[-rtlonly]</code>	PHY RTL コードを生成するプロセスをすべて実行しますが、PHY コアのネットリストは置き換えません。
<code>[-force]</code>	最適化されていないすべてのメモリ コアをインプリメントします。 <code>-rtlonly</code> オプションを使用している場合は、最適化されているコアも含まれます。
<code>[-debug_output]</code>	デバッグ出力をイネーブルにします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Memory (メモリ)

説明

現在のプロジェクトのメモリ IP をインプリメントします。

ザイリンクス® IP カタログに含まれるメモリ IP は、ザイリンクス デバイス用のメモリ コントローラーおよびインターフェイスを生成するために使用されます。メモリ IP には、指定されたデバイス アーキテクチャおよびメモリ インターフェイスに応じて、ザイリンクス IP カタログからの異なる IP コアが含まれます。使用可能なメモリ IP の詳細は、『Zynq-7000 SoC および 7 シリーズ デバイス メモリ インターフェイス ソリューション ユーザー ガイド』(UG586) または『LogiCORE IP UltraScale アーキテクチャ FPGA メモリ インターフェイス ソリューション製品ガイド』(PG150) を参照してください。

`implement_mig_cores` コマンドは、メモリ コントローラーの物理インターフェイス (PHY) の RTL 情報を生成し、メモリ コントローラーの合成済みネットリストを最上位デザインに統合します。

ザイリンクス IP カタログからデザインに追加する際に、メモリ コントローラーでデバッグをイネーブルにできます。Vivado ロジック解析または Vivado Lab Edition では、デザインにインプリメントされているメモリ コントローラーは、デバッグがイネーブルになっているメモリ コントローラーごとに 1 つの `hw_mig` オブジェクトに関連付けられます。`hw_mig` オブジェクトには、キャリブレーション ステータスを取得し、ビットごとのアイ マージン表示を描画するために必要なプロパティがすべて含まれます。

メモリ IP およびデバッグ コアは、`launch_runs` コマンドを使用してインプリメンテーション `run` を実行したとき、または `opt_design` コマンドで実行されるデザイン最適化中に自動的にインプリメントされますが、`implement_mig_cores` コマンドを使用すると、デザイン全体をインプリメントせずに、メモリ コントローラーをインプリメントして統合できます。



ヒント: `implement_mig_cores` コマンドを実行する前に、メモリ コントローラーのピンをすべて割り当てておく必要があります。そのようにしないと、エラーが返されます。 `report_drc` を使用して、メモリ コントローラーのステータスを確認できます。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-outputdir <arg>` (オプション): メモリ IP の生成される出力ファイルの出力ディレクトリを指定します。 `-outputdir` オプションを指定しない場合、出力は現在のプロジェクト フォルダーに保存されます。

`-rtlonly` (オプション): メモリ コントローラーの PHY RTL 情報のみを生成します。

`-force` (オプション): メモリ IP がアップデート必要な状態であっても、メモリ IP を強制的にインプリメントします。

`-debug_output` (オプション): メモリ IP のデバッグ機能をイネーブルにします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインに含まれるメモリ IP をインプリメントしています。

```
implement_mig_cores
```

関連項目

- [commit_hw_mig](#)
- [get_hw_migs](#)
- [launch_runs](#)
- [opt_design](#)
- [refresh_hw_mig](#)
- [report_hw_mig](#)

implement_xphy_cores

IP サービスを呼び出して IP を再生成し、現在のネットリストに統合します。

構文

```
implement_xphy_cores [-outputdir <arg>] [-rtlonly] [-force] [-debug_output]
                    [-quiet] [-verbose]
```

使用法

名前	説明
[-outputdir]	PHY IP の生成ファイルの保存先ディレクトリを指定します。
[-rtlonly]	PHY RTL コードを生成するプロセスをすべて実行しますが、PHY コアのネットリストは置き換えません。
[-force]	最適化されていないすべてのメモリ コアをインプリメントします。-rtlonly オプションを使用している場合は、最適化されているコアも含まれます。
[-debug_output]	デバッグ出力をイネーブルにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Memory (メモリ)

import_files

ファイルまたはディレクトリをアクティブなファイルセットにインポートします。

構文

```
import_files [-fileset <arg>] [-force] [-of_objects <args>] [-norecurse]
             [-flat] [-relative_to <arg>] [-quiet] [-verbose] [<files>...]
```

戻り値

インポートされたファイル オブジェクトのリスト

使用法

名前	説明
[-fileset]	ファイルセット名を指定します。
[-force]	プロジェクト ディレクトリの同じ名前のファイルを上書きします。
[-of_objects]	ファイルをインポートする RM を指定します。
[-norecurse]	下位ディレクトリの検索をディスエーブルにします。
[-flat]	ファイルをフラットなディレクトリ構造でインポートします。
[-relative_to]	指定したディレクトリに相対するパスでファイルをインポートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<files>]	ファイルセットにインポートするファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

1 つ以上のファイルまたは 1 つ以上のディレクトリ内のソース ファイルを指定したファイルセットにインポートします。

Vivado Design Suite では、プロジェクトに追加されたすべてのファイルに対して、そのファイルまたはディレクトリへの相対パスと絶対パスの両方が格納および管理されます。プロジェクトを開くと、これらのパスを使用してファイルおよびディレクトリが検索されます。デフォルトでは、パスを検索するのにまず相対パスが使用され、その後に絶対パスが使用されます。PATH_MODE プロパティを使用すると、Vivado ツールでの指定したオブジェクトのファイルパスまたはプロパティの処理方法を変更できます。詳細は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912)を参照してください。



重要: 複数のファイルを 1 つずつインポートすると、パフォーマンスが著しく低下します。1 つの `import_files` コマンドで複数のファイルをインポートの方が効率的です。

```
import_files {file1 file2 file3 ... fileN}
```

このコマンドは、指定したファイルセットを参照することによりファイルを追加する `add_files` コマンドとは異なり、`project.srcs\<fileset>\imports` の下のローカル プロジェクト フォルダにファイルをインポートしてから、そのファイルを指定したファイルセットに追加します。

引数

`-fileset <name>` (オプション): 指定したソース ファイルを追加するファイルセットを指定します。指定したファイルセットが存在しない場合は、エラー メッセージが表示されます。ファイルセットを指定しない場合は、デフォルトでソース ファイルセットに追加されます。

`-force` (オプション): ローカル プロジェクト ディレクトリおよびファイルセットの同じ名前のファイルを上書きします。

`-norecurse` (オプション): 指定したディレクトリの下位ディレクトリでコマンドを実行しないよう指定します。このオプションを指定しない場合、下位ディレクトリでもプロジェクトに追加可能なソース ファイルが検索されます。

`-flat` (オプション): 相対パスを保持せずに、すべてのファイルをインポート フォルダにインポートします。デフォルトでは、ファイルをデザインにインポートする際にディレクトリ構造が保持されます。

`-relative_to <arg>` (オプション): 指定したディレクトリに相対するパスでファイルをインポートします。これにより、ローカル プロジェクトのディレクトリ構造でインポート ファイルへのパスを保持できます。ファイルは、指定したディレクトリに相対的なパスを使用してインポート フォルダにインポートされます。

注記: `-relative_to` オプションは、`-flat` オプションを指定すると無視されます。`-flat` オプションを使用すると、インポート ファイルのディレクトリ構造が削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (オプション): 指定したファイルセットに追加する 1 つまたは複数のファイルまたはディレクトリ名を指定します。ディレクトリ名を指定した場合は、そのディレクトリとそれに含まれる下位ディレクトリの有効なソース ファイルすべてが追加されます。ファイルを指定しない場合は、現在のプロジェクトのソース セットのファイルがインポートされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、`top.ucf` ファイルを `constrs_1` 制約ファイルセットにインポートしています。

```
import_files -fileset constrs_1 top.ucf
```

次の例では、`-fileset` オプションが指定されていないので、有効なソース ファイルがソース ファイルセット (`sources_1`) にデフォルトでインポートされます。また、`-norecurse` オプションが指定されているので、指定した `\level1` ディレクトリのみが検索され、下位ディレクトリは検索されません。`-flat` オプションは指定されていないので、すべての有効なソース ファイルがプロジェクトの `\imports` フォルダーにインポートされます。

```
import_files C:/Data/FPGA_Design/level1 -norecurse -flat
```

注記: `-flat` オプションが指定されていないので、プロジェクトの `\imports` フォルダー内に `\level1` ディレクトリが作成されます。

次の例では、`-fileset` オプションが指定されていないので、ファイルがソース ファイルセット (`sources_1`) にインポートされます。有効なソース ファイルは `\level1` ディレクトリおよびすべての下位ディレクトリからインポートされ、`-relative_to` オプションが使用されているので、プロジェクトの `\imports` フォルダーに `\Data` ディレクトリから書き込まれます。

```
import_files C:/Data/FPGA_Design/level1 -relative_to C:/Data
```

関連項目

- [add_files](#)

import_ip

IP ファイルをインポートしてファイルセットに追加します。

構文

```
import_ip [-srcset <arg>] [-name <arg>] [-quiet] [-verbose] [<files>]
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
<code>[-srcset]</code>	アップグレードするオブジェクトを含むソース ファイル セットを指定します。デフォルトは、現在のソース ファイルセットです。
<code>[-name]</code>	インポートされた IP の新しい名前を指定します。複数のファイルを指定する場合は使用できません。デフォルトは、インポートされた IP の現在の名前です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	インポートする IP ファイルの名前を指定します。XCI または XCO ファイル名を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

既存の XCI または XCO ファイルを IP ソースとして現在のプロジェクトにインポートし、ローカル ディレクトリ構造にコピーします。

`import_ip` コマンドを使用すると、既存の IP ファイル直接読み込み、ローカル プロジェクト フォルダにコピーできます。IP ファイルを参照することにより現在のプロジェクトに追加するには、`read_ip` または `add_files` コマンドを使用します。

現在の IP カタログから新しい IP ファイルを作成するには、`create_ip` を使用します。

引数

`-srcset <arg>` (オプション): IP ファイルをインポートするソース ファイルセットを指定します。このオプションを指定しない場合のデフォルトのソース ファイルセットは `sources_1` です。

`-name <arg>` (オプション): 現在のソース ファイルセットに追加する IP オブジェクトに割り当てる名前を指定します。このオプションは、`<files>` で 1 つのファイルを指定している場合にのみ使用できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (オプション): 現在のプロジェクトにインポートする IP ファイルの名前を指定します。既存の XCI または XCO ファイルを指定する必要があります。XCI ファイルは、IP のパラメーター情報を含む IP-XACT 形式のファイルです。XCO ファイルは、IP コアを生成する際に使用されるカスタマイズ パラメーターとプロジェクト オプションがすべて記述された CORE Generator ファイルです。XCI または XCO ファイルは、現在のプロジェクトでコアを作成し直すために使用されます。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、10 ギガビット イーサネット コアを現在のプロジェクトにコピーし、`IP_block1` という名前を割り当てています。

```
import_ip C:/Data/FPGA_Design/10gig_eth.xci -name IP_block1
```

関連項目

- [add_files](#)
- [create_ip](#)
- [generate_target](#)
- [read_ip](#)

import_synplify

指定した Synplify プロジェクト ファイルをインポートします。

構文

```
import_synplify [-copy_sources] [-quiet] [-verbose] <file>
```

戻り値

Synplify ファイルからインポートされたファイル オブジェクトのリスト

使用法

名前	説明
<code>[-copy_sources]</code>	作成したプロジェクトに Synplify プロジェクト ファイルのソースをすべてコピーします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	インポートする Synplify プロジェクト ファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

Synplify 合成プロジェクト ファイル (`.prj`) を、合成で使用するさまざまなソース ファイルも含めて現在のプロジェクトに追加します。

引数

`-copy_sources` (オプション): Synplify プロジェクト ソース ファイルを、現在のディレクトリから参照するのではなく、ローカル プロジェクト ディレクトリ 構造にコピーします。デフォルトでは、現在のディレクトリからソース ファイルが参照されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): ソース ファイルをインポートする Synplify プロジェクト ファイルの名前を指定します。

例

次の例では、新しいプロジェクトを作成し、指定の Synplify プロジェクト ファイルにインポートしています。Synplify プロジェクトからのソース ファイルは、ローカル プロジェクト ディレクトリにコピーされます。

```
create_project syn_test C:/Data/FPGA_Design/syn_test  
import_synplify -copy_sources C:/Data/syn_data.prj
```

関連項目

- [create_project](#)

import_xise

作成したプロジェクトに XISE プロジェクト ファイルの設定をインポートします。

構文

```
import_xise [-copy_sources] [-quiet] [-verbose] <file>
```

戻り値

true

使用法

名前	説明
<code>[-copy_sources]</code>	作成したプロジェクトに ISE ソースすべてをコピーします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	インポートする XISE プロジェクト ファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のプロジェクトに ISE プロジェクト ファイル (XISE) をインポートします。これにより、ISE プロジェクトを Vivado Design Suite にすばやく移行して、合成、シミュレーション、インプリメンテーションを実行できます。すべてのプロジェクト ソース ファイル、制約ファイル、シミュレーション ファイルおよび run 設定が ISE プロジェクトからインポートされ、現在のプロジェクトで作成され直されます。

このコマンドは、新しい空のプロジェクトで実行する必要があります。ソース ファイル、制約、および run 設定は ISE プロジェクトからインポートされるので、既存のソース ファイルや制約ファイルがある場合はすべて上書きされます。

引数

`-copy_sources` (オプション): ISE プロジェクトのソース ファイルを、現在のディレクトリから参照するのではなく、ローカル プロジェクト ディレクトリ構造にコピーします。デフォルトでは、現在のディレクトリからソース ファイルが参照されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 現在のプロジェクトにインポートする ISE プロジェクト ファイル (.xise) の名前を指定します。

例

次の例では、importISE という新規プロジェクトを作成し、ISE プロジェクト ファイル (first_use.xise) をインポートしています。

```
create_project importISE C:/Data/importISE import_xise \  
C:/Data/FPGA_design/ise_designs/drp_des/first_use.xise
```

注記: この例では `-copy_sources` オプションが指定されていないので、ISE プロジェクトのすべてのソース ファイルは現在の場所から参照されてプロジェクトに追加されます。

関連項目

- [create_project](#)

import_xst

指定した XST プロジェクト ファイルをインポートします。

構文

```
import_xst [-copy_sources] [-quiet] [-verbose] <file>
```

戻り値

XST ファイルからインポートされたファイル オブジェクトのリスト

使用法

名前	説明
<code>[-copy_sources]</code>	作成したプロジェクトに XST プロジェクト ファイルのソースをすべてコピーします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	インポートする XST プロジェクト ファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

XST 合成プロジェクト ファイル (XST の実行に使用されるさまざまなソース ファイル) を現在のプロジェクトに追加します。

引数

`-copy_sources` (オプション): XST プロジェクト ソース ファイルを、現在のディレクトリから参照するのではなく、ローカル プロジェクト ディレクトリ 構造にコピーします。デフォルトでは、現在のディレクトリからソース ファイルが参照されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): ソース ファイルをインポートする XST プロジェクト ファイルの名前を指定します。

例

次の例では、`xst_test` という新規プロジェクトを作成し、`drp_des.xst` ファイルをインポートしています。

```
create_project xst_test C:/Data/FPGA_Design/xst_test  
import_xst C:/Data/ise_designs/drp_des.xst
```

関連項目

- [create_project](#)

include_bd_addr_seg

アドレス空間からのセグメントを含めます。

構文

```
include_bd_addr_seg [-quiet] [-verbose] [<segment_to_include>]
```

戻り値

新しく含まれたセグメント オブジェクト、エラーが発生した場合は ""。

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<segment_to_include>]	含めるセグメントを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

AXI ペリフェラル アドレス セグメントの AXI マスターによるアクセスの除外を解除し、アドレス セグメントをマップされた状態に戻します。

ブロック デザインでは、AXI ペリフェラルのアドレス セグメントは次の 3 つのいずれかのステートになります。

- Unmapped (マップされていない): AXI ペリフェラル (スレーブ インターフェイス) は、AXI マスターに接続されていますが、ペリフェラルはマスターのアドレス空間のアドレス セグメントに割り当てられておらず、マスターで認識されません。
- Mapped (マップ済み): AXI ペリフェラルは AXI マスターに接続され、アドレス セグメントまたは範囲に割り当てられており、マスターでアクセス可能です。
- Excluded (除外): AXI ペリフェラルは AXI マスターにマップされ、アドレス セグメントに割り当てられていますが、マスターでアクセスできません。マスターのアドレス空間の AXI スレーブが割り当てられているアドレス セグメントは、占有されているとみなされます。

exclude_bd_addr_seg コマンドを使用すると、特定のアドレス セグメントを AXI マスターによるアクセスから除外できます。include_bd_addr_seg コマンドは、マップされたアドレス セグメントへのアクセスを復元します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<segment_to_include>`: AXI マスター アドレス空間に復元する 1 つのアドレス セグメント オブジェクト (`bd_addr_seg`) を指定します。

例

次の例では、AXI ペリフェラル アドレス セグメントへの AXI マスターによるアクセスを復元しています。

```
include_bd_addr_seg [get_bd_addr_segs microblaze_1/Data/SEG_axi_gpio_1_Reg]
```

関連項目

- [assign_bd_address](#)
- [create_bd_addr_seg](#)
- [exclude_bd_addr_seg](#)
- [get_bd_addr_segs](#)
- [get_bd_addr_spaces](#)

infer_diff_pairs

CSV または XDC ファイルからインポートされたポートに対して差動ペアを推論します。

構文

```
infer_diff_pairs [-file_type <arg>] [-quiet] [-verbose] [<file>...]
```

使用法

名前	説明
[-file_type]	入力ファイル タイプを指定します。有効な値は csv または xdc です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	ピン プランニング CSV ファイルまたは XDC ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

`infer_diff_pairs` コマンドは、I/O ピン プランニング プロジェクトで `read_csv` または `read_xdc` コマンドを使用して I/O ピン情報をインポートした後に使用できます。

信号名、差動ペア信号、差動ペア タイプ、I/O 規格など、ファイルで差動ペアを定義するために使用される属性は複数あります。

ツールでは、次の条件で差動ペアが特定されます。

- 差動ペアの定義が一致: 2 つのポート エントリで、[DiffPair Signal] にお互いの [Signal Name] の値が指定され、[DiffPair Type] で一方に N、もう一方に P が指定されている。ツールで差動ペアを作成する際に、[I/O Standard] などのほかの属性に互換性があるかどうかチェックされます。
- 差動ペアの定義が片方のみ: 2 つのポート エントリの [DiffPair Type] で一方に N、もう一方に P が指定されているが、1 つのポートでのみ [DiffPair Signal] にもう一方の [Signal Name] の値が指定されている。この場合、すべての属性に互換性があれば、差動ペアが作成されます。
- 1 つのポートのみでの差動ペア定義: 1 つのポート エントリで、[I/O Standard] に差動規格、[DiffPair Type] に値が指定されているが、[DiffPair Signal] に指定されている信号が CSV ファイルに含まれない。この場合、このポートのプロパティと一致するプロパティで、もう一方の差動ペア (N 側または P 側) が作成されます。
- 名前から差動ペアを推論: 2 つのポート エントリで、N 側と P 側が推論されるような名前が指定されている。この場合、すべての属性に互換性があれば、差動ペアが推論されます。

CSV または XDC ファイルからポート定義を読み込むと、データから差動ペアが推論されたことがレポートされます。`infer_diff_pairs` コマンドを使用すると、必要に応じてこれらの差動ペアを推論できます。

引数

`-file_type [csv | xdc]` (オプション): 差動ペアを推論する際に読み込むファイルのタイプを指定します。有効なファイル タイプは CSV および XDC です。デフォルトはありません。 `-file_type` オプションの指定は必須です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (オプション): 以前にインポートしたファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した XDC ファイルをインポートし、差動ペアを推論しています。

```
read_xdc C:/Vivado_Install/io_1.xdc
infer_diff_pairs C:/Vivado_Install/io_1.xdc -file_type xdc
```

関連項目

- [read_csv](#)
- [read_xdc](#)

instantiate_example_design

開いているプロジェクトの定義済みテンプレートから、サンプル デザインを作成します。

構文

```
instantiate_example_design [-design <arg>] [-hier <arg>] [-project <arg>]
    [-project_location <arg>] [-options <args>] [-quiet] [-verbose]
    <template>
```

戻り値

適用されたテンプレートの名前。

使用法

名前	説明
[-design]	ブロック デザイン名を指定します。
[-hier]	階層ブロックを指定します。
[-project]	プロジェクト名を指定します。
[-project_location]	プロジェクトの場所を指定します。デフォルトは. です。
[-options]	設定可能なオプションを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<template>	テンプレート デザインの名前を指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

開いているプロジェクトの定義済みテンプレートから、サンプル デザインを作成します。開いているプロジェクトで指定されているターゲット パーツは、サンプル デザインの SUPPORTED_PARTS プロパティで指定されているターゲット パーツに互換したものである必要があります。そうでないと、エラーが返されます。

エンベデッド プロセッサのサンプル デザイン (base_microblaze および base_zynq) は、サンプル デザインを Vivado IP インテグレーターで開いているブロック デザインで作成する必要があります。エンベデッド プロセッサのサンプル デザインでは、ターゲット パーツではなく、BOARD_PART プロパティでボードを定義する必要があります。詳細は、current_board_part コマンドを参照してください。

このコマンドを実行すると、使用されたサンプル デザインの名前と実行されたコマンドが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-design <arg>` (オプション): エンベデッド プロセッサのサンプル デザインの場合に、サンプル デザインにインスタンスエートする、開いている現在のブロック デザインの名前を指定します。このオプションは、エンベデッド プロセッサのサンプル デザインでは必須です。`-design` オプションを指定しないとエラーが返されます。

`-hier` (オプション): 階層ブロックを指定します。

`-options <args>` (オプション): サンプル デザインの設定可能なプロパティの値を指定します。



ヒント: サンプル デザインのプロパティは、`report_property` または `get_property` コマンドを使用して返すことができます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<template>`: 指定のデザインにインスタンスエートするテンプレート デザインを指定します。テンプレートは、名前指定するか、または `get_example_designs` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、新規プロジェクトを作成し (同じ名前のプロジェクトが存在する場合は上書き)、プロジェクトの `BOARD_PART` プロパティを設定して、Vivado IP インテグレーターに新しい空のブロック デザインを作成した後、Zynq エンベデッド プロセッサのサンプル デザインをインスタンスエートしています。

```
create_project zynq1 -force
set_property BOARD_PART em.avnet.com:zed:1.3 [current_project]
create_bd_design myFirstZynq
instantiate_example_design -design myFirstZynq \
[lindex [get_example_designs] 1]
```

次の例では、指定のサンプル デザインのコンフィギュレーション プロパティをレポートしています。

```
report_property [lindex [get_example_designs] 3] CONFIG.*
```

次の例では、新規プロジェクトを作成し、プロジェクトのターゲット パーツを設定して新しい空のブロック デザインを開き、サンプル デザインをインスタンスエートしています。

```
create_project mb1 C:/Data/Vivado_Tutorial/Tutorial_Created_Data/mb1
set_property board_part xilinx.com:kcu105:part0:1.1 [current_project]
create_bd_design design_1
instantiate_example_design -design design_1 \
-options { Data_Cache.VALUE 8K Include_DDR4.VALUE true \
Local_memory.VALUE 128K }\
xilinx.com:design:config_mb:1.0
```

関連項目

- [create_bd_design](#)
- [create_project](#)
- [get_example_designs](#)
- [set_property](#)

instantiate_template_bd_design

定義済みテンプレート IP インテグレーターのブロック デザインを作成します。

構文

```
instantiate_template_bd_design -design <arg> [-hier <arg>]  
                                [-options <args>] [-quiet] [-verbose] <template>
```

戻り値

適用されたテンプレートの名前。

使用法

名前	説明
-design	ブロック デザイン名を指定します。
[-hier]	階層ブロックを指定します。
[-options]	設定可能なオプションを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<template>	テンプレート デザインの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado Design Suite の IP インテグレーターのテンプレート ブロック デザインからサンプル デザインを作成します。

テンプレート ダイアグラムは、既存の開いているブロック デザインに作成されます。現在のプロジェクトまたはインメモリ プロジェクトで指定されているターゲット パーツが、テンプレート デザインと互換性のあるものである必要があります。そうでないと、エラーが返されます。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-design <arg> (必須): テンプレート ダイアグラムにインスタンスiertするブロック デザインの名前を指定します。存在し、IP インテグレーターで開いているブロック デザインを指定する必要があります。そうしないと、エラーが返されます。

-hier (オプション): 階層ブロックを指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<template>`: 指定のデザインにインスタンスエートするテンプレート ダイアグラムを指定します。テンプレートは、名前で指定するか、または `get_template_bd_designs` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、指定のデザインに指定のテンプレート ブロック デザインを構築しています。

```
instantiate_template_bd_design -design myFirstZynq \  
[lindex [get_template_bd_designs] 1]
```

関連項目

- [get_template_bd_designs](#)

iphys_opt_design

最適化をインタラクティブに実行します。

構文

```
iphys_opt_design [-fanout_opt] [-critical_cell_opt] [-placement_opt]
  [-rewire] [-net <arg>] -cluster <args> -place_cell <args> [-place]
  [-dsp_register_opt] [-bram_register_opt] [-uram_register_opt]
  [-shift_register_opt] [-cell <arg>] [-packing] [-unpacking]
  [-port <arg>] [-critical_pin_opt] [-skipped_optimization]
  [-insert_negative_edge_ffs] [-hold_fix] [-slr_crossing_opt]
  [-auto_pipeline] [-quiet] [-verbose]
```

使用法

名前	説明
[-fanout_opt]	ファンアウト最適化を実行します。
[-critical_cell_opt]	タイミングクリティカル ネットをセルを複製して最適化します。
[-placement_opt]	タイミングクリティカル ネットの遅延を削減するようセルを移動します。
[-rewire]	配線を再実行します。
[-net]	最適化するネットを指定します。
-cluster	ロード ピンのクラスターを指定します。
-place_cell	セルまたはピンに接続されているセルを配置して固定します。
[-place]	配置を再実行します。
[-dsp_register_opt]	DSP レジスタの最適化を実行します。
[-bram_register_opt]	BRAM レジスタの最適化を実行します。
[-uram_register_opt]	UltraRAM レジスタの最適化を実行します。
[-shift_register_opt]	シフト レジスタの最適化を実行します。
[-cell]	最適化するセルを指定します。
[-packing]	DSP/BRAM へのパックを実行します。
[-unpacking]	DSP/BRAM へのパックを解除します。
[-port]	最適化された DSP/BRAM のポートを指定します。
[-critical_pin_opt]	ピン スワップ最適化を実行します。
[-skipped_optimization]	スキップされた変更を指定します。
[-insert_negative_edge_ffs]	ホールドが大きくなるのを回避するため、立ち下がりエッジでトリガーされるフリップフロップを挿入します。
[-hold_fix]	ホールド修正最適化用にバッファァーを挿入します。
[-slr_crossing_opt]	SLR をまたぐネットを最適化します。
[-auto_pipeline]	自動パイプライン処理を実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Tools (ツール)

説明

`iphys_opt_design` コマンドは、クリティカル セルの複製やブロック RAM からのレジスタの取り出しなど、`phys_opt_design` コマンドで実行された最適化を記述します。`iphys_opt_design` コマンドには、最適化済み論理ネットリストと、最適化済みネットリストに必要な配置の変更を再作成するために必要な情報がすべて含まれます。

インタラクティブ物理最適化は、次の 2 つの方法で使用できます。

- 全体的な配置結果とデザイン パフォーマンスを向上するため、配置前のネットリストに配置後の物理最適化を適用する。
- 物理最適化を必要に応じて繰り返すことができるように Tcl スクリプトに保存する。

`phys_opt_design` コマンドで実行されたさまざまな最適化は、`write_iphys_opt_tcl` コマンドを使用して `iphys_opt` Tcl スクリプトに記述し、`read_iphys_opt_tcl` コマンドを使用してデザインに読み込むことができます。



ヒント: `iphys_opt_design` コマンドは、`iphys_opt` Tcl スクリプト ファイル内で使用するためのコマンドです。これらのコマンドは、`iphys_opt` Tcl スクリプト内で編集することはできますが、コマンド ラインでは指定しないでください。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-fanout_opt` (オプション): 指定したネットに対し、ドライバーを複製して遅延を削減することにより、遅延ドリブンの最適化を実行します。

`-critical_cell_opt` (オプション): 指定したネットのセルを複製して遅延を削減します。

`-placement_opt` (オプション): 指定したネットの遅延を削減するようセルを移動します。

`-rewire` (オプション): ロジック コーンを再構築してロジック レベルを削減し、クリティカル信号の遅延を削減します。

`-net <arg>` (オプション): 最適化を適用するネットを指定します。

`-cluster <args>` (必須): ロード ピンのクラスターを指定します。

`-place_cell <args>` (必須): 指定したデバイス サイトの指定したピンに接続されているセルを配置します。

`-place` (オプション): 配置を再実行します。

`-dsp_register_opt` (オプション): レジスタをスライスから DSP ブロックに移動したり、DSP ブロックからスライスに移動したりして、クリティカル パス遅延を削減します。

`-bram_register_opt` (オプション): レジスタをスライスからブロック RAM に移動したり、ブロック RAM からスライスに移動したりして、クリティカル パス遅延を削減します。

`-uram_register_opt` (オプション): レジスタをスライスから UltraRAM に移動したり、UltraRAM からスライスに移動したりして、クリティカル パス遅延を削減します。

-shift_register_opt (オプション): シフトレジスタの最適化を実行し、シフトレジスタセル (SRL) とほかのロジックセル間の負の Slack パスのタイミングを向上します。

-cell <arg> (オプション): 最適化を適用するセルを指定します。

-packing (オプション): DSP/BRAM へのパックを実行します。

-unpacking (オプション): DSP/BRAM へのパックを解除します。

-port <arg> (オプション): 最適化を適用するセル上のポートを指定します。

-critical_pin_opt: LUT 入力に対して、論理ピンと物理ピンのリマップ (ピンスワップ) を実行し、クリティカルパスのタイミングを向上します。

-skipped_optimization (オプション): 指定の最適化を実行されなかったとして定義します。これらは、phys_opt_design により最適化されたロジックに適切な場所が見つからなかったためにスキップされた最適化です。たとえば、Slack を向上するためのブロック RAM レジスタの最適化が、レジスタに適切な場所が見つからなかったためにスキップされた場合などです。

-insert_negative_edge_ffs (オプション): ホールドタイミングを制御するため、立ち下がりエッジでトリガーされるフリップフロップを挿入します。

-hold_fix (オプション): ホールド修正最適化用にデータパス遅延バッファを挿入します。

-slr_crossing_opt (オプション): SLR 間の接続のパス遅延を削減するため、配置後または配線後の最適化を実行します。この最適化では、複製の後に SLR をまたぐ部分のドライバー、ロード、またはその両方の位置を調整します。UltraScale および UltraScale+ デバイスで使います。

-auto_pipeline (オプション): デザインの自動パイプライン処理を実行します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、指定したネットおよびポートのクラスターに対してクリティカルセルの最適化を実行しています。

```
iphys_opt_design -critical_cell_opt -net \
  {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
  /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST3 \
  /Q_PNI_MULT_INST/pn_mult_reg[3][0]} \
  -cluster {pn_mult[3]_i_14_replica { \
  {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
  /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST2 \
  /Q_ADD_INST/pn_mult_reg[3]_i_6_CARRY8/S[0]}}} \
  -cluster {pn_mult[3]_i_14_replica_1 { \
  {ADUR_CORE_INST/CPE_INST/CPE_ANT_RESOURCE_TDM_INST0 \
  /CPE_ANT_LINE_IQ_TDM_ANT0_INST/CPE_PN_MULT_INST/CPE_PN_MUL_INST0 \
  /Q_ADD_INST/pn_mult_reg[3]_i_10_CARRY8/S[0]}}} \
```

次の例では、指定したセルに対してシフト レジスタ最適化を実行しています。

```
iphys_opt_design -shift_register_opt -cell \  
  {ADUR_CORE_INST/EMIF_INTERFACE_INST/EMIF_HOST_IF_INST/DLY_INST1 \  
    /PD_INST_FPGA/delay_chain_reg[9][16]_sr19} -port D
```

関連項目

- [phys_opt_design](#)
- [read_iphys_opt_tcl](#)
- [write_iphys_opt_tcl](#)

launch_chipscope_analyzer

このコマンドは実行できません。

構文

```
launch_chipscope_analyzer [-run <arg>] [-csproject <arg>] [-quiet]  
[-verbose]
```

使用法

名前	説明
[-run]	ChipScope Analyzer で開くインプリメント済み run を指定します。
[-csproject]	ChipScope プロジェクトを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[ToolLaunch \(ツール起動\)](#)

説明

アクティブな run または指定したインプリメント済みデザイン run に対して ChipScope™ Pro Analyzer ツールを起動します。インプリメンテーションの前に、ネットリスト デザインを ChipScope で使用するよう設定するには、create_debug_core、create_debug_port、および connect_debug_port コマンドを使用します。

インプリメント済みデザインに対して launch_chipscope_analyzer コマンドを実行するには、BitGen でビットストリーム ファイルを生成しておく必要があります。BitGen が実行されていない場合、エラー メッセージが表示されます。

注記: ビットストリーム ファイルを作成するには、write_bitstream コマンドを使用するだけでは不十分です。2 番目の例に示されている手順に従う必要があります。

引数

-run <arg>: ChipScope Pro Analyzer を起動するのに使用する run の名前を指定します。run は、インプリメントが完了しており、ビットストリーム ファイル(.bit) が生成されていることが必要です。ChipScope では、指定の run のビットストリーム ファイルと debug_nets.cdc ファイルが使用されます。

-csproject <arg>: ChipScope Pro Analyzer で開くプロジェクトの名前を指定します。プロジェクト名を指定しない場合、デフォルトのプロジェクト名 csdefaultproj.cpj が使用されます。プロジェクト名を指定する場合は、拡張子 .cpj も含める必要があります。

注記: プロジェクトは project/project.data/sources_1/cs フォルダに作成されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、使用するインプリメンテーション `run` を指定し、作成する ChipScope プロジェクトの名前を指定して、ChipScope Pro Analyzer を起動します。

```
launch_chipscope_analyzer -run impl_3 -csproject impl_3_cs_project
```

次の例では、`impl_4` という `run` に `add_step Bitgen` プロパティを設定し、`impl_4 run` を実行して、その `run` に対して ChipScope Pro Analyzer を起動します。

```
set_property add_step Bitgen [get_runs impl_4]
launch_runs impl_4 -jobs 2
launch_chipscope_analyzer -run impl_4
```

注記: この例では、ChipScope プロジェクトの名前は `csdefaultproj.cpj` となります。

関連項目

- [connect_debug_port](#)
- [create_debug_core](#)
- [create_debug_port](#)
- [launch_runs](#)
- [set_property](#)
- [write_bitstream](#)

launch_impact

このコマンドは実行できません。

構文

```
launch_impact [-run <arg>] [-ipf <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-run]</code>	iMPACT で開くインプリメント済み run を指定します。
<code>[-ipf]</code>	iMPACT のプロジェクトを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[ToolLaunch \(ツール起動\)](#)

説明

iMPACT を起動し、デバイスをコンフィギュレーションしてプログラム ファイルを生成します。また、デザイン コンフィギュレーション データをリードバックして検証、コンフィギュレーション問題をデバッグ、XSVF ファイルを実行できます。

iMPACT を使用する前に、`write_bitstream` コマンドを使用してビットストリーム ファイルを生成する必要があります。

このコマンドを実行すると、読み込まれたファイルのリストが返されます。

引数

`-run` (オプション): 指定した run で iMPACT を起動します。run を指定しない場合、アクティブ インプリメンテーション run で iMPACT が起動します。

`-ipf` (オプション): 結果を保存する iMPACT プロジェクト ファイルを指定します。iMPACT プロジェクト ファイル (IPF) には、iMPACT の前回のセッションからの情報が含まれます。ターゲット デバイスは、指定した IPF ファイルの設定に基づいてコンフィギュレーションされます。`-ipf` オプションを指定しない場合、ターゲット デバイスはデフォルト設定でコンフィギュレーションされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したインプリメンテーション `run` で iMPACT を起動しています。

```
launch_impact -run impl_3
```

関連項目

- [write_bitstream](#)

launch_runs

run のセットを実行します。

構文

```
launch_runs [-jobs <arg>] [-scripts_only] [-lsf <arg>] [-sge <arg>]
  [-cluster_configuration <arg>] [-dir <arg>] [-to_step <arg>]
  [-next_step] [-host <args>] [-remote_cmd <arg>] [-email_to <args>]
  [-email_all] [-pre_launch_script <arg>] [-post_launch_script <arg>]
  [-custom_script <arg>] [-force] [-quiet] [-verbose] <runs>...
```

使用法

名前	説明
[-jobs]	並列実行するジョブ数を指定します。デフォルトは 1 です。
[-scripts_only]	スクリプトの生成のみを実行します。
[-lsf]	LSF を使用してジョブを実行します。LSF に渡す bsub コマンドラインを引数として指定する必要があります。
[-sge]	SGE を使用してジョブを実行します。SGE に渡す bsub コマンドラインを引数として指定する必要があります。
[-cluster_configuration]	ジョブを特定のクラスター コンフィギュレーションで実行します (GUI 設定からクラスター コンフィギュレーションを選択)。デフォルトは空です。
[-dir]	実行ディレクトリを指定します。
[-to_step]	実行する最後のステップを指定します。複数の run を実行する際は無視されます。-next_step と共に使用することはできません。
[-next_step]	次のステップを実行します。複数の run を実行する際は無視されます。-to_step と共に使用することはできません。
[-host]	指定したリモート ホストで指定したジョブ数を実行します。例: -host {machine1 2} -host {machine2 4}。
[-remote_cmd]	リモート ホストにログインするコマンドを指定します。デフォルトは ssh -q -o BatchMode=yes です。
[-email_to]	ジョブが完了したときに通知を送付する電子メール アドレスを指定します。-host オプションと共に使用する必要があります。
[-email_all]	各ジョブの完了時に電子メールを送信します。-host オプションと共に使用する必要があります。
[-pre_launch_script]	各ジョブの前に実行するスクリプトを指定します。-host オプションと共に使用する必要があります。
[-post_launch_script]	各ジョブの完了後に実行するスクリプトを指定します。-host オプションと共に使用する必要があります。
[-custom_script]	run 名とユーザー run スクリプトのマップを含むユーザー run スクリプト マップ ファイルを指定します。
[-force]	未決定の制約変更があっても、コマンドを実行します。変更は失われます (パーシャル リコンフィギュレーション デザインの場合)。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

名前	説明
<code><runs></code>	実行する run を指定します。

カテゴリ

Project (プロジェクト)

説明

Vivado ツールをプロジェクト モードで実行している場合に合成 run およびインプリメンテーション run を実行します。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

run は、`create_run` コマンドを使用して設定し、その属性を `set_property` コマンドを使用して設定しておく必要があります。同じ `launch_runs` コマンドで合成 run とインプリメンテーション run の両方を指定できますが、インプリメンテーション run を実行するには、親の合成 run が完了している必要があります。

非プロジェクト モードでは、Vivado 合成は `synth_design` コマンドを使用して直接実行できるので、run を定義しておく必要はありません。

非プロジェクト モードでは、`opt_design`、`power_opt_design`、`place_design`、`route_design`、`phys_opt_design`、および `write_bitstream` コマンドを使用して、Vivado インプリメンテーションの各段階を実行できます。

引数

`-jobs <arg>` (オプション): ローカル ホストで並列実行するジョブ数を指定します。リモート ホストのジョブ数は、`-host` オプションの一部として指定されるので、`-jobs` と `-host` の両方を指定する必要はありません。

`-scripts_only` (オプション): 指定した run ごとに `runme.bat` というスクリプトを生成し、後で実行できるようにします。

`-lsf <arg>` (オプション): IBM Platform LSF (Load Sharing Facility) を使用して合成およびインプリメンテーション run を実行します。LSF に渡す `bsub` コマンド ラインを引数として指定する必要があります。

`-sge <arg>` (オプション): Oracle Grid Engine または Sun Grid Engine (SGE) を使用して合成およびインプリメンテーション run を実行します。SGE に渡す `qsub` コマンド ラインを引数として指定する必要があります。

`-dir <arg>` (オプション): run 結果を書き込むディレクトリを指定します。指定したディレクトリに、各 run ごとに個別のフォルダーが作成されます。デフォルトでは、各 run の結果が `<project>.runs` ディレクトリの下に個別のフォルダーで書き込まれます。

`-to_step <arg>` (オプション): run をインプリメンテーション プロセスの指定した段階まで実行し、停止します。たとえば、インプリメンテーションを `place_design` の段階まで実行し、停止できます。これにより、run 全体を完了せずに特定の段階の run 結果を確認できます。インプリメンテーション run での有効な段階は、次のとおりです。

- `opt_design`: ターゲット デバイスのリソースがより効率的に使用されるように、論理デザインを最適化します (オプション)。これはオプションですが、通常はデフォルトでイネーブルになっています。
- `power_opt_design`: インプリメント済み FPGA の消費電力を削減するため、ロジック デザインの要素を最適化します (オプション)。
- `place_design`: ロジック セルをターゲット デバイスに配置します。この段階は必須です。

- `power_opt_design` (Post-Place): 配置済みロジック エLEMENTの消費電力を最適化します (オプション)。複数の単語が含まれているので、ダブルクォーテーションまたは波かっこで囲んで指定する必要があります (例: `-to_step "power_opt_design (Post-Place)"`)。
- `phys_opt_design`: ファンアウトの大きいネットのドライバーを複製してロードを分散することにより、デザインのタイミングを最適化します (オプション)。
- `route_design`: デザインの接続をターゲット FPGA 上に配線します。この段階は必須です。
- `write_bitstream`: ザイリンクス デバイス コンフィギュレーションのビットストリーム ファイルを生成します。この段階は必須です。

注記: `-to_step` オプションで指定するには、`set_property` コマンドを使用して、インプリメンテーション run でその段階をイネーブルにしておく必要があります。そうでないと、エラーが返されます。

`-next_step` (オプション): 前回の run の実行を停止した段階から再開します。このオプションは、`-to_step` オプションを使用して実行した前回の run を完了するために使用します。

注記: `-to_step` と `-next_step` オプションを同時に指定することはできません。また、複数の run を実行する場合は無視されます。

`-host <args>` (オプション): 指定したリモート ホストで指定したジョブ数を実行します。オプションは、「`-host {machine1 2}`」のように `{<hostname> <jobs>}` という形式で使用します。`-host` オプションを指定しない場合、run はローカル ホストから実行されます。

注記: このオプションは、Linux でのみサポートされます。

`-remote_cmd <arg>` (オプション): ジョブを実行するため、リモート ホストにログインするのに使用するコマンドを指定します。デフォルトのリモート コマンドは `"ssh -q -o BatchMode=yes"` です。

`-email_to <args>` (オプション): run が完了したことを示す通知を送付する電子メール アドレスを指定します。このオプションを使用する場合は、電子メール通知を送信するため、`-host` オプションも使用して稼働中の SMTP サーバーを指定する必要があります。

`-email_all` (オプション): 各 run が完了するごとに電子メールが送付されます。このオプションを使用する場合は、電子メール通知を送信するため、`-host` オプションも使用して稼働中の SMTP サーバーを指定する必要があります。

`-pre_launch_script <arg>` (オプション): 指定のホスト上で各ジョブの前に実行する Tcl スクリプトを指定します。このオプションを使用する場合は、`-host` も使用する必要があります。

`-post_launch_script arg` (オプション): 指定のホスト上ですべてのジョブを完了した後に実行するシェル スクリプトを指定します。このオプションを使用する場合は、`-host` も使用する必要があります。

`-custom_script <arg>` (オプション): run 名とその run で実行するスクリプトのマップを指定するスクリプト マップ ファイルを指定します。

`-force` (オプション): パーシャル リコンフィギュレーション デザインの制約の変更が未決定かどうかにかかわらず、run を実行します。

注記: このオプションは、パーシャル リコンフィギュレーション プロジェクトでのみ使用できます。未決定の制約の変更は、指定した run では失われます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<runs>` (必須): 実行する合成 run およびインプリメンテーション run を指定します。1 つまたは複数の run を指定できます。

例

次の例では、並列実行するジョブ数を 2 に設定し、3 つの異なる合成 run を実行します。

```
launch_runs synth_1 synth_2 synth_4 -jobs 2
```

注記: 各 run の結果は、`<project>.runs` ディレクトリの `synth_1`、`synth_2`、および `synth_4` フォルダーにそれぞれ書き込まれます。

次の例では、run 結果を書き込むディレクトリが作成されます。この場合、指定したディレクトリに `impl_3`、`impl_4`、および `synth_3` が書き込まれます。また、`-scripts_only` オプションにより、各フォルダーに `runme.bat` スクリプトが書き込まれますが、この段階では run は実行されません。

```
launch_runs impl_3 impl_4 synth_3 -dir C:/Data/FPGA_Design/results -scripts_only
```

次の例では、`impl_1` run に対して Vivado Implementation 2013 フローを設定し、オプションの最適化のいくつかをイネーブルにした後、`place_design` 段階まで実行しています。

```
set_property flow {Vivado Implementation 2013} [get_runs impl_1]
set_property STEPS.POWER_OPT_DESIGN.IS_ENABLED true [get_runs impl_1]
set_property STEPS.POST_PLACE_POWER_OPT_DESIGN.IS_ENABLED true \
[get_runs impl_1]
set_property STEPS.PHYS_OPT_DESIGN.IS_ENABLED true [get_runs impl_1]
launch_runs -to_step place_design impl_1
```

関連項目

- [create_run](#)
- [get_runs](#)
- [opt_design](#)
- [phys_opt_design](#)
- [place_design](#)
- [power_opt_design](#)
- [route_design](#)
- [set_property](#)
- [synth_design](#)
- [write_bitstream](#)

launch_simulation

シミュレーションを実行します。

構文

```
launch_simulation [-step <arg>] [-simset <arg>] [-mode <arg>] [-type <arg>]
                  [-scripts_only] [-of_objects <args>] [-absolute_path]
                  [-install_path <arg>] [-noclean_dir] [-quiet] [-verbose]
```

使用法

名前	説明
[-step]	実行するシミュレーション手順を指定します。有効な値は all、compile、elaborate、simulate です。デフォルトは all で、すべての手順が実行されます。
[-simset]	シミュレーション ファイルセットの名前を指定します。
[-mode]	シミュレーション モードを指定します。有効な値は behavioral、post-synthesis、post-implementation で、デフォルトは behavioral です。
[-type]	ネットリストのタイプを指定します。有効な値は functional、timing です。このオプションは、-mode を post-synthesis または post-implementation に設定している場合にのみ有効です。
[-scripts_only]	スクリプトの生成のみを実行します。
[-of_objects]	指定したオブジェクトのコンパイル順ファイルを生成します。-scripts_only オプションを使用している場合にのみ有効です。
[-absolute_path]	デザイン ソース ファイルのパスを絶対パスにします。
[-install_path]	インストール ディレクトリ パスを指定します。
[-noclean_dir]	シミュレーション実行ディレクトリ ファイルを削除しません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[ToolLaunch \(ツール起動\)](#)、[Simulation \(シミュレーション\)](#)

説明

シミュレータを実行し、デザインを解析および検証します。

launch_simulation コマンドを実行すると、ターゲット シミュレータ用のスクリプト ファイルが作成され、このファイルがシミュレーション実行ディレクトリで実行されます。シミュレーション結果は、シミュレーション実行ディレクトリに作成されたログ ファイルに保存されます。

特定のシミュレータでシミュレーションを実行するには、デザイン プロジェクトの TARGET_SIMULATOR プロパティを設定してターゲット シミュレータを指定しておく必要があります。

```
set_property TARGET_SIMULATOR <name> [current_project]
```

TARGET_SIMULATOR プロパティに有効な値は XSim、ModelSim、IES、Xcelium、VCS、Riviera、および ActiveHDL で、デフォルト値は XSim (Vivado シミュレータ) です。

ターゲット シミュレータは、Vivado IDE から設定できます。プロジェクトを作成するか開き、[Tools]→[Settings]→[Simulation] をクリックし、[Target simulator] ドロップダウン リストからターゲット シミュレータを選択します。選択可能なシミュレータは [Vivado Simulator]、[ModelSim Simulator]、[Questa Advanced Simulator]、[Incisive Enterprise Simulator (IES)]、[Xcelium Parallel Simulator]、[Verilog Compiler Simulator (VCS)]、[Riviera-PRO Simulator]、および [Active-HDL Simulator] です。



ヒント: シミュレータの中には、Linux でのみ使用可能なもの、Windows のみで使用可能なものがあります。

launch_simulation コマンドは、コンパイル、エラボレート、シミュレーションの 3 つの手順から構成されるプロセスを使用します。プロセスの各手順に対してターゲット シミュレーション用のスクリプト ファイル (compile.bat、elaborate.bat、simulate.bat) が作成され、シミュレーション実行ディレクトリに保存されます。



ヒント: Linux では、スクリプト ファイルの拡張子は .sh ではなく .bat になります。

デフォルトでは、launch_simulation コマンドでこれらのスクリプト ファイルが順に実行されます。 -scripts_only オプションを使用すると、スクリプト ファイルを実行せずにスクリプトのみを作成できます。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-simset <arg> (オプション): シミュレーションで使用するシミュレーション テストベンチおよびソースを含むシミュレーション ファイルセットの名前を指定します。指定しない場合、現在のシミュレーション ファイルセットが使用されます。

-mode [behavioral | post-synthesis | post-implementation] (オプション): 構文を検証してデザインが意図したとおりに動作することを確認する HDL デザイン ソースのビヘイビアー シミュレーション、合成後のネットリストの論理またはタイミング シミュレーション、配置配線後の回路の動作を検証するインプリメンテーション後の論理またはタイミング シミュレーションを指定します。デフォルト モードは behavioral です。

-type [functional | timing] (オプション): ネットリストのみの論理シミュレーションを指定するか、ネットリストと SDF ファイルを使用したタイミング シミュレーションを指定します。-mode オプションが post-synthesis または post-implementation に指定されている場合に指定する必要があります。-mode behavioral の場合は使用できません。合成後のタイミング シミュレーションでは、synth_design コマンドからの SDF コンポーネント遅延が使用されます。インプリメンテーション後のタイミング シミュレーションでは、place_design および route_design コマンドからの SDF 遅延が使用されます。



重要: -type オプションを -mode behavioral と共に使用すると、エラーが返されます。

-scripts_only (オプション): ターゲット シミュレータ用のシミュレーション スクリプトのみを生成します。compile、elaborate、および simulation 手順を開始するために作成されたスクリプトを実行することはありません。作成したスクリプト ファイルは、後でシミュレーション フローを実行するのに使用できます。

-of_objects <arg> (オプション): 指定した 1 つのサブデザインまたは複合ファイルのシミュレーションを実行します。サブデザインは、名前で指定するのではなく、get_files コマンドを使用してデザイン オブジェクトとして指定する必要があります。

`-absolute_path` (オプション): シミュレーション スクリプトで使用されるソースおよびインクルード パスを絶対パスで定義します。デフォルトでは、すべてのパスはシミュレーション実行ディレクトリの相対パスで定義されます。相対パスには、シミュレーション スクリプトで設定される `origin_dir` 変数が含まれますが、デザインおよびシミュレーション スクリプトの場所を移動した場合、`$origin_dir` 変数を編集してパスを指定できます。

`-install_path <arg>` (オプション): シミュレータ実行ファイル (`vlog.exe`、`ncvlog`、`vlogan`) を含むディレクトリを指定します。このオプションを指定しない場合、現在 `$PATH` で定義されているディレクトリが検索されます。

`-noclean_dir` (オプション): シミュレータを実行する前にシミュレーション実行ディレクトリのファイルを削除しません。デフォルトでは、シミュレータを実行する前にシミュレーション実行ディレクトリのファイルが削除されます。`-noclean_dir` オプションを使用すると、シミュレーション実行ディレクトリの既存のファイルはそのまま保持されます。ただし、シミュレータで生成された一部のファイルは、シミュレータを再実行すると上書きされるかアップデートされます。

`-step` (オプション): 実行するシミュレーション手順を指定します。有効な値は `all`、`compile`、`elaborate`、`simulate` です。



ヒント: デフォルトは `all` で、すべての手順が実行されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、Vivado シミュレータを使用してデザインのビヘイビア シミュレーションを実行しています。

```
set_property target_simulator "XSim" [current_project]
launch_simulation
```

次の例では、ModelSim シミュレータを使用してデザインの合成後の論理シミュレーションを実行しています。

```
set_property target_simulator "ModelSim" [current_project]
launch_simulation -mode "post-synthesis" -type "functional"
```

次の例では、Cadence IES シミュレータを使用してデザインのインプリメンテーション後の論理シミュレーションを実行しています。

```
set_property target_simulator "IES" [current_project]
launch_simulation -mode "post-implementation" -type "functional"
```

次の例では、Synopsys VCS シミュレータを使用してデザインのインプリメンテーション後のタイミング シミュレーションを実行しています。

```
set_property target_simulator "VCS" [current_project]
launch_simulation -mode "post-implementation" -type "timing"
```

次の例では、ターゲット シミュレータ用のビヘイビア シミュレーション スクリプトをシミュレーション実行ディレクトリに生成しています。

```
launch_simulation -scripts_only
```

次の例では、my_simset シミュレーション ファイルセットを使用して、ターゲット シミュレータでデザインのビヘイビア シミュレーション フローをシミュレーション実行ディレクトリで実行しています。

```
launch_simulation -simset [get_filesets my_simset]
```

次の例では、以前のシミュレーション ファイルを削除せずに、ターゲット シミュレータで char_fifo.xci IP のビヘイビア シミュレーション フローをシミュレーション実行ディレクトリで実行しています。

```
launch_simulation -noclean_dir -of_objects [get_files char_fifo.xci]
```

次の例では、生成されたスクリプト ファイルのソース ファイルの絶対パスを生成しています。

```
launch_simulation -absolute_path
```

次の例では、シミュレーション ツールを PATH 変数からでなく指定のインストール パスから選択しています。

```
launch_simulation -install_path /tools/ius/13.20.005/tools/bin
```

関連項目

- [close_sim](#)
- [current_sim](#)
- [relaunch_sim](#)
- [xsim](#)

limit_vcd

ディスク上の VCD ファイルの最大サイズをバイト数で指定します (\$dumplimit Verilog タスクと同等)。

構文

```
limit_vcd [-quiet] [-verbose] <filesize>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<filesize>	VCD ファイルの最大サイズをバイト数で指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

VCD (Value Change Dump) ファイルの最大サイズをバイト数で指定します。このコマンドは、Verilog の \$dumplimit シミュレータ指示子と同様の操作を実行します。

ファイルのサイズが指定の最大値に達した場合、ダンプ プロセスは停止し、VCD ファイルにファイル サイズの最大値に達したことを示すコメントが記述されます。

注記: limit_vcd コマンドを実行する前に、open_vcd コマンドを実行する必要があります。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<filesize> (必須): 開いている VCD ファイルの最大サイズをバイト数で指定します。

例

次の例では、現在の VCD オブジェクトの最大サイズを設定しています。

```
limit_vcd 1000
```

関連項目

- [checkpoint_vcd](#)
- [flush_vcd](#)
- [log_vcd](#)
- [open_vcd](#)

link_design

ネットリスト デザインを開きます。

構文

```
link_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-top <arg>]
            [-mode <arg>] [-pr_config <arg>] [-reconfig_partitions <args>]
            [-partitions <args>] [-ignore_timing] [-quiet] [-verbose]
```

戻り値

デザイン オブジェクト

使用法

名前	説明
[-name]	デザイン名
[-part]	ターゲット パーツを指定します。
[-constrset]	使用する制約ファイルセットを指定します。
[-top]	構造ネットリストが Verilog の場合、最上位モジュール名を指定します。
[-mode]	デザイン モードを指定します。有効な値は default、out_of_context で、デフォルトは default です。
[-pr_config]	デザインを開くときに適用する PR コンフィギュレーションを指定します。
[-reconfig_partitions]	デザインを開くときに読み込むリコンフィギャラブル パーティションを指定します。
[-partitions]	デザインを開くときに読み込むパーティションを指定します。
[-ignore_timing]	ネットリスト デザインをタイミング制約なしで開きます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Tools \(ツール\)](#)

説明

新規または既存のネットリスト デザインを開き、ネットリスト ファイルと制約をターゲット パーツとリンクしてデザインを作成します。このコマンドは、サードパーティ 合成ツールまたは synth_design コマンドを使用して Vivado 合成で生成されたネットリスト ソース ファイルで使します。

ネットリスト デザインを開くには、現在のソース ファイルセットの DESIGN_MODE プロパティを GateLvl に設定しておく必要があります。GateLvl に設定していないと、次のようなエラー メッセージが表示されます。

```
ERROR: The design mode of 'sources_1' must be GateLvl.
```

サードパーティ合成デザインでは、`-top` オプションが必要です。デザインのネットリストは、特定のモジュールに基づいている必要があります。プロジェクトベースのデザインでは、プロジェクトに TOP プロパティを指定できます。非プロジェクトモードでは、`link_design` コマンドに `-top` オプションを使用する必要があります。

RTL ソース ファイルを含むプロジェクトモードのデザインでは、`launch_runs` コマンドを使用して合成またはインプリメンテーションを実行し、`open_run` コマンドを使用してデザインを開きます。

非プロジェクトモードのデザインでは、`open_checkpoint` コマンドを使用してチェックポイントをメモリに読み込み、デザインを非プロジェクトモードで開きます。プロジェクトモードおよび非プロジェクトモードの詳細は、『Vivado Design Suite ユーザーガイド: デザインフローの概要』(UG892)を参照してください。

引数

`-name <arg>` (オプション): Vivado ツールで開いたときのネットリスト デザインの名前を指定します。この名前は参照用であり、デザインの最上位またはデザインに含まれるロジックとは関係ありません。

`-part <arg>` (オプション): 新規デザインを作成する際に使用するサイリンクス デバイスを指定します。`-part` オプションを指定しない場合、デフォルトのパーツが使用されます。

`-constrset <arg>` (オプション): デザインを開く際に使用する制約ファイルセットの名前を指定します。

注記: `-constrset` オプションには、既存の制約ファイルセットを指定する必要があります。新しいファイルセットを作成することはできません。新しいファイルセットを作成する場合は、`create_fileset` コマンドを使用します。

`-top <arg>` (オプション): ネットリストのデザイン階層の最上位モジュールの名前を指定します。



重要: EDIF ネットリストベースのデザインまたはデザイン チェックポイント (DCP) ファイルに対して `-top` オプションを使用する場合は、最上位セルの名前が EDIF または DCP ファイルで定義されている最上位セルと一致している必要があります。

`-mode [default | out_of_context]` (オプション): ブロックが合成済みで、I/O バッファの挿入がディスエーブルであった場合、生成された EDIF を `-mode out_of_context` を使用して Vivado Design Suite に読み込むことができます。これにより、I/O バッファなしのモジュールをインプリメントでき、入力または出力が未接続のために最適化で削除されるのを回避でき、デザイン用に DRC ルールを変更できます。詳細は、『Vivado Design Suite ユーザーガイド: 階層デザイン』(UG905)を参照してください。

`-pr_config <arg>` (オプション): プロジェクトモードのパーシャル リコンフィギュレーション (PR) フローで、デザインを開くときに適用する PR コンフィギュレーションを指定します。PR デザインでは、`create_pr_configuration` コマンドを使用してリコンフィギュラブル モジュール (RM) をデザインの各パーティション定義に関連付けます。このオプションは、RM のデザイン チェックポイント ファイルをデザインにリンクします。詳細は、『Vivado Design Suite ユーザーガイド: Dynamic Function eXchange』(UG909)を参照してください。

`-reconfig_partitions <args>` (オプション): デザインを開くときに読み込むリコンフィギュラブル パーティションを指定します。指定したリコンフィギュラブル パーティションには、デザインで正しく処理されるように、HD.RECONFIGURABLE プロパティが設定されます。

`-partitions <args>` (オプション): デザインを開くときに読み込む階層デザイン パーティションを指定します。階層デザイン パーティションには、HD.PARTITION プロパティが設定されます。詳細は、『Vivado Design Suite ユーザーガイド: 階層デザイン』(UG905)を参照してください。

`-ignore_timing` (オプション): ネットリスト デザインをタイミング制約なしで開きます。このオプションを使用すると、デザインをタイミング制約なしで評価するためにすばやく開くことができます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、Net1 というネットリスト デザインが新規に作成されます。

```
link_design -name Net1
```

注記: この例では、デフォルトのソース セット、制約セット、パーツが使用されます。

次の例では、Net1 というネットリスト デザインを開いて、使用する制約セットを指定しています。

```
link_design -name Net1 -constrset con1
```

関連項目

- [launch_runs](#)
- [open_checkpoint](#)
- [open_run](#)
- [synth_design](#)

list_features

使用可能な機能をリストします。

構文

```
list_features [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Tools \(ツール\)](#)

説明

Vivado Design Suite のメモリ使用量を削減するため、Tcl コマンドの機能グループがあり、それらの機能グループのコマンドは、その機能グループからのコマンドを使用するか、`load_features` コマンドを使用して機能を読み込むまで使用できません。

このコマンドは、Vivado Design Suite で使用可能な、`load_features` コマンドを使用して読み込むことができる機能をリストします。

注記: 機能が既に読み込まれている場合は、読み込み可能な機能としてリストされません。

このコマンドを実行すると、機能のリストまたはエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、Vivado Design Suite に読み込むことができる機能をリストしています。

```
list_features
```

関連項目

- [help](#)
- [load_features](#)

list_hw_samples

プローブのサンプル値を返します。

構文

```
list_hw_samples [-quiet] [-verbose] [<hw_probe>]
```

戻り値

サンプル

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_probe>]	hw_probe オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

現在のハードウェア ILA (hw_ila) 上の指定したハードウェア プローブ (hw_probe) オブジェクトからのデータ サンプルを記述します。

指定したプローブから返されるキャプチャされたサンプルの数は、ILA コアの DATA_DEPTH プロパティに指定されている数です。デフォルトのデータ深さは 1024 サンプルです。データ値は、hw_probe の DISPLAY_RADIX プロパティで指定された基数で返されます。



ヒント: 指定したポートでキャプチャされたデータ サンプルのみが返されます。

値は標準出力にリストされますが、その後の処理用に Tcl 変数またはファイルに出力することもできます。

次に、指定の hw_probe からのデータ サンプルをリストする Tcl スクリプトの例を示します。

```
# Define a list of probes to get the data samples from
set probeList [get_hw_probes *AR*]
#Specify the radix for the return values
set_property DISPLAY_RADIX BINARY [get_hw_probes *AR*]
# Define a filename to write data to
set fileName C:/Data/probeData1.txt
# Open the specified file in write mode
set FH [open $fileName w]
# Write probe data for each probe
foreach x $probeList {
    puts $FH "$x:"
```

```
    puts $FH [list_hw_samples $x]
}
# Close the output file
close $FH
puts "Probe data written to $fileName\n"
```

このコマンドを実行すると、データ サンプルが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_probe>: データ サンプルをリストする現在の hw_ila 上の hw_probe オブジェクトのリストを指定します。プローブは、`get_hw_probes` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したプローブのデータ サンプルが返されます。

```
list_hw_samples [get_hw_probes *probe18]
```

関連項目

- [current_hw_ila](#)
- [get_hw_ilas](#)
- [get_hw_probes](#)
- [create_hw_probe](#)

list_param

すべてのパラメーター名を取得します。

構文

```
list_param [-quiet] [-verbose]
```

戻り値

list

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

ユーザー定義可能なパラメーターのリストを取得します。これらのパラメーターでは、さまざまなツールの設定と動作が指定されます。特定のパラメーターの詳細を確認するには、`report_param` コマンドを実行し、パラメーターの説明とそのパラメーターの現在の値を返します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ユーザー定義可能なパラメーターすべてのリストが表示されます。

```
list_param
```

関連項目

- [get_param](#)
- [report_param](#)
- [reset_param](#)
- [set_param](#)

list_property

オブジェクトのプロパティをリストします。

構文

```
list_property [-class <arg>] [-regexp] [-quiet] [-verbose] [<object>]  
               [<pattern>]
```

戻り値

プロパティ名のリスト

使用法

名前	説明
[-class]	プロパティを取得するオブジェクト タイプを指定します。オブジェクトを指定した場合は無視されます。
[-regexp]	検索パターンを正規表現として処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<object>]	プロパティを取得するオブジェクトを指定します。
[<pattern>]	プロパティを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

指定したオブジェクトまたはクラスのプロパティすべてのリストを取得します。

注記: `report_property` コマンドでもオブジェクトまたはオブジェクトのクラスのプロパティ リストが返されますが、プロパティのタイプと値も含まれます。

引数

`-class <arg>` (オプション): 指定したオブジェクトのクラスのプロパティを返します。大文字/小文字が区別されます。ほとんどのクラス名は小文字で記述されます。

注記: `-class` オプションを `<object>` と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<object>` (オプション): プロパティをレポートするオブジェクトを 1 つ指定します。

注記: 複数のオブジェクトを指定すると、エラー メッセージが表示されます。

`<pattern>` (オプション): 指定した `<object>` のプロパティまたは `-class` オプションで指定したオブジェクト クラスのプロパティを検索するパターンを指定します。プロパティ名が指定の検索パターンに一致するプロパティのみが返されます。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトのプロパティすべてのリストが取得されます。

注記: 大文字/小文字が区別されます。ほとんどのプロパティ名は大文字で記述されます。

例

次の例では、指定したセル オブジェクトのプロパティすべてが返されます。

```
list_property [get_cells cpuEngine]
```

次の例では、BEL オブジェクト クラスのプロパティで指定した検索パターンに一致するプロパティが返されます。

```
list_property -class bel *NUM*
```

関連項目

- [create_property](#)
- [get_cells](#)
- [get_property](#)
- [list_property_value](#)
- [report_property](#)
- [reset_property](#)
- [set_property](#)

list_property_value

オブジェクトの有効なプロパティ値をリストします。

構文

```
list_property_value [-default] [-class <arg>] [-quiet] [-verbose] <name>
[<object>]
```

戻り値

プロパティ値のリスト

使用法

名前	説明
<code>[-default]</code>	デフォルト値のみを表示します。
<code>[-class]</code>	有効なプロパティ値を取得するオブジェクトタイプを指定します。オブジェクトを指定した場合は無視されます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	有効な値を取得するプロパティの名前を指定します。
<code>[<object>]</code>	有効なプロパティ値を取得するオブジェクトを指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

指定のオブジェクトのクラスまたは指定のオブジェクトの列挙型プロパティに有効な値をリストします。

注記: このコマンドでは、列挙型プロパティ以外のプロパティの有効な値はリストされません。report_property コマンドを実行すると、プロパティ タイプが返されるので、列挙型プロパティを特定できます。

引数

`-default` (オプション): 指定したオブジェクトのクラスのデフォルトのプロパティ値を返します。

`-class <arg>` (オプション): 指定したオブジェクトのクラスのプロパティ値を返します。大文字/小文字が区別されます。ほとんどのクラス名は小文字で記述されます。

注記: `-class` オプションを `<object>` と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 値を取得するプロパティの名前を指定します。このコマンドでは、列挙型の値、または定義済みの値のセットを持つプロパティのみを指定できます。指定したプロパティの有効な値すべてが返されます。

`<object>` (オプション): プロパティをレポートするオブジェクトを 1 つ指定します。

注記: 複数のオブジェクトを指定すると、エラー メッセージが表示されます。

例

次の例では、セル オブジェクトから `KEEP_HIERARCHY` プロパティの有効な値のリストが返されます。

```
list_property_value KEEP_HIERARCHY -class cell
```

次の例でも同じリストが返されますが、セル クラスに代わりに実際のセル オブジェクトが指定されています。

```
list_property_value KEEP_HIERARCHY [get_cells cpuEngine]
```

次の例では、現在のデザインをデザイン クラスとして使用し、指定のプロパティのデフォルト値を取得しています。

```
list_property_value -default BITSTREAM.GENERAL.COMPRESS [current_design]
```

関連項目

- [create_property](#)
- [current_design](#)
- [get_cells](#)
- [get_property](#)
- [list_property](#)
- [report_property](#)
- [reset_property](#)
- [set_property](#)

list_targets

指定したソースのターゲットをリストします。

構文

```
list_targets [-quiet] [-verbose] <files>
```

戻り値

ターゲットのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	ターゲットをリストするソース ファイルを指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

指定の IP コア、DSP モジュール、または IP サブシステムに使用可能なターゲットをリストします。指定可能なファイル タイプは .xci、.xco、.mdl、.bd、.bxml です。

ターゲットを生成するには、`generate_targets` コマンドを使用します。

このコマンドを実行すると、使用可能なターゲットのリストが返されます。指定のファイル オブジェクトに使用可能なターゲットがない場合は、戻り値はありません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): ソース ファイルのリストを含むファイル オブジェクトを指定します。

注記: ファイル名を指定するのではなく、`get_files` コマンドを使用してファイル オブジェクトを指定します。

例

次の例では、デザインに含まれる DSP モジュールに使用可能なターゲットをリストしています。

```
list_targets [get_files *.mdl]
```

関連項目

- [create_bd_design](#)
- [create_sysgen](#)
- [generate_target](#)
- [get_files](#)
- [import_ip](#)
- [read_ip](#)

load_features

指定の機能の Tcl コマンドを読み込みます。

構文

```
load_features [-quiet] [-verbose] [<features>...]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<features>]</code>	読み込む機能を指定します。使用可能な機能のリストは、 <code>list_features</code> コマンドを使用すると取得できます。

カテゴリ

[Tools \(ツール\)](#)

説明

Vivado Design Suite の指定の機能をメモリに読み込みます。

Vivado Design Suite のメモリ使用量を削減するため、Tcl コマンドの機能グループがあり、それらの機能グループのコマンドは、その機能グループからのコマンドを使用するか、`load_features` コマンドを使用して機能を読み込むまで使用できません。

たとえば、`load_features simulator` コマンドを使用するか、`launch_xsim` を使用して Vivado シミュレータを起動すると、Vivado シミュレータ用のコマンドが読み込まれます。

`load_features` コマンドを使用すると、ツールの機能を実際に実行せずに、Vivado Design Suite のある機能に関連するすべての Tcl コマンドをリストし、それらのコマンドのヘルプ テキストを表示できます。

`list_features` コマンドを使用すると、読み込むことができる機能のリストを取得できます。機能のリストは、リリースによって異なります。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<features>: 読み込む機能を指定します。

例

次の例では、Vivado シミュレータ機能を読み込んでいます。

```
load_features simulator
```

次の例では、Vivado Design Suite で読み込み可能な機能をすべて読み込んでいます。

```
load_features [list_features]
```

関連項目

- [help](#)
- [list_features](#)

lock_design

デザインのネットリスト、配置、または配線を固定または固定解除します。固定/固定解除は、物理的に配置されているセルおよび配線済みネットにのみ適用されます。

構文

```
lock_design [-level <arg>] [-unlock] [-export] [-quiet] [-verbose] [<cell>]
```

使用法

名前	説明
[-level]	固定レベルおよび固定解除レベルを指定します。有効な値は logical、placement、および routing で、デフォルトは placement です。
[-unlock]	セルの固定を解除します。セルを指定しない場合は、デザイン全体の固定が解除されます。-level オプションを指定する必要があります。
[-export]	制約がエクスポート可能であることをマークします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<cell>]	固定するセルを指定します。セルを指定しない場合は、デザイン全体が固定されます。配置済みのセルおよび配線済みネットのみが固定されます。デフォルトは * です。

カテゴリ

Project (プロジェクト)

説明

このコマンドは、デザイン保持およびパーシャル リコンフィギュレーション用の階層デザインで使われます。これらのデザイン フローの詳細およびこのコマンドの使用方法は、『Vivado Design Suite ユーザー ガイド: 階層デザイン』(UG905) を参照してください。

lock_design コマンドは、デザインまたはデザインの指定のセルの配置および配線を固定します。read_checkpoint コマンドを使用して独立階層 (OOC) チェックポイントを読み込んだら、このモジュールの保持レベルを定義する必要があります。

このコマンドは、指定したロジックの IS_LOC_FIXED、IS_BEL_FIXED、および IS_ROUTE_FIXED プロパティを設定します。

引数

```
lock_design [-level <arg>] [-unlock] [-export] [-quiet] [-verbose] [<cell>]
```

-level <arg> (オプション): 現在のデザインでセルまたはデザインを保持するレベルを指定します。デフォルトでは、配置データが保持されます。有効な値は次のとおりです。

- `logical`: 論理デザインを保持します。配置または配線情報も使用されますが、ツールで結果を改善できる可能性がある場合は変更可能です。
- `placement`: 論理および配置デザインが保持されます。配線情報も使用されますが、ツールで結果を改善できる可能性がある場合は変更可能です。これがデフォルト設定です。
- `routing`: 論理、配置、および配線デザインが保持されます。内部配線は保持されますが、インターフェイス ネットは保持されません。配線を保持するには、OOC インプリメンテーション中に Pblock に `CONTAIN_ROUTING` プロパティを使用しておく必要があります。これにより、OOC インプリメンテーションが再利用される際に配線が競合することはありません。

`-unlock` (オプション): セルの固定を解除します。セルを指定しない場合は、デザイン全体の固定が解除されます。固定を解除するには、固定する場合と同様、`-level` オプションを指定する必要があります。



ヒント: `-level routing` を指定すると、デザインの論理、配置、および配線が固定されますが、`-unlock -level routing` を指定すると、配線データの固定のみが解除されます。デザインの配線、配置、およびロジックの固定を解除するには、`-unlock -level logical` を指定する必要があります。

`-export` (オプション): 配置および配線データを XDC ファイルとしてエクスポートできるようにします。固定されたデザインまたはセルの制約は、`write_xdc` コマンドを使用してエクスポートできます。デフォルトでは、固定されたデザインまたはセルの制約はエクスポートできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cell>` (オプション): デザインで固定するセルを指定します。セルを指定しない場合は、デザイン全体が固定されます。デフォルトでは、デザインのすべてのセルが固定されます。



ヒント: 指定のレベルで配置済みのセルおよび配線済みネットのみが固定されます。

例

次の例では、現在のデザインの指定したセルのネットリスト、配置、および配線データを固定しています。

```
lock_design -level routing [get_cells usbEngine*]
```

次の例では、現在のデザインの指定したセルの配線データのみの固定を解除し、ネットリストおよび配置データの固定はそのままにしています。

```
lock_design -unlock -level routing [get_cells usbEngine*]
```

次の例では、現在のデザインの指定したセルの配線、配置、およびネットリスト データの固定を解除しています。

```
lock_design -unlock -level logical [get_cells usbEngine*]
```

関連項目

- [read_checkpoint](#)
- [write_xdc](#)

log_saif

指定したワイヤ、信号、またはレジスタのスイッチング アクティビティを SAIF (Switching Activity Interchange Format) ファイルに記録します。

構文

```
log_saif [-quiet] [-verbose] <hdl_objects>...
```

戻り値

なし

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hdl_objects></code>	記録する HDL オブジェクトを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーション中に指定の HDL 信号のスイッチング アクティビティ レートを記述します。

SAIF (Switching Activity Interchange format) ファイルは、ヘッダー情報とデザインの指定の信号のトグル回数を含む ASCII ファイルです。信号が 0、1、X、または Z となる期間を指定するタイミング属性も含まれます。

log_saif コマンドを使用するには、open_saif コマンドを使用して現在のシミュレーションのスイッチング アクティビティ レートを記録する SAIF ファイルを開いておく必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

`<hdl_objects>`: スwitchング アクティビティを記録する HDL 信号の名前を指定します。

例

次の例では、現在のスコープのすべての信号のスイッチング アクティビティを記録しています。

```
log_saif [ get_objects ]
```

次の例では、スコープ /tb/UUT 内の c で始まる内部信号のみを SAIF に記録しています。

```
log_saif [get_objects -filter { type == internal_signal }/tb/UUT/c*]
```

関連項目

- [close_saif](#)
- [get_objects](#)
- [open_saif](#)

log_vcd

指定したワイヤ、信号、またはレジスタのシミュレーション出力を VCD (Value Change Dump) ファイルに記録します。

構文

```
log_vcd [-level <arg>] [-quiet] [-verbose] [<hdl_objects>...]
```

戻り値

なし

使用法

名前	説明
[-level]	記録する HDL スコープのレベル数を指定します。デフォルトは 0 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hdl_objects>]	記録する HDL オブジェクトを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

VCD (Value Change Dump) ファイルに記録する HDL オブジェクトを指定します。デザインによっては、シミュレーション結果が非常に大きくなります。log_vcd コマンドを使用すると、記録する内容を指定できます。このコマンドは、Verilog の \$dumpvars システム タスクと同様の操作を実行します。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event などがあります。

このコマンドでは、VCD ファイルに記録する HDL オブジェクトとデザイン階層のレベル数を指定します。実際のオブジェクトの値は、シミュレーションの特定の時間に checkpoint_vcd または flush_vcd コマンドを実行したときに VCD ファイルに記述されます。



重要: open_vcd コマンドを実行する前に、*_vcd コマンドを実行する必要があります。

戻り値はありません。

引数

`-level <arg>` (オプション): VCD ファイルに記述する HDL オブジェクトを検索するデザイン階層レベル数を指定します。デフォルト値は 0 で、`<hdl_objects>` で定義された階層レベルとその下のレベルすべてにある指定の HDL オブジェクトの値が記述されます。1 に設定すると、`<hdl_objects>` で指定した階層レベルの HDL オブジェクトの値が VCD ファイルに記述されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hdl_objects>` (オプション): VCD ファイルに値の変化を記録する HDL オブジェクトを指定します。階層レベルも指定します。たとえば `/tb/UUT/*` を指定した場合は、デザインの `/tb/UUT` レベル内のすべての HDL オブジェクトが指定されます。

例

次の例では、スコープ `/tb/UUT` に含まれるすべてのポートの値の変化を記録しています。

```
log_vcd [get_objects -filter { type == port } /tb/UUT/* ]
```

注記: `-levels` が指定されていないので、指定したレベルの下すべてのレベルで指定した検索パターンに一致するポートが検索されます。

次の例では、現在のスコープのすべてのオブジェクトを VCD ファイルに記録しています。

```
log_vcd *
log_vcd [ get_objects *]
```

次の例では、ルート スコープ `/tb/UUT` に含まれる内部信号で、名前が `C` で開始するものの値の変化を記録しています。

```
log_vcd [get_objects -filter { type == internal_signal } ./C*]
```

関連項目

- [checkpoint_vcd](#)
- [flush_vcd](#)
- [open_vcd](#)

log_wave

指定したワイヤ、信号、またはレジスタシミュレーション出力を Vivado シミュレーション波形ビューアーに記録します。add_wave とは異なり、波形ビューアー (波形設定) に波形オブジェクトは追加しません。出力の波形データベース (wdb) への記録をイネーブルにするだけです。

構文

```
log_wave [-recursive] [-r] [-verbose] [-v] [-quiet] <hdl_objects>...
```

使用法

名前	説明
[-recursive]	サブスコープでも検索を実行します。
[-r]	サブスコープでも検索を実行します。
[-verbose]	すべての警告を表示します。
[-v]	すべての警告を表示します。
[-quiet]	コマンド エラーを表示しません。
<hdl_objects>	トレースする HDL オブジェクトを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

Vivado シミュレーション波形ビューアーを使用して表示するため、指定した HDL オブジェクトのシミュレーションアクティビティを 波形データベース ファイル (.wdb) に記録します。

Vivado シミュレータでは、HDL オブジェクトはワイヤ、信号、レジスタなどの値を保持できるエンティティです。

add_wave とは異なり、このコマンドは波形設定に波形オブジェクトは追加しません。波形アクティビティを Vivado シミュレータ波形データベース (WDB) に記録できるようにするだけです。詳細は、『Vivado Design Suite ユーザー ガイド: ロジック シミュレーション』 (UG900) を参照してください。

このコマンドを実行しても、何も返されません。

引数

-recursive | -r (オプション): 指定の HDL オブジェクトとその子オブジェクトの波形アクティビティを記録します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hdl_objects> (必須): Vivado シミュレータの波形データベース ファイルに含める HDL オブジェクトを指定します。階層レベルも指定します。たとえば `/tb/UUT/*` を指定した場合は、デザインの `/tb/UUT` レベル内のすべての HDL オブジェクトが指定されます。

例

次の例では、指定の HDL オブジェクトの波形アクティビティを記録しています。

```
log_wave -r [get_objects /testbench/dut/*]
```

関連項目

- [get_objects](#)

ltrace

シミュレーションする HDL 文のファイル名と行番号の表示のオン/オフを切り替えます。

構文

```
ltrace [-quiet] [-verbose] <value>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<value>	有効な値は、on、true、yes、off、false、no です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレーション デバッグ用に行レベル トレースをイネーブルにします。

シミュレーション中、シミュレーション ソース ファイルと評価されている行番号が Tcl コンソールに返されます。



ヒント: ptrace コマンドを使用したプロセス トレースでは、行トレースよりも詳細な情報が得られます。

この機能をイネーブルにするには、次のように現在のシミュレーション オブジェクトの LINE_TRACING プロパティを設定します。

```
set_property LINE_TRACING on [current_sim]
```

このコマンドを実行すると、行トレースの設定が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<value> (必須): シミュレーション中の行トレースをイネーブル/ディスエーブルにします。イネーブルにする場合は true、ディスエーブルにする場合は false に設定します。

例

次の例では、行トレースをイネーブルにしています。

```
ltrace true
```

関連項目

- [current_sim](#)
- [ptrace](#)
- [set_property](#)

make_bd_intf_pins_external

インターフェイス ピンの外部ポートを作成します。セルが指定されている場合、未接続のインターフェイス ピンすべての外部インターフェイス ポートを作成します。

構文

```
make_bd_intf_pins_external [-quiet] [-verbose] <objects>...
```

戻り値

少なくとも 1 つのインターフェイス ポートが作成された場合は Pass

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	外部にするインターフェイス ピン/セルを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在のブロック デザインに外部インターフェイス ポートを作成し、選択したブロック インターフェイス ピンに接続します。bd_cell を引数として指定した場合、そのセルの接続されていないブロック インターフェイス ピンがすべて外部になります。作成された外部インターフェイス ポートのプロパティは、選択したブロック インターフェイス ピンのプロパティと同じになります。



重要: ブロック インターフェイス ピンのグループに対しては、ブロック インターフェイス ピンごとに 1 つの外部ポートが作成されます。

選択したブロック インターフェイス ピンが接続されていない場合、ブロック デザインの最上位で新しいインターフェイス ポートに接続されます。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は、-quiet が指定されていなければ TCL_ERROR が返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): `get_bd_intf_pins` コマンドを使用してブロック インターフェイス ピン オブジェクトを指定するか、`get_bd_cells` コマンドを使用してブロック デザイン セル オブジェクトを指定します。

例

次の例は、ブロック デザイン セル `axi_gpio_0` の指定したブロック インターフェイス ピンを選択し、外部インターフェイス ポートに接続します。

```
make_bd_intf_pins_external [get_bd_intf_pins axi_gpio_0/S_AXI]
```

次の例は、ブロック デザイン セル `axi_gpio_0` の接続されていないすべてブロック インターフェイス ピンを、個別に外部インターフェイス ポートに接続します。

```
make_bd_intf_pins_external [get_bd_cells axi_gpio_0]
```

関連項目

- [create_bd_cell](#)
- [create_bd_intf_port](#)
- [connect_bd_intf_net](#)

make_bd_pins_external

ピンの外部ポートを作成します。セルが指定されている場合、未接続のピンすべての外部ポートを作成します。

構文

```
make_bd_pins_external [-quiet] [-verbose] <objects>...
```

戻り値

少なくとも 1 つのポートが作成された場合は Pass

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	外部にするピン/セルを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在のブロック デザインに外部ポートを作成し、選択したブロック ピンに接続します。bd_cell を引数として指定した場合、そのセルの接続されていないブロック ピンがすべて外部になります。作成された外部ポートのプロパティは、選択したブロック ピンのプロパティと同じになります。



重要: ブロック ピンのグループに対しては、ブロック ピンごとに 1 つの外部ポートが作成されます。

選択したブロック ピンが接続されていない場合、ブロック デザインの最上位で新しいポートに接続されます。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は、-quiet が指定されていない場合は TCL_ERROR が返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<objects> (必須): `get_bd_pins` コマンドを使用してブロック ピン オブジェクトを指定するか、`get_bd_cells` コマンドを使用してブロック デザイン セル オブジェクトを指定します。

例

次の例は、ブロック デザイン セル `axi-gpio-0` のブロック ピンを選択し、外部ポートに接続します。

```
make_bd_pins_external [get_bd_pins axi-gpio-0/s_axi_aclk]
```

次の例は、ブロック デザイン セル `axi-gpio-0` の接続されていないすべてブロック ピンを、個別に外部ポートに接続します。

```
make_bd_pins_external [get_bd_cells axi-gpio-0]
```

関連項目

- [create_bd_cell](#)
- [create_bd_port](#)
- [connect_bd_net](#)

make_diff_pair_ports

2 つのポートから差動ペアを作成します。

構文

```
make_diff_pair_ports [-quiet] [-verbose] <ports>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><ports></code>	差動ペアにするポートを指定します。

カテゴリ

XDC、PinPlanning (ピン プランニング)

説明

既存の 2 つのポートから差動ペアを作成します。差動ペアを構成するためには、2 つのポートの方向、インターフェイス、およびその他のプロパティが一致している必要があります。そうでない場合は、エラーが返されます。



重要: ポートを差動ペアとして指定する前に、`create_port` コマンドを使用するか XDC ファイルを読み込んで 2 つのポートを作成しておく必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<ports>` (必須): 差動ペアを作成する 2 つのポート オブジェクトを指定します。差動入力 of 正の側のポートを最初に指定する必要があります。

例

次の例では、ポートを 2 つ指定して差動ペアを作成しています。

```
make_diff_pair_ports port_Pos1 port_Neg1
```

関連項目

- [create_interface](#)
- [create_port](#)
- [split_diff_pair_ports](#)

make_wrapper

指定したソースの HDL ラッパーを生成します。

構文

```
make_wrapper [-top] [-testbench] [-inst_template] [-fileset <arg>]
              [-import] [-force] [-quiet] [-verbose] <files>
```

使用法

名前	説明
[-top]	指定したソースの最上位ラッパーを作成します。
[-testbench]	指定したソースのテストベンチを作成します。
[-inst_template]	指定したソースのインスタンス化テンプレートを作成します。テンプレートはプロジェクトには追加されず、参照用のみ作成されます。
[-fileset]	ファイルセット名を指定します。
[-import]	生成されたラッパーをプロジェクトにインポートします。
[-force]	既存のソースを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	ラッパーを生成するソース ファイルを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[SysGen \(System Generator\)](#)

説明

サブデザインをプロジェクトにインスタンス化するための Verilog または VHDL ラッパーを作成します。

make_wrapper コマンドは、Vivado Design Suite の IP インテグレーターからのエンベデッド プロセッサ デザイン、IP インテグレーター ブロック デザイン、および System Generator または MathWorks MatLab で作成された DSP モジュールのラッパーを作成します。

ラッパーは、サブデザインをスタンドアロン デザインの最上位にしたり、既存のデザインにサブデザインをインスタンス化するために生成できます。また、System Generator サブデザインのシミュレーション テストベンチ用のラッパーも生成できます。

注記: ラッパーは、プロジェクトの TARGET_LANGUAGE プロパティの設定に応じて Verilog または VHDL で生成されます。

このコマンドを実行すると、ラッパーの生成に関する情報が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-top` (オプション): 指定したソースの最上位 Verilog または VHDL ラッパーを作成します。ラッパーにより、サブデザインがデザイン階層の最上位としてインスタンス化されます。

`-testbench` (オプション): 指定したサブデザインのシミュレーション テストベンチ テンプレートを作成します。このテンプレートには DUT モジュール インスタンス化は含まれますが、シミュレーションのスティミュラスは含まれません。



重要: このオプションは、System Generator 複合ファイルに対してのみ有効です。その他のソースの場合は、エラーが返されます。

`-inst_template` (オプション): 指定したソースのインスタンス化 テンプレートを作成します。テンプレートはプロジェクトには追加されず、参照用にのみ作成されます。インスタンス化 テンプレートは別の RTL ファイルにコピーして貼り付けると、その階層でモジュールのインスタンスを作成できます。

`-fileset` (オプション): プロジェクトにラッパーをインポートする際に、ラッパーをインポートするファイルセットを指定します。デフォルトでは、ラッパーは `sources_1` にインポートされます。

`-import` (オプション): ラッパー ファイルをプロジェクトにインポートし、適切なファイルセットに追加します。

`-force` (オプション): 既存のラッパー ファイルを上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): ラッパーの生成元となるファイルを指定します。`make_wrapper` コマンドでは、System Generator for DSP からの `.mdl` ファイル フォーマット、MathWorks MATLAB からの `.slx` フォーマット、および Vivado Design Suite の IP インテグレーターからの `.bd` ファイル フォーマットのみがサポートされます。

例

次の例では、指定の IP インテグレーター ブロック デザインを現在のプロジェクトのデザイン階層に統合するためのインスタンス化 テンプレートを作成しています。

```
make_wrapper -inst_template -fileset [get_filesets sources_1] \
-files [get_files C:/Data/design_1/design_1.bd]
```

関連項目

- [add_files](#)
- [create_bd_design](#)
- [create_sysgen](#)
- [generate_target](#)
- [get_filesets](#)
- [list_targets](#)

mark_objects

GUI でオブジェクトをマークします。

構文

```
mark_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-rgb]</code>	色を RGB で指定します。
<code>[-color]</code>	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	マークするオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

GUI で指定のオブジェクトをマークします。指定したオブジェクトの位置を見つけやすくするため、アイコン マークが配置されます。マークは、色オプションのいずれかで指定した色で表示されます。

オブジェクトのマークを解除するには、`unmark_objects` コマンドを使用します。

注記: 1 つの色オプションのみを使用してください。両方の色オプションを指定すると、`-rgb` の方が `-color` より優先されます。

引数

`-rgb <args>` (オプション): オブジェクトをマークする色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定し、{0 255 0} は緑を指定します。

`-color <arg>` (オプション): オブジェクトをマークする色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): マークするオブジェクトを 1 つ以上指定します。

例

次の例では、現在選択されているオブジェクトをマークする赤いアイコンを追加します。

```
mark_objects -color red [get_selected_objects]
```

関連項目

- [get_highlighted_objects](#)
- [get_marked_objects](#)
- [get_selected_objects](#)
- [highlight_objects](#)
- [select_objects](#)
- [unmark_objects](#)

modify_debug_ports

デバッグ コアへの配線済みプローブ接続を変更します。

構文

```
modify_debug_ports [-probes <args>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-probes]</code>	接続するプローブを指定します。各プローブ接続をデバッグ コア ピン、チャンネル インデックス、論理ネットで指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Debug (デバッグ)

説明

デバッグ コアの指定したポートへの配線済みプローブ接続を変更します。指定したデバッグ プローブへの接続リストを指定します。各接続は Tcl リストとして指定し、次の 3 つの要素をスペースで区切って波かっこ ({}) で囲んで指定します。

1. 接続するデバッグ コアの論理ピン。
2. 指定したプローブのチャンネル インデックス。
3. プローブする信号の論理ネット。

複数のプローブ接続をリストのリストとして指定できます。例に示すように、各接続も Tcl リストです。

このコマンドは、指定したプローブ ポートへの既存のネットの接続を必要に応じて解除し、プローブする各ネットを指定したプローブ ポートに接続し、変更された接続を自動的に配線します。このプロセスで接続が解除されたネットは、未接続のままになります。

引数

`-probes <args>` (必須): プローブ接続のリストを Tcl リストのリストとして指定します。各プローブ接続は、1) 接続するデバッグ コアの論理ピン、2) プローブのチャンネル インデックス、3) プローブする信号の論理ネットをスペースで区切って定義します。複数のプローブ接続は、次に示すようにリストのリストとして指定します。

```
[list \{probe1 channel1 net1} \{probe2 channel1 net2} \{probe2 channel2 net3} ]
```

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、3 つのプロブ接続を変更しています。

```
modify_debug_ports -probes [list {top/x_ila/probe0 0 top/inst_A/net_0} \
    {top/x_ila/probe1 1 top/inst_A/net_a} {top/x_ila/probe1 2 top/inst_A/
net_b}]
```



ヒント: `modify_debug_ports` コマンドにより、ポート プロブが 1 つの信号から別の信号に移動します。

関連項目

- [connect_debug_port](#)
- [create_debug_core](#)
- [create_debug_port](#)
- [disconnect_debug_port](#)
- [set_property](#)

move_bd_cells

セルを階層セルに移動します。セル間の接続は保持されます。これらのセルとその他のセルの接続は、階層セルをまたぐことにより保持されます。

構文

```
move_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<parent_cell>]  
               [<cells>...]
```

戻り値

正しく実行された場合は 0。

使用法

名前	説明
[-prefix]	セルに追加する接頭辞を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<parent_cell>]	親セルを指定します。
[<cells>]	セル名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

IP インテグレーター セルを、現在のサブシステム デザイン内の指定の階層モジュールに移動します。移動するセル間の接続は保持されます。移動するセルとその他の移動されないセルの間の接続は、IP インテグレーターで階層の境界をまたぐためにピンおよびポートを追加することにより自動的に保持されます。

IP サブシステム デザインのセルは、`copy_bd_objs` コマンドを使用して階層モジュールにコピー、および `group_bd_cells` コマンドを使用してグループ化して階層モジュールに追加することもできます。

このコマンドを実行すると、親セル (<parent_cell>) モジュールの名前が返されるか、正常に実行されなかった場合はエラー メッセージが返されます。

引数

-prefix <arg> (オプション): 階層モジュールに移動するセルに適用する接頭辞を指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<parent_cell>: セルの移動先となる階層モジュールの名前を指定します。

<cells> (オプション): 現在の IP サブシステム デザインから階層モジュールに移動するセルを指定します。セルの指定には、get_bd_cells コマンドを使用します。

例

次に例を示します。

```
move_bd_cells -prefix mod1_ /myModule1 [get_bd_cells /myAxiFifo_1]
/myModule1
```

関連項目

- [copy_bd_objs](#)

move_dashboard_gadget

プロジェクト サマリ ダッシュボードのガジェットを移動します。

構文

```
move_dashboard_gadget -name <arg> -row <arg> -col <arg> [-dashboard <arg>]
                        [-quiet] [-verbose]
```

使用法

名前	説明
-name	ガジェットの名前を指定します。
-row	ガジェットの移動先の行番号を指定します。
-col	ガジェットの移動先の列番号を指定します。
[-dashboard]	ガジェットを関連付けるダッシュボードを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

ダッシュボード ガジェットのダッシュボード内の位置を指定します。



ヒント: 現在のところ、プロジェクト サマリが唯一のダッシュボードなので、ガジェットはプロジェクト サマリ ダッシュボードに配置されます。

ダッシュボード内のガジェットの配置は、行と列で指定します。プロジェクト サマリ ダッシュボードには 2 つの列 (0 および 1) があり、行はすべてのガジェットを表示するのに必要なだけあります。0 または 1 以外の列を指定したり、現在定義されている行数 + 1 を超える行を指定すると、エラーが返されます。

引数

-name <arg> (必須): 移動するガジェットの名前を指定します。

-row <arg> (必須): プロジェクト サマリ ダッシュボード内でガジェットを配置する行を指定します。

-col <arg> (必須): プロジェクト サマリ ダッシュボード内でガジェットを配置する列を指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したガジェットを列 0、行 3 に移動しています。

```
move_dashboard_gadget -name utilization_1 -row 3 -col 0
```

関連項目

- [create_dashboard_gadget](#)
- [delete_dashboard_gadgets](#)
- [get_dashboard_gadgets](#)

move_files

ファイルを、元のプロパティを保持したまま、1 つのファイルセットから別のファイルセットに移動します。

構文

```
move_files [-fileset <arg>] [-of_objects <args>] [-quiet] [-verbose]  
[<files>...]
```

戻り値

移動されたファイルのリスト

使用法

名前	説明
[-fileset]	移動先のファイルセットの名前を指定します。
[-of_objects]	ファイルの移動先のリコンフィギャラブル モジュールを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<files>]	移動するファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

get_files コマンドで返されたファイルを 1 つのファイルセットから別のファイルセットに移動します。

このコマンドを実行すると、移動されたファイルのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-fileset <arg> (オプション): 指定したソース ファイルの移動先となるファイルセットを指定します。ファイルセットを指定しない場合は、デフォルトで sources_1 ファイルセットに移動されます。指定したファイルセットが存在しない場合は、エラーが返されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<files> (オプション): `get_files` コマンドを使用して 1 つまたは複数のファイルを指定します。



重要: ファイルは `get_files` コマンドで指定する必要があります。名前で指定することはできません。

例

次の例では、`top_full.xdc` ファイルを `constrs_2` ファイルセットに移動しています。

```
move_files -fileset constrs_2 [get_files top_full.xdc]
```

関連項目

- [get_files](#)

move_wave

波形オブジェクトを波形設定の現在の位置から指定の位置に移動します。

構文

```
move_wave [-into <args>] [-at_wave <args>] [-after_wave <args>]  
          [-before_wave <args>] [-quiet] [-verbose] <items>...
```

使用法

名前	説明
[-into]	波形オブジェクトを移動する波形設定、グループ、または仮想バスを指定します。
[-at_wave]	指定の波形オブジェクト、あるいはグループまたは仮想表示しませんでない場合は指定の波形オブジェクトの後に新しい波形オブジェクトを挿入します。
[-after_wave]	指定の波形オブジェクトの後に新しい波形オブジェクトを挿入します。
[-before_wave]	指定の波形オブジェクトの前に新しい波形オブジェクトを挿入します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<items>	移動する波形オブジェクトを指定します。

カテゴリ

[Waveform \(波形\)](#)

open_bd_design

ディスク ファイルから既存の IP サブシステム デザインを開きます。

構文

```
open_bd_design [-quiet] [-verbose] <name>
```

戻り値

デザイン オブジェクト、コマンドが正しく実行されなかった場合はなし。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	開く IP サブシステム デザインの名前を指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

Vivado IDE の IP インテグレーターの IP サブシステムを開きます。IP サブシステムは、`create_bd_design` コマンドを使用して作成されている必要があります。

このコマンドを実行すると、開いた IP サブシステム デザインの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: Vivado Design Suite の IP インテグレーターで開く IP サブシステム デザインのパスとファイル名を指定します。ファイル名にはファイル拡張子を含める必要があります。

例

次の例では、現在のプロジェクトの指定した IP サブシステム デザインを開いています。

```
open_bd_design C:/Data/project1/project1.src/sources_1/bd/design_1/  
design_1.bd
```

関連項目

- [close_bd_design](#)
- [create_bd_design](#)
- [current_bd_design](#)
- [save_bd_design](#)

open_checkpoint

新しいプロジェクトのデザイン チェックポイントを開きます。

構文

```
open_checkpoint [-part <arg>] [-ignore_timing] [-quiet] [-verbose] <file>
```

使用法

名前	説明
<code>[-part]</code>	チェックポイント パーツを変更します。チェックポイントに XDEF が含まれている場合、エラーが発生することがあります。
<code>[-ignore_timing]</code>	チェックポイントをタイミング制約なしで読み込みます。-time オプションが指定されている場合は無視されます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	デザイン チェックポイント ファイルを指定します。

カテゴリ

Project (プロジェクト)

説明

デザイン チェックポイント ファイル (DCP) を開き、新しいインメモリ プロジェクトを作成して、新しいプロジェクトをチェックポイントの内容で初期化します。このコマンドでは、最上位デザイン チェックポイントまたは独立階層 (OOC) モジュールに対して作成されたチェックポイントを開くことができます。

チェックポイントを開く場合、あらかじめプロジェクトを作成する必要はありません。open_checkpoint コマンドは、デザイン データをメモリに読み込み、デザインを非プロジェクト モードで開きます。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

注記: Vivado ツールで複数のデザイン チェックポイントを開いている場合、current_project コマンドを使用してデザインを切り替える必要があります。current_design を使用して、どのチェックポイントがアクティブ デザインかを確認できます。

引数

-part <arg> (オプション): インポートしたチェックポイント デザインのターゲット パーツを指定します。このオプションを使用すると、デザイン チェックポイント ファイルで使用されているパーツのスピード グレードを変更したり、同じデバイスおよびパッケージの早期に使用可能だったパーツをプロダクション パーツに変更したりできます。



重要: -part オプションでは、使用できるパーツの範囲は限られ、互換性のないパーツを指定してチェックポイントを開こうとするとエラーが返されます。

-ignore_timing (オプション): チェックポイントをすばやく開けるように、タイミング制約なしで開きます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): チェックポイント ファイルのパスとファイル名を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定したチェックポイント ファイルを開き、デザインのターゲット パーツを指定しています。

```
open_checkpoint C:/Data/state1/checkpoint.dcp -part xc7k325tffg900-2
```

注記: 指定したパーツが指定のチェックポイントで使用されているデバイスおよびパッケージと互換していない場合は、エラーが返されます。

関連項目

- [current_design](#)
- [current_project](#)
- [read_checkpoint](#)
- [write_checkpoint](#)

open_example_project

指定した IP のサンプル プロジェクトを開きます。

構文

```
open_example_project [-dir <arg>] [-force] [-in_process] [-quiet]
                    [-verbose] <objects>...
```

戻り値

開いたプロジェクト

使用法

名前	説明
[-dir]	サンプル プロジェクトを作成するディレクトリへのパスを指定します。
[-force]	サンプル プロジェクトが既に存在する場合に上書きします。
[-in_process]	同じプロセス内にサンプル プロジェクトを開きます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	サンプル プロジェクトを開くオブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)

説明

指定した IP コアのサンプル プロジェクトを開きます。サンプル プロジェクトを使用すると、現在のプロジェクトに統合せずに、スタンドアロン プロジェクトで IP コアの機能を試すことができます。

引数

-dir <arg> (オプション): サンプル プロジェクトを保存するディレクトリへのパスを指定します。

-force (オプション): サンプル プロジェクトを新しく開き、指定のパスに存在しているサンプル プロジェクトを上書きします。

-in_process (オプション): 現在のプロジェクトと同じツール プロセス内にサンプル プロジェクトを開きます。このオプションを指定しない場合 (デフォルト)、サンプル プロジェクトの新しいプロセス インスタンスが開きます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): サンプル プロジェクトを開く IP コアを指定します。

例

次の例では、IP のカスタマイズをコピーして指定の IP コアのサンプル プロジェクトを新しい場所で開きます。

```
open_example_project -dir C:/Data/examples -force [get_ips blk_mem*]
```

関連項目

- [create_ip](#)
- [generate_target](#)
- [get_ips](#)
- [import_ip](#)

open_hw_manager

ハードウェア ツールを開きます。

構文

```
open_hw_manager [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite のハードウェア マネージャーを Vivado IDE または Tcl で、あるいはバッチ モードで開きます。デザインをサイリンクス FPGA ハードウェアにプログラムおよびデバッグするには、まずハードウェア マネージャーを開きます。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

ハードウェア マネージャーは、ボード上の FPGA デバイスにアクセスするための Vivado Design Suite の機能です。ハードウェア マネージャーの機能は、次のとおりです。

- ロジックのデバッグまたはロジック解析: FPGA プログラマブル ロジック デザインをデバッグします。
- プログラム/コンフィギュレーション: FPGA に接続されている JTAG およびコンフィギュレーション フラッシュ メモリ デバイスを使用して FPGA デバイスをプログラムします。
- インシステム シリアル I/O デバッグ: SERDES 受信/送信設定を調整し、送信ビット エラー レートを計測します。
- システム モニター: オンチップシステム モニターを制御し、システム モニターの温度および電圧値を読み取ります。

ハードウェア マネージャーでは、ハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、ハードウェア デバイス (hw_device)、ハードウェア ILA (hw_ila) など、多数のファースト クラス オブジェクトが使用されます。これらの各オブジェクトはほかのオブジェクトに関連しており、ハードウェア マネージャーでの機能を設定または制御するために `set_property` および `get_property` コマンドを使用して設定および取得可能なプロパティがあります。これらのオブジェクトの詳細は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) を参照してください。

ハードウェアに接続し、ターゲット FPGA をプログラムする手順は、次のとおりです。

- IDE でハードウェア マネージャーを開きます (`open_hw_manager`)。



ヒント: バッチ モードまたは Tcl モードを実行している場合は、この手順はスキップできます。

2. ローカル マシンまたはリモート ネットワークのアクセス可能なホストで実行中のハードウェア サーバーに接続します (`connect_hw_server`)。
3. 接続されているハードウェア サーバー上のハードウェア ターゲットを開きます (`open_hw_target`)。
4. 開いているハードウェア ターゲットでサイリンクス FPGA を特定します (`current_hw_device`、`get_hw_devices`)。
5. ビットストリーム データ プログラム ファイル (`.bit`) および存在する場合はプローブ ファイル (`.ltx`) を適切な FPGA に関連付けます (`set_property`)。
6. プログラム ファイルをハードウェア デバイスにプログラム (ダウンロード) します (`program_hw_device`、`refresh_hw_device`)。

ハードウェア マネージャーは、プロジェクトまたはデザインを開かずに Vivado ツール内から実行できます。ハードウェア マネージャーを開き、ハードウェア サーバーに接続して、ビットストリーム ファイルおよびデバッグ用にプローブ ファイルを指定してターゲット上のデバイスをプログラムできます。

ハードウェア マネージャーを閉じるには、`close_hw_manager` コマンドを使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、Vivado Design Suite でハードウェア マネージャーを開きます。

```
open_hw_manager
```

関連項目

- [close_hw_manager](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [current_hw_server](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)
- [refresh_hw_device](#)
- [set_property](#)

open_hw_platform

ザイリンクス シェル アーカイブを開きます。

構文

```
open_hw_platform [-auto_upgrade] [-quiet] [-verbose] [<file>]
```

戻り値

シェル ファイル名

使用法

名前	説明
<code>[-auto_upgrade]</code>	BD を自動的にアップグレードします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<file>]</code>	ザイリンクス シェル アーカイブ ファイルを指定します。英数字を使用した名前に拡張子 <code>.dsa/.xsa</code> を付けて指定します。

カテゴリ

[FileIO](#) (ファイル入力および出力)、[Platform](#) (プラットフォーム)、[Vitis](#)

説明

ザイリンクス サポート アーカイブ (XSA) ファイルを開き、アーカイブから Vivado プロジェクト、ブロック デザイン、および IP を抽出します。XSA からプロジェクト ディレクトリおよびプロジェクト ファイル (`.xpr`) を作成します。

注記: プロジェクトは現在の作業ディレクトリまたは Vivado ツールを起動したディレクトリに作成されます。

このコマンドを実行すると、実行されたアクションが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-auto_upgrade` (オプション): ブロック デザインで使用されている IP を最新バージョンにアップグレードします。デフォルトでは、XSA からの元の IP が使用され、最新でない場合はそのことがレポートされます。最新でない IP を特定するには、`report_ip_status` コマンドを使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): XSA ファイルのパスとファイル名を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した XSA を開き、IP を必要に応じて自動的にアップグレードしています。

```
open_hw_platform -auto_upgrade C:/Data/zc706.xsa
```

関連項目

- [validate_hw_platform](#)
- [write_hw_platform](#)

open_hw_target

ハードウェア サーバー上のハードウェア ターゲットへの接続を開きます。

構文

```
open_hw_target [-jtag_mode <arg>] [-xvc_url <arg>] [-auto_calibrate]
               [-quiet] [-verbose] [<hw_target>]
```

使用法

名前	説明
<code>[-jtag_mode]</code>	ターゲットを JTAG モードで開きます。
<code>[-xvc_url]</code>	XVC サーバーへのターゲット接続を開きます。
<code>[-auto_calibrate]</code>	最適な周波数を達成するためターゲットを自動キャリブレーションします (SmartLynq ケーブルのみ)。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_target>]</code>	ハードウェア ターゲットを指定します。デフォルトは、現在のハードウェア ターゲットです。

カテゴリ

Hardware (ハードウェア)

説明

接続されているハードウェア サーバーで、指定のハードウェア ターゲットまたは現在のハードウェア ターゲットへの接続を開きます。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のザイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、`connect_hw_server` コマンドを使用して開き、ザイリンクス ハードウェア サーバー アプリケーションで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

各ハードウェア ターゲットには、プログラムまたはデバッグ目的で使用するザイリンクス デバイスを 1 つまたは複数含めることができます。`current_hw_device` コマンドは、現在のデバイスを指定するか、返します。

使用可能なハードウェア ターゲットのいずれかへの接続を開くには、`open_hw_target` コマンドを使用します。開いたターゲットは、自動的に現在のハードウェア ターゲットとなります。現在のハードウェア ターゲットを `current_hw_target` コマンドを使用して定義し、そのターゲットを開くこともできます。プログラムおよびデバッグ コマンドは、ハードウェア サーバー接続を介して開いているターゲットに対して実行されます。

開いているハードウェア ターゲットへの接続を閉じるには、`close_hw_target` コマンドを使用します。

`open_hw_target` コマンドを実行すると、ハードウェア サーバーからの接続メッセージが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-jtag_mode [on | off]` (オプション): ハードウェア ターゲットをバウンダリスキャン アクセス用に JTAG モードで開きます。ターゲットを JTAG モードで実行すると、命令レジスタ (IR) とデータ レジスタ (DR) に `scan_ir_hw_jtag` および `scan_dr_hw_jtag` コマンドを使用してアクセスできるようになり、ターゲット上のデバイスを `run_state_hw_jtag` コマンドを使用してさまざまなステートにすることができます。

注記: これはハードウェア ターゲット (`hw_target`) オブジェクトのブール プロパティで、1 または True に設定してイネーブルにします。

`-xvc_url arg` (オプション): ハードウェア ターゲットをザイリンクス仮想ケーブルへの接続として開きます。ザイリンクス仮想ケーブル (XVC) は JTAG ケーブルのように動作する TCP/IP ベースのプロトコルで、物理的なケーブルを使用せずにザイリンクス デバイスにアクセスし、デバッグできます。引数としては、`xvcServer` アプリケーションの IP アドレスおよびポート番号を指定します。

注記: 詳細は、<https://japan.xilinx.com/products/intellectual-property/xvc.html> を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_target>` (オプション): プログラムおよびデバッグ用に接続を開く `hw_target` オブジェクトを指定します。`hw_target` は、`get_hw_targets` または `current_hw_target` コマンドを使用してオブジェクトとして指定する必要があります。ターゲットを指定しない場合は、現在のハードウェア ターゲット (`current_hw_target`) への接続が開きます。

例

次の例では、`get_hw_targets` コマンドで返されたハードウェア ターゲットを開いています。

```
open_hw_target [lindex [get_hw_targets] 0]
```

次の例では、ハードウェア マネージャーを開き、リモート ホストで実行中の `hw_server` アプリケーションに接続して、現在のハードウェア ターゲットを設定し、JTAG モードをイネーブルにしてそのターゲットを開くフローを示します。

```
open_hw
connect_hw_server -url jupiter:3121
current_hw_target [get_hw_targets */xilinx-tcf/Digilent/210203327463A]
open_hw_target -jtag_mode on
```

関連項目

- [close_hw_target](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)

- [get_hw_targets](#)

open_io_design

I/O ピン プランニング デザインを開きます。

構文

```
open_io_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-quiet]
               [-verbose]
```

戻り値

デザイン オブジェクト

使用法

名前	説明
[-name]	デザイン名
[-part]	ターゲット パーツを指定します。
[-constrset]	使用する制約ファイルセットを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

新規または既存の I/O ピン プランニング デザインを開きます。

注記: I/O ピン プランニング デザインを開くには、現在のソース ファイルセットの design_mode プロパティを PinPlanning に設定しておく必要があります。GateLvl に設定していないと、次のようなエラー メッセージが表示されます。

```
ERROR: The design mode of 'sources_1' must be PinPlanning
```

引数

-name <arg> (オプション): 新規または既存の I/O ピン プランニング デザインの名前を指定します。

-part <arg> (オプション): 新規デザインを作成する際に使用するザイリンクス デバイスを指定します。-part オプションを指定しない場合、デフォルトのパーツが使用されます。

-constrset <arg> (オプション): I/O ピン プランニング デザインを開く際に使用する制約ファイルセットの名前を指定します。

注記: -constrset オプションには、既存の制約ファイルセットを指定する必要があります。新しいファイルセットを作成することはできません。新しいファイルセットを作成する場合は、create_fileset コマンドを使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、myIO という I/O ピン プランニング デザインが新規に作成されます。

```
open_io_design -name myIO
```

注記: この場合、デフォルトのソース セット、制約セット、パーツが使用されます。

次の例では、myIO という既存の I/O デザインを開いて、使用する制約セットを指定しています。

```
open_io_design -name myIO -constrset topCon
```

関連項目

- [create_project](#)

open_project

Vivado プロジェクト ファイル (.xpr) を開きます。

構文

```
open_project [-part <arg>] [-read_only] [-quiet] [-verbose] <file>
```

戻り値

開いたプロジェクト オブジェクト

使用法

名前	説明
[-part]	プロジェクトを指定のパーツを使用して開きます。プロジェクトに設定されているパーツが置き換えられます。
[-read_only]	プロジェクトを読み出し専用モードで開きます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	開くプロジェクト ファイルを指定します。

カテゴリ

Project (プロジェクト)

説明

Vivado Design Suite プロジェクト ファイル (.xpr) または Vivado Lab Edition 用のプロジェクト ファイル (.lpr) を開きます。



重要: Vivado Lab Edition では open_project コマンドの構文は異なり、オプションの数は少なくなります。Vivado Lab Edition プロジェクト (.lpr) ではターゲット パーツは指定されないため、-part オプションはサポートされません。ターゲット パーツは、current_hw_target および current_hw_device コマンドで判断されます。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスと作成されたプロジェクトの名前が返され、正常に実行されなかった場合はエラーが返されます。

引数

-part <arg> (オプション): プロジェクトを開くときに使用するターゲット パーツを指定します。このオプションを使用すると、保存されているプロジェクトで使用されているパーツのスピード グレードを変更したり、同じデバイスおよびパッケージの早期に使用可能だったパーツをプロダクション パーツに変更したりできます。このオプションは Vivado Lab Edition ではサポートされていません。



重要: -part オプションでは、使用できるパーツの範囲は限られ、互換性のないパーツを指定してプロジェクトを開こうとするとエラーが返されます。

`-read_only` (オプション): プロジェクトを読み出し専用モードで開きます。プロジェクトへの変更は保存できません。変更を保存する必要がある場合は、`save_project_as` コマンドを使用して新しい編集可能なプロジェクトとして保存します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 開くプロジェクト ファイルを指定します。ファイルへのパスと `.xpr` ファイル拡張子を含める必要があります。

例

次の例では、Designs ディレクトリの `my_project1` という名前のプロジェクトを開きます。

```
open_project C:/Designs/project1.xpr
```

注記: プロジェクトに拡張子 `.xpr` を付けて指定しないと、ツールでファイルがプロジェクト ファイルとして認識されません。プロジェクト ファイル名にファイルへのパスも含めておかないと、指定したファイルが見つからないことを示すエラー メッセージが表示されます。

関連項目

- [close_project](#)
- [create_project](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [current_project](#)

open_report

.rpx ファイルからレポートを開きます。

構文

```
open_report [-file <arg>] [-append] [-console] [-name <arg>]
            [-return_string] [-quiet] [-verbose] <rpx>
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-console]	出力をコンソールに表示します。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<rpx>	読み込むレポート データ ファイルを指定します。

カテゴリ

Report (レポート)

説明

RPX (protobuf) ファイルをメモリに読み込み、Vivado Design Suite にレポートの結果を読み込み直します。このコマンドを実行するには、合成済みデザインまたはインプリメント済みデザインを開いている必要があります。

RPX ファイルは、report_timing_summary および report_pulse_width など、-rpx オプションをサポートするコマンドを使用して記述するインタラクティブなレポート ファイルで、メモリに読み込み直すことができます。レポートをメモリに読み込み直すと、レポートのオブジェクトがデザイン オブジェクトに再接続され、Vivado IDE のレポート間で相互選択されるようになり、デザインがイネーブルになります。

このコマンドを実行すると、デフォルトまたは -console オプションが指定されている場合はレポート結果が Tcl コンソールに表示され、-name オプションが指定されている場合は Vivado IDE でレポート ウィンドウが開きます。正常に実行されなかった場合はエラーが返されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。-append オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-console` (オプション): レポート結果を Vivado IDE の [Tcl Console] ウィンドウまたは Tcl シェルに出力します。ほかのオプションを指定しない場合、これが `open_report` コマンドのデフォルトです。

`-name <arg>` (オプション): GUI で表示する場合の結果の名前を指定します。GUI のタイミング サマリ レポートは、`delete_timing_results` コマンドを使用して削除できます。



ヒント: RPX ファイルを Vivado IDE の指定した名前のウィンドウで開くと、レポートとデザインのオブジェクトがリンクされ、相互選択されるようになります。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<rpx>`: メモリに読み込む RPX ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した RPX ファイルを読み込んで、Vivado IDE で開いています。

```
open_report -name RPX1 design1_summary.rpx
```

関連項目

- [check_timing](#)
- [report_bus_skew](#)
- [report_config_timing](#)
- [report_datasheet](#)
- [report_drc](#)
- [report_methodology](#)
- [report_power](#)
- [report_pulse_width](#)
- [report_timing](#)
- [report_timing_summary](#)

open_run

ネットリストまたはインプリメンテーション デザインで run を開きます。

構文

```
open_run [-name <arg>] [-pr_config <arg>] [-quiet] [-verbose] <run>
```

戻り値

デザイン オブジェクト

使用法

名前	説明
[-name]	デザイン名
[-pr_config]	デザインを開いたときに適用する PR コンフィギュレーションを指定します。合成 run を開いている場合にのみ有効です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<run>	デザインで開く run を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

指定の合成 run をネットリスト デザインで開くか、インプリメンテーション run をインプリメンテーション デザインで開きます。ターゲット パーツと制約セットを定義する run プロパティを合成結果またはインプリメンテーション結果と組み合わせて、ツールにデザイン ビューが作成されます。

このコマンドは、Vivado Design Suite のプロジェクト モードでデザイン run から生成された合成済みデザインまたはインプリメント済みデザインを開くために使用します。

非プロジェクト モードのデザインを開くには、open_checkpoint コマンドを使用してチェックポイントをメモリに読み込みます。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

引数

-name (オプション): run を Vivado ツールで開いたときに合成済みまたはインプリメント済みデザインに割り当てる名前を指定します。この名前は参照用であり、デザインの最上位またはデザインに含まれるロジックとは関係ありません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<run>` (必須): 開く合成 run またはインプリメンテーション run の名前を指定します。run をデザインとして開く前に、合成またはインプリメンテーションを完了している必要があります。

注記: 実行されていない run を開こうとすると、エラー メッセージが表示されます。

例

次の例では、指定の合成 run をネットリスト デザインとして `synthPass1` という名前で開きます。

```
open_run -name synthPass1 synth_1
```

次の例では、`impl_1` というインプリメンテーション run を開いています。

```
open_run impl_1
```

関連項目

- [launch_runs](#)
- [link_design](#)
- [open_checkpoint](#)
- [read_checkpoint](#)
- [write_checkpoint](#)

open_saif

消費電力見積もり用に信号のスイッチング レートを保存するファイルを開きます。スイッチング レートは、SAIF (Switching Activity Interchange Format) 形式で記述されます。1 つのシミュレーション run に対して開くことができるのは 1 つの SAIF ファイルのみです。

構文

```
open_saif [-quiet] [-verbose] <file_name>
```

戻り値

開いた SAIF オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file_name></code>	情報を保存する SAIF ファイルの名前を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションにおいて、信号のスイッチング アクティビティを保存する SAIF (Switching Activity Interchange Format) ファイルを作成するか開きます。このファイルは、後で `report_power` コマンドで使用できます。

SAIF (Switching Activity Interchange format) ファイルは、ヘッダー情報とデザインの指定の信号のトグル回数を含む ASCII ファイルです。信号が 0、1、X、または Z となる期間を指定するタイミング属性も含まれます。

SAIF ファイルは VCD ファイルより小さいので、消費電力解析に推奨されます。

SAIF ファイルを開くと、`log_saif` コマンドを使用して、シミュレーションからのスイッチング アクティビティを SAIF ファイルに記述できます。

シミュレーション中、一度に開くことのできる SAIF ファイルは 1 つのみです。SAIF ファイルを閉じるには、`close_saif` コマンドを使用します。

このコマンドを実行すると、開いた SAIF ファイルのオブジェクト ID が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet`: コマンドをメッセージを表示せずに実行します。コマンドでエラーが発生した場合でもエラー メッセージは表示されません。

-verbose: メッセージの非表示設定を解除し、すべてのメッセージを表示します。

<file_name>: 開く SAIF ファイルの名前を指定します。

例

次の例では、指定の SAIF ファイルを開いています。

```
open_saif myData.saif
```

関連項目

- [close_saif](#)
- [log_saif](#)

open_vcd

シミュレーション出力を保存する VCD (Value Change Dump) ファイルを開きます。\$dumpfile Verilog システム タスクと同様の操作を実行します。

構文

```
open_vcd [-quiet] [-verbose] [<file_name>]
```

戻り値

VCD オブジェクト (このオブジェクトが現在の VCD)

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file_name>]	ファイル名を指定します。デフォルトは dump.vcd です (LRM 規格)。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレーション出力を保存する VCD (Value Change Dump) ファイルを作成するか開きます。このコマンドは、Verilog の \$dumpfile シミュレータ指示子と同様の操作を実行します。

VCD は、ヘッダー情報、変数定義、HDL 信号値の変更の詳細を含む ASCII 形式のファイルです。VCD ファイルを使用すると、VCD ビューアでシミュレーション結果を表示したり、デザインの消費電力見積もりを実行したりできます。

VCD ファイルを開くと、checkpoint_vcd、flush_vcd、または log_vcd コマンドを使用してシミュレーションでの値の変化を VCD ファイルに記述できます。また、stop_vcd および start_vcd コマンドを使用して、データの記述を一時停止したり再開したりできます。

VCD ファイルのサイズを制限するには、limit_vcd コマンドを使用します。

VCD ファイルを閉じるには、close_vcd コマンドを使用します。

注記: open_vcd コマンドを実行する前に、*_vcd コマンドを実行する必要があります。一度に開くことのできる VCD ファイルは 1 つのみです。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file_name>` (オプション): VCD 情報を保存するファイルの名前を指定します。ファイル名を指定しない場合、デフォルトのファイル名 `dump.vcd` が使用されます。指定の VCD ファイルが既に存在する場合は、その `open_vcd` ファイルがリセットされ、内容が上書きされます。

例

次の例では、指定の VCD ファイル (`design1.vcd`) を開き、値の変化を記述できるようにしています。`log_vcd` コマンドは `/tb/UUT` スコープのデザイン階層レベルのみにあるすべてのポートを特定し、VCD ファイルに記述します。シミュレーションは指定の時間実行され、`flush_vcd` コマンドで HDL オブジェクトの現在の値を VCD ファイルに記述します。その後、`close_vcd` コマンドで開いているファイルを閉じています。

```
open_vcd design1.vcd
log_vcd -level 1 [get_objects filter { type == port } /tb/UUT/* ]
run 1000
flush_vcd
close_vcd
```

関連項目

- [checkpoint_vcd](#)
- [close_vcd](#)
- [flush_vcd](#)
- [limit_vcd](#)
- [log_vcd](#)
- [start_vcd](#)
- [stop_vcd](#)

open_wave_config

波形設定を開きます。

構文

```
open_wave_config [-quiet] [-verbose] [<filename>]
```

戻り値

開いた波形設定

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<filename>]	新しい波形設定および対応する波形ウィンドウを作成する WCFG ファイルの名前を指定します。

カテゴリ

[Waveform \(波形\)](#)

説明

現在のシミュレーションの指定した波形設定ファイル (.wcfg) を開きます。

Vivado シミュレータでは、シミュレーション デバッグ データ モデルが使用され、ソース コードの 1 行ずつの実行、ブレークポイント、条件、および波形表示ツールを使用して HDL ソース ファイルをデバッグできます。デバッグ データ モデルには HDL オブジェクトおよびスコープ名が含まれ、それらをメモリ アドレスにマップすることにより、シミュレーション中の信号、変数、定数値の変化を調べることができます。

波形データベースは、シミュレーションの波形アクティビティを保存する波形設定ファイルとは異なります。波形設定ファイルには、単に表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。波形データベース (WDB) には、表示されるかどうかにかかわらず、トレースされる信号すべてのイベント データ、値の時間による変化が含まれます。

create_wave_config コマンドを使用すると、現在のシミュレーションで波形設定オブジェクトを作成できます。波形設定ファイルをディスクに保存するには save_wave_config コマンドを使用し、開くには open_wave_config コマンドを使用します。

open_wave_config コマンドは、波形設定ファイルを開き、現在のシミュレーションのデータ ソースにマップします。



重要: 波形設定ファイルで指定されている HDL オブジェクトで現在のシミュレーションに含まれないものは無視されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<filename>` (オプション): 開く波形設定ファイルのパスと名前を指定します。

ファイル名 (`<filename>`) の拡張子は `.wcfg` である必要があり、拡張子を付けない場合は自動的に付けられます。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した波形設定ファイルを開いています。

```
open_wave_config testbench.wcfg
```

関連項目

- [close_wave_config](#)
- [create_wave_config](#)
- [open_wave_database](#)
- [save_wave_config](#)

open_wave_database

以前のシミュレーションで生成された波形データベース (WDB) ファイルを開き、シミュレーション オブジェクトを返します。

構文

```
open_wave_database [-noautoloadwcfg] [-protoinst <args>] [-quiet]
                   [-verbose] <wdb>
```

使用法

名前	説明
<code>[-noautoloadwcfg]</code>	関連付けられている WCFG ファイルを自動的に開きません。
<code>[-protoinst]</code>	プロトコル解析の .protoinst ファイルを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><wdb></code>	ファイル名を指定します。

カテゴリ

Simulation (シミュレーション)

説明

`open_wave_database` コマンドは、既存のスタティック シミュレータ データベース ファイル (WDB) とそれに関連付けられている波形設定ファイル (WCFG) を開きます。シミュレーションは実行中のものでなくスタティック シミュレーションで、以前の結果を表示するためにのみ使用できます。

注記: シミュレーションを実行およびリセットする多くのコマンドは、スタティック シミュレーションでは使用できません。

Vivado シミュレータでは、シミュレーション デバッグ データ モデルが使用され、ソース コードの 1 行ずつの実行、ブレークポイント、条件、および波形表示ツールを使用して HDL ソース ファイルをデバッグできます。デバッグ データ モデルには HDL オブジェクトおよびスコープ名が含まれ、それらをメモリ アドレスにマップすることにより、シミュレーション中の信号、変数、定数値の変化を調べることができます。シミュレーションが完了すると、スタティック シミュレータ データベース ファイル (WDB) に記述されます。

`log_wave` コマンドを使用して、シミュレーション波形データベースに HDL オブジェクトを追加できます。このコマンドは、指定のオブジェクトの波形アクティビティを Vivado シミュレータ波形データベースに記録できるようにします。

波形データベースは、シミュレーションの波形アクティビティを保存する波形設定ファイルに関連付けられています。波形設定ファイルには、単に表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。波形データベース (WDB) には、表示されるかどうかにかかわらず、トレースされる信号すべてのイベント データ、値の時間による変化が含まれます。

波形設定ファイルをディスクに保存するには `save_wave_config` コマンドを使用し、開くには `open_wave_config` コマンドを使用します。

`open_wave_database` コマンドを `open_wave_config` コマンドと共に使用し、Vivado IDE で既に完了したシミュレーションを確認するために開きます。



ヒント: `log_wave` コマンドを使用してシミュレーション波形データベースに記録したオブジェクトは、`add_wave` コマンドを使用してスタティック シミュレーションの波形設定に追加できます。

引数

`-noautoloadwcfg` (オプション): 波形データベースに関連付けられている波形設定ファイル (WCFG) を自動的に開きません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<wdb>: 開く WDB ファイルのパスと名前を指定します。

例

つぎの例では、指定の名前の WDB ファイルを開き、関連付けられている波形設定ファイルを開いて、`current_fileset` コマンドを使用してシミュレーション データベースを Vivado IDE で開いています。

```
open_wave_database {C:/Data/project_xsim/testbench_behav.wdb}
open_wave_config {C:/Data/project_xsim/testbench_behav.wcfg}
current_fileset
```

関連項目

- [create_wave_config](#)
- [current_fileset](#)
- [open_wave_config](#)
- [save_wave_config](#)

opt_design

現在のネットリストを最適化します。デフォルトでは、リターゲット、定数伝搬、スイープ、およびブロック RAM の消費電力最適化が実行されます。

構文

```
opt_design [-retarget] [-propconst] [-sweep] [-bram_power_opt] [-remap]
  [-aggressive_remap] [-resynth_area] [-resynth_seq_area]
  [-directive <arg>] [-muxf_remap] [-hier_fanout_limit <arg>] [-bufg_opt]
  [-shift_register_opt] [-dsp_register_opt] [-srl_remap_modes <arg>]
  [-control_set_merge] [-merge_equivalent_drivers] [-carry_remap]
  [-debug_log] [-property_opt_only] [-quiet] [-verbose]
```

使用法

名前	説明
[-retarget]	リターゲット
[-propconst]	制約を最下位インスタンスに伝搬します。
[-sweep]	未接続の最下位インスタンスを削除します。
[-bram_power_opt]	ブロック RAM の消費電力最適化を実行します。
[-remap]	ロジックを LUT に最適にリマップします。
[-aggressive_remap]	リマップ最適化を高いエフォート レベルで実行します。
[-resynth_area]	再合成を実行します。
[-resynth_seq_area]	順序ロジックの最適化を含む再合成を実行します。
[-directive]	コマンドのモードを指定します。有効な値は、「引数」セクションを参照してください。デフォルトは Default です。
[-muxf_remap]	すべての MuxFx セルを LUT3 に最適化します。
[-hier_fanout_limit]	モジュールごとにファンアウトが指定した値より大きいドライバーを複製します。
[-bufg_opt]	BUFG を挿入、結合、および分離します。
[-shift_register_opt]	シフト レジスタからレジスタ段を取り出します。
[-dsp_register_opt]	レジスタを DSP 内に挿入するか、DSP から取り出すかを指定します。
[-srl_remap_modes]	シフト レジスタをフリップフロップに、またはフリップフロップをシフト レジスタにリマップします。
[-control_set_merge]	すべての等価制御セットドライバーを 1 つのドライバーに結合します。
[-merge_equivalent_drivers]	すべての LUT、フリップフロップ等価ドライバーを統合します。
[-carry_remap]	キャリーを LUT にリマップします。
[-debug_log]	デバッグ メッセージを表示します。
[-property_opt_only]	タグの付いたセルに対して最適化を実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Tools (ツール)

説明

デザイン ネットリストをターゲット パーツ用に最適化します。最適化では、サードパーティ ツールからの合成済み ネットリストや合成中に最適化されなかったネットリストを向上できます。

このコマンドは、合成後、インプリメンテーションでデザインを最適化する前に実行し、配置配線前にネットリストを簡略化します。セルの最適化が実行されないようにするには、DONT_TOUCH プロパティを設定します。



ヒント: `opt_design` コマンドで実行されたデザインの最適化を確認するには、`-verbose` オプションを使用してプロセスのより詳細なログを表示します。これは、デザインに加えられた変更を理解し、デバッグするのに役立ちます。

`opt_design` コマンドは、デフォルトで次の最適化を実行します。

- リターゲット
- 定数伝搬
- スイープ
- グローバル バッファ (BUFG) の最適化
- DSP レジスタの最適化
- シフト レジスタ ロジック最適化 (デフォルト)
- ブロック RAM 消費電力最適化
- MIG コアのインプリメント
- デバッグ コアのインプリメント



重要: `opt_design` コマンドで特定の最適化のコマンド ライン オプションを使用すると、その最適化のみが実行され、ほかの最適化は通常デフォルトで実行される最適化であってもディスエーブルになります。

LUT リマップを実行するには、`-remap` を指定する必要があります。

エリア ベースの再合成を実行するには、`-resynth_area` または `-directive ExploreArea` を指定する必要があります。

順序ロジック エリア ベースの再合成を実行するには、`-resynth_seq_area` または `-directive ExploreSequentialArea` を指定する必要があります。

引数

`-retarget` (オプション): デザインをあるデバイス ファミリから別のファミリに移行する際に、あるタイプのブロックを別のタイプのブロックに移行します。たとえば、インスタンス化された MUXCY または XORCY コンポーネントを CARRY4 ブロックに、DCM を MMCM に変更します。リターゲット最適化では、可能な場合に、インバーターのダウストリーム ロジックへの吸収も実行されます。

注記: `-retarget` の指定はオプションですが、ほかの最適化のコマンド ライン オプションを指定していなければ、この最適化はデフォルトで実行されます。

-propconst (オプション): 定数入力を回路で伝搬し、ネットリストを簡略化します。定数の伝搬により、ネットリストから余分な組み合わせロジックを削除できる可能性があります。

-sweep (オプション): ロードのないセルおよびネットを削除することにより、不要なロジックを削除します。

-bram_power_opt (オプション): ブロック RAM セルで消費電力の最適化をイネーブルにします。完全なデュアルポート RAM の読み出されないポートの WRITE_MODE を NO_CHANGE に変更し、ブロック RAM の出力にクロックゲーティングを適用します。

注記: set_power_opt コマンドを使用して、特定の BRAM セルをこの最適化から除外できます。

-remap (オプション): デザインをリマップし、複数の LUT を 1 つの LUT に組み合わせてロジックの段数を削減します。

-aggressive_remap (オプション): -remap オプションと同様に LUT のロジック レベル数を削減しますが、より徹底的に実行します。LUT のロジック レベル数は削減されますが、最適化の実行時間は長くなります。

-resynth_area (オプション): エリア モードで合成を再実行し、LUT の数を削減します。

-resynth_seq_area (オプション): 組み合わせロジックおよび順序ロジックを削減するよう再合成を実行します。-resynth_area の最適化のスーパーセットを実行します。

-directive <arg> (オプション): 最適化モード (デザインの目標) を指定します。1 つの opt_design コマンドに対して 1 つのモードのみを指定できます。値では大文字/小文字が区別されます。有効な値は次のとおりです。

- Explore: 結果を向上するために最適化を複数回実行します。
- ExploreArea: エリアを縮小することを優先して最適化を複数回実行します。
- ExploreWithRemap: ExploreArea と同様ですが、リマップ最適化を追加してロジック レベルを圧縮します。
- ExploreSequentialArea: レジスタおよび関連の組み合わせロジックを縮小することを優先して最適化を複数回実行します。
- AddRemap: デフォルトの最適化を実行し、LUT リマップを含めてロジック レベルを削減します。
- NoBramPowerOpt: opt_design コマンドで BRAM 消費電力の最適化を除外して実行します。
- RuntimeOptimized: 反復回数を少なくし、最適化の結果よりも実行時間を短縮することを優先します。
- Default: デフォルトの最適化を実行します。

各モードの効果は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

注記: -directive オプションは全体的な最適化ストラテジを制御するもので、特定の最適化オプションとは互換性がありません。-debug_log、-quiet、および -verbose オプションとのみ使用できます。

-muxf_remap (オプション): タイミングに影響を与えずにデザインの配線性を向上する可能性がある場合に、MUXF を LUT3 に変換します。

-hier_fanout_limit <arg> (オプション): ファンアウトが指定した値 (<arg>) より大きいネット ドライバーを論理階層に従って複製します。ファンアウトの大きいネットで駆動される各階層インスタンスでファンアウトが指定の上限を超えている場合、階層内のネットは複製されたドライバーで駆動されます。

-bufg_opt (オプション): グローバル バッファ (BUFG/BUFGCE) に関するさまざまな最適化を実行します。バッファのないクロック ネット (ファンアウト > 30) には、バッファを挿入します。ネット上の制御セット ピン数がツールのしきい値を超えるファンアウトの大きいネットには、BUFG を挿入します。ファンアウトの大きいネット ドライバーが組み合わせロジックおよび順序ロジックの両方を駆動する場合には、負荷を分割して追加遅延が大きすぎる場合に組み合わせ部分が BUFG をバイパスするようにします。



ヒント: `phys_opt_design` コマンドを使用して、順次部分を BUFG で駆動しながら組み合わせ部分を最適化できます。

`-shift_register_opt` (オプション): SRL からのファンアウトの大きいネットに出力レジスタ パイプライン段を追加し、ファンアウトの大きいネットの複製前にタイミングを向上します。この最適化は、デフォルト最適化の一部として実行されます。

`-dsp_register_opt` (オプション): レジスタを DSP から取り出すか、DSP に挿入して、タイミングおよびリソース使用率を向上します。この最適化は、デフォルト最適化の一部として実行されます。

`-srl_remap_modes <args>` (オプション): 多くのデザインには、深い SRL および長いレジスタ チェーンに関するタイミング クロージャ問題があります。このオプションは、フリップフロップ チェーンと SRL の間でリマップをトリガーする自動ルールと、フリップフロップ チェーンから SRL または SRL からフリップフロップ チェーンへの変換をいつ実行するかを指定する手動ルールを提供します。自動リマップは、フリップフロップの使用率を削減する必要がある場合に一番長いフリップフロップ チェーンから開始してそれらを SRL に変換し、LUT の使用率を削減する必要がある場合に浅い SRL をフリップフロップ変換します。手動リマップは、指定した長さを超えるフリップフロップ チェーンを SRL にリマップし、指定した深さの SRL をフリップフロップ チェーンにリマップします。1 回の `opt_design` の実行で自動モードと手動モードの両方を使用することはできません。自動ルールと手動ルールの構文は、次のとおりです。

- 自動:

- `-srl_remap_modes {{target_ff_util <util> target_lutram_util <util>}}: <util>` には、使用率のパーセントを 0 ~ 100 で指定します。フリップフロップまたは LUT の使用率が指定の値を超える場合、リマップが必要になります。両方の使用率が指定の値を超えている場合、リマップは実行されません。

- 手動:

- `-srl_remap_modes {{min_depth_ffs_to_srl <val>}{max_depth_srl_to_ffs <val>}}: <val>` には SRL の深さまたはフリップフロップ チェーンの長さを指定します。フリップフロップ チェーンが指定した最小値を超える場合、そのチェーンは必要な深さの SRL にリマップされます。SRL の深さが指定した最大値以下の場合、その SRL はフリップフロップ チェーンにリマップされます。

`-control_set_merge` (オプション): 論理的に等価の制御信号のドライバーを 1 つのドライバーにまとめます。これは、ファンアウト複製の逆のような操作で、ネットがモジュール ベース複製により適したものになります。

`-merge_equivalent_drivers` (オプション): 制御信号および制御信号でない信号両方の等価ドライバーを統合し、デザインのロジックを削減します。

`-carry_remap` (オプション): キャリー信号を LUT リマップします。

`-debug_log` (オプション): デバッグ用にログ ファイルを生成します。

`-property_opt_only` (オプション): `opt_design` コマンドを、コマンド構文の一部として指定したオプションではなく、デザインに含まれるオブジェクトのプロパティのみを使用して実行します。プロパティの詳細は、『Vivado Design Suite プロパティ リファレンス ガイド』 (UG912) を参照してください。プロパティのみのみの最適化をトリガーするプロパティは、`CARRY_REMAP`、`CONTROL_SET_REMAP`、`EQUIVALENT_DRIVER_OPT`、`LUT_REMAP`、`MUXF_REMAP`、`REG_TO_SRL`、`SRL_STAGES_TO_REG_INPUT`、`SRL_STAGES_TO_REG_OUTPUT`、`SRL_TO_REG` です。



ヒント: `-property_opt_only` を指定した場合、`opt_design` コマンドに追加の最適化が含まれているとエラーが返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。各最適化により変更されたロジックに関する詳細な情報が表示されます。

例

次の例では、4 つのデフォルトの最適化 (リターゲット、定数伝搬、スweep、および BRAM 消費電力最適化) すべてを実行しています。`-verbose` オプションが使用されているので、詳細な結果が返されます。

```
opt_design -verbose
```

次の例では、`set_power_opt` オプションを使用して特定の BRAM セルを消費電力最適化から除外し、`opt_design` コマンドでデフォルトの 4 つの最適化を実行しています。

```
set_power_opt -exclude_cells [get_cells \
    -filter {PRIMITIVE_TYPE =~ BMEM.*.*} \
    -of_objects [get_pins -leaf -filter {DIRECTION == IN} \
    -of_objects [get_nets -of_objects [get_pins clock/bufgctrl_clk_mld/
O]]]]
opt_design
```

次の例では、スweepおよびリターゲット最適化を実行しています。

```
opt_design -sweep -retarget
```

注記: この例では `-sweep` と `-retarget` が指定されているため、`-propconst` および `-bram_power_opt` 最適化はディスエーブルになります。

次の例では、よりよい結果が得られるように、`opt_design` コマンドでさまざまなアルゴリズムを使用しています。

```
opt_design -directive Explore
```

次の例では、エリアを縮小することを優先しながら、よりよい結果が得られるように `opt_design` コマンドでさまざまなアルゴリズムを使用しています。

```
opt_design -directive ExploreArea
```

次の例では、`-srl_remap_modes` オプションを使用して、フリップフロップの利用率が 20% を超える場合にフリップフロップ チェーンを SRL に変換し、SRL の使用率が 50% を超える場合に SRL をフリップフロップに変換する自動モードを設定しています。

```
-srl_remap_modes {{target_ff_util 20 target_lutram_util 50}}
```



ヒント: この場合、いずれかの使用率が指定の値を超えた場合にのみリマップが実行され、両方の使用率が指定の値を超えた場合は実行されません。

次の例では、`-srl_remap_modes` オプションを使用して、長さが 5 を超えるフリップフロップ チェーンを SRL に変換し、深さが 8 以下の SRL をフリップフロップに変換する手動モードを設定しています。

```
-srl_remap_modes {{min_depth_ffs_to_srl 5}{max_depth_srl_to_ffs 8}}
```

関連項目

- [phys_opt_design](#)
- [place_design](#)
- [power_opt_design](#)
- [route_design](#)
- [set_power_opt](#)
- [synth_design](#)

pause_hw_hbm_amon

指定したハードウェア HBM に対してアクティビティ モニターの実行を一時停止します。

構文

```
pause_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

`pause_hw_hbm_amon` コマンドは、Vivado ハードウェア マネージャーで `run_hw_hbm_amon` コマンドを使用して開始した HBM アクティビティ モニターの実行を一時停止します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (必須): 一時停止する HBM アクティビティ モニター オブジェクト (`hw_hbm`) を指定します。

例

次の例では、指定した HBM コアに関連付けられている HBM アクティビティ モニターを一時停止しています。

```
pause_hw_hbm_amon [get_hw_hbms *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)

- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

phys_opt_design

現在の配置済みネットリストを最適化します。

構文

```
phys_opt_design [-fanout_opt] [-placement_opt] [-routing_opt]
  [-slr_crossing_opt] [-rewire] [-insert_negative_edge_ffs]
  [-critical_cell_opt] [-dsp_register_opt] [-bram_register_opt]
  [-uram_register_opt] [-bram_enable_opt] [-shift_register_opt]
  [-hold_fix] [-aggressive_hold_fix] [-retime]
  [-force_replication_on_nets <args>] [-directive <arg>]
  [-critical_pin_opt] [-clock_opt] [-path_groups <args>] [-tns_cleanup]
  [-sll_reg_hold_fix] [-quiet] [-verbose]
```

使用法

名前	説明
[-fanout_opt]	ファンアウトの大きいタイミング クリティカル ネットをセルを複製して最適化します。
[-placement_opt]	タイミング クリティカル ネットに対して配置に基づく最適化を実行します。
[-routing_opt]	タイミング クリティカル ネットに対して配線に基づく最適化を実行します。
[-slr_crossing_opt]	SLR をまたぐタイミング クリティカル ネットの配置最適化を実行します。
[-rewire]	配線を再実行します。
[-insert_negative_edge_ffs]	ホールド最適化のため立ち下がりエッジでトリガーされるフリップフロップを挿入します。
[-critical_cell_opt]	タイミング クリティカル ネットをセルを複製して最適化します。
[-dsp_register_opt]	DSP レジスタの最適化を実行します。
[-bram_register_opt]	ブロック RAM レジスタの最適化を実行します。
[-uram_register_opt]	UltraRAM レジスタの最適化を実行します。
[-bram_enable_opt]	ブロック RAM イネーブルの最適化を実行します。
[-shift_register_opt]	シフト レジスタの最適化を実行します。
[-hold_fix]	ホールド違反のスラックの向上を試みます。
[-aggressive_hold_fix]	積極的にホールド違反のスラックの向上を試みます。
[-retime]	リタイミング最適化を実行します。
[-force_replication_on_nets]	ネットに対して複製最適化を強制的に実行します。
[-directive]	コマンドのモードを指定します。有効な値は、「引数」セクションを参照してください。デフォルトは Default です。
[-critical_pin_opt]	タイミング クリティカル ネットに対してピン スワップに基づく最適化を実行します。
[-clock_opt]	配線後の最適化でクロック スキューの最適化を実行します。
[-path_groups]	指定したパス グループに対してのみ処理を実行します。
[-tns_cleanup]	指定した最適化の条件を満たすデザイン内のすべてのネットに対して最適化を実行し、デザインの TNS を向上します。

名前	説明
<code>[-sll_reg_hold_fix]</code>	SLL Tx-Rx パスに対してホールド違反修正を実行します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Tools (ツール)

説明

デザインの負の Slack パスに対してタイミング ドリブンの最適化を実行します。最適化が実行されるためには、パスの負の Slack がワースト ネガティブ Slack (WNS) に近い必要があります。負の Slack がないデザインには最適化は実行されません。

このオプションのコマンドは、配置後または配線最適化後に実行できます。



推奨: 物理最適化には配置後にのみ提供されるタイミング データが必要なので、このコマンドは配置前には実行できません。ただし、`write_iphys_opt_tcl` および `read_iphys_opt_tcl` コマンドを使用すると、配置後のデザインに実行された物理最適化を記述し、それを配置前のデザイン ネットリストに適用できます。インタラクティブ 物理最適化の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

配置後の `phys_opt_design` コマンドでは、デフォルトで次の最適化が実行されます。

- ファンアウトの大きいネットの最適化
- クリティカル パスの配置に基づく最適化
- リワイヤ
- クリティカル セルの最適化
- DSP レジスタの最適化
- ブロック RAM レジスタの最適化
- UltraRAM レジスタの最適化
- 最終的なファンアウトの最適化



ヒント: `phys_opt_design` コマンドで特定の最適化のコマンド ライン オプションを使用すると、その最適化のみが実行され、ほかの最適化は通常デフォルトで実行される最適化であってもディスエーブルになります。

配線後の `phys_opt_design` コマンドでは、デフォルトで次の最適化が実行されます。

- クリティカル パスの配置に基づく最適化
- 配線最適化
- リワイヤ
- クリティカル セルの最適化

物理最適化には、複製、リタイミング、ホールド違反の修正、および配置の向上が関連します。`phys_opt_design` コマンドでは、ネットリストおよび配置の必要な変更が自動的に実行されます。

リタイミングを実行するには、`-retime` オプションまたは `-directive AddRetime` オプションを指定する必要があります。

ホールド違反の修正を実行するには、`-hold_fix` オプションまたは `-directive ExploreWithHoldFix` オプションを指定する必要があります。

`phys_opt_design` コマンドを複数回実行すると、前回の結果が最適化されます。



ヒント: `phys_opt_design` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメータの設定に関する詳細は、`set_param` コマンドを参照してください。

このコマンドを実行すると、処理された各ネット、実行された最適化のサマリ、および最適化前後の WNS がレポートされます。複製されたオブジェクトの名前は、元のオブジェクト名に `_replica` と複製されたオブジェクトカウントが付いたものになります。

引数

`-fanout_opt` (オプション): ファンアウトの大きいタイミング クリティカル ネットに対し、ドライバーを複製して遅延を削減することにより、遅延ドリブンの最適化を実行します。

注記: `-fanout_opt` の指定はオプションですが、ほかの最適化のコマンド ライン オプションを指定していなければ、この最適化はデフォルトで実行されます。

`-placement_opt` (オプション): タイミング クリティカル ネットの遅延を削減するようセルを移動します。

`-routing_opt` (オプション): タイミング クリティカル ネットの遅延を削減するために配線最適化を実行します。

`-slr_crossing_opt` (オプション): SLR 間の接続のパス遅延を削減するため、配置後または配線後の最適化を実行します。この最適化では、複製の後に SLR をまたぐ部分のドライバー、ロード、またはその両方の位置を調整します。UltraScale および UltraScale+ デバイスで使用します。

`-rewire` (オプション): ロジック コーンを再構築してロジック レベルを削減し、クリティカル信号の遅延を削減します。

`-insert_negative_edge_ffs` (オプション): ホールド最適化のため立ち下がりエッジでトリガーされるフリップフロップを挿入します。

`-critical_cell_opt` (オプション): タイミング クリティカル ネットのセルを複製して遅延を削減します。

`-dsp_register_opt` (オプション): レジスタをスライスから DSP ブロックに移動したり、DSP ブロックからスライスに移動したりして、クリティカル パス遅延を削減します。

`-bram_register_opt` (オプション): レジスタをスライスからブロック RAM に移動したり、ブロック RAM からスライスに移動したりして、クリティカル パス遅延を削減します。

`-uram_register_opt` (オプション): レジスタをスライスから URAM に、または URAM からスライスに URAM セルの境界を越えて移動することにより、クリティカル パス遅延を削減します。

`-bram_enable_opt` (オプション): ブロック RAM イネーブルを最適化すると、消費電力最適化されたブロック RAM に関連するクリティカル パスのタイミングを向上できます。配置前のブロック RAM 消費電力最適化では、ダイナミック消費電力を削減するためにブロック RAM の読み出しおよび書き込みイネーブル入力を駆動するロジックが再構築されます。配置後、再構築されたロジックがタイミング クリティカルになる可能性があります。ブロック RAM イネーブルの最適化を実行すると、イネーブル ロジックの最適化が元に戻され、クリティカルなイネーブル ロジックパスのスラックが向上します。

`-shift_register_opt` (オプション): シフトレジスタの最適化を実行し、シフトレジスタセル (SRL) とほかのロジックセル間の負の Slack パスのタイミングを向上します。シフトレジスタセル (SRL16E または SRLC32E) に入出力するパスにタイミング違反がある場合、この最適化で SRL レジスタチェーンの最初または最後のレジスタを抽出してロジックファブリックに配置することにより、タイミングを向上します。この最適化により、元のクリティカルパスのワイヤ長が短くなります。詳細は、『Vivado Design Suite ユーザーガイド: インプリメンテーション』(UG904) を参照してください。

`-hold_fix` (オプション): データパス遅延を挿入し、ホールドタイム違反を修正します。

`-aggressive_hold_fix` (オプション): データパス遅延を挿入し、ホールドタイム違反を修正します。通常のホールド違反アルゴリズムよりもかなり多くのホールド違反を考慮します。

`-retime` (オプション): レジスタを組み合わせてロジックの後に移動してリタイミングし、パス遅延のバランスを取ります。

`-force_replication_on_nets <args>` (オプション): タイミング Slack にかかわらず、指定のネットオブジェクトのドライバーを強制的に複製します。このオプションでは、`get_nets` コマンドを使用してネットオブジェクトを指定する必要があります。複製はロードの配置に基づいて実行され、複製が十分かどうかを解析する必要があります。さらに複製が必要な場合は、コマンドを複数回実行することによりネットを繰り返し複製できます。

`-directive <arg>` (オプション): 物理最適化モード (デザインの目標) を指定します。1 つの `phys_opt_design` コマンドに対して 1 つのモードのみを指定できます。値では大文字/小文字が区別されます。有効な値は次のとおりです。

- `Explore`: 最適化を複数回実行し、ファンアウトの大きいネットの複製を含め、異なるアルゴリズムを実行します。
- `ExploreWithHoldFix`: 最適化を複数回実行し、ホールド違反の修正およびファンアウトの大きいネットの複製を含め、異なるアルゴリズムを実行します。
- `ExploreWithAggressiveHoldFix`: 最適化を複数回実行し、ホールド違反の修正およびファンアウトの大きいネットの複製を含め、異なるアルゴリズムを実行します。
- `AggressiveExplore`: `Explore` と似ていますが、異なる最適化アルゴリズムが使用され、より厳しい目標が設定されます。
- `AlternateReplication`: クリティカルセルの複製に異なるアルゴリズムを使用します。
- `AggressiveFanoutOpt`: ファンアウトに関連する最適化に異なるアルゴリズムを使用し、より厳しい目標を設定します。
- `AlternateFlowWithRetiming`: 複製および DSP と BRAM の最適化をより積極的に実行し、レジスタのリタイミングをイネーブルにします。
- `AddRetime`: デフォルトの `phys_opt_design` フローを実行し、リタイミングを追加します。
- `RuntimeOptimized`: 実行する物理最適化を少なくし、実行時間を短縮します。デザインパフォーマンスよりもコンパイル時間を短縮することが重要な場合に使用します。`RuntimeOptimized` では、`fanout_opt`、`critical_cell_opt`、`placement_opt`、および `bram_enable_opt` 最適化が実行されます。
- `Default`: `phys_opt_design` をデフォルト設定で実行します。

各モードの効果は、『Vivado Design Suite ユーザーガイド: インプリメンテーション』(UG904) を参照してください。

注記: `-directive` オプションは全体的な最適化ストラテジを制御するもので、特定の最適化オプションとは互換性がありません。`-quiet` および `-verbose` オプションとのみ使用できます。

`-critical_pin_opt`: LUT 入力に対して、論理ピンと物理ピンのリマップ (ピンスワップ) を実行し、クリティカルパスのタイミングを向上します。A1 や A2 などの低速の物理ピンにマップされているクリティカルパス上の論理ピンが、タイミングが向上する場合に A6 や A5 などの高速の物理ピンに割り当て直されます。

注記: LOCK_PINS プロパティが設定されているセルはスキップされ、LOCK_PINS で指定されているピン マップが保持されます。論理ピンから物理ピンへのマップは、論理ピン オブジェクトに対して `get_site_pins` を実行します。

`-clock_opt` (オプション): 配線後の最適化でクロック スキューの最適化を実行します。グローバル クロック バッファを挿入してデスティネーション クロックを遅延させることにより、クリティカル パスのセットアップを向上します。

`-path_groups <args>` (オプション): 指定したパス グループに対してのみ最適化を実行します。

`-tns_cleanup` (オプション): このオプションは `-slr_crossing_opt` オプションと共に使用でき、トータル ネガティブ スラック (TNS) クリーンアップを実行します。このクリーンアップでは、全体的な WNS が低下しなければ、ほかのパスのスラックが多少悪化しても、SLR をまたぐパスの最適化が実行されます。これは、UltraScale および UltraScale+ デバイスのみに適用されます。

`-sll_reg_hold_fix` (オプション): SLL レジスタのホールド違反修正最適化を実行します。このオプションは、配線段階で SLR をまたぐパスのホールド違反が解決しにくい場合に使用できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在の配置後のデザインに物理最適化を実行し、配置前のデザインに適用するために `iphys_opt` Tcl スクリプトを記述しています。

```
phys_opt_design
write_iphys_opt_tcl C:/Data/my_iphys_opt.tcl
```

次の例では、指定のセルに LOCK_PINS プロパティを設定し、タイミングを向上するためレジスタのリタイミング、DSP ブロックおよびブロック RAM のレジスタの最適化、ピン スワップ (固定されているピンを除く) を含む物理最適化を実行します。

```
set_property LOCK_PINS {I3:A1 I2:A4} [get_cell cpuEngine/
qmem_dack_reg_i_1]
phys_opt_design -retime -dsp_register_opt -bram_register_opt \
-critical_pin_opt
```

次の例では、よりよい結果を得るため、ホールド違反の修正を含め、`phys_opt_design` を複数回実行しています。

```
phys_opt_design -directive ExploreWithHoldFix
```

次の例では、`phys_opt_design` コマンドでネットの複製がより考慮されるよう指定しています。

```
phys_opt_design -directive AggressiveFanoutOpt
```

関連項目

- [get_pins](#)
- [get_site_pins](#)
- [opt_design](#)
- [place_design](#)
- [power_opt_design](#)
- [read_iphys_opt_tcl](#)
- [route_design](#)
- [write_iphys_opt_tcl](#)

place_cell

1 つまたは複数のインスタンスを新しい場所に移動または配置します。サイトおよびセルは正しい順序でリストし、サイトの数とセルの数は同じにする必要があります。

構文

```
place_cell [-quiet] [-verbose] <cell_site_list>...
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<cell_site_list>	セルおよびサイトを交互にリストします。

カテゴリ

[Floorplan \(フロアプラン\)](#)

説明

ターゲット パーツのデバイス リソースにセルを配置します。セルは、特定の BEL サイト (SLICE_X49Y60/A6LUT など) または使用可能なスライス リソース (SLICE_X49Y60 など) に配置できます。スライスを指定して BEL を指定しない場合、ツールにより指定されたスライス内の適切な BEL が検索されます。

指定のサイトにセルを配置する際、そのサイトに既に別のオブジェクトが配置されている場合、「Cannot set site and bel property of instances. Site SLICE_X49Y61 is already occupied.」というエラー メッセージが表示されます。

サイトに既にオブジェクトが配置されているかどうかは、BEL サイトの IS_OCCUPIED プロパティをクエリすると確認できます。

```
get_property IS_OCCUPIED [get_bels SLICE_X48Y60/D6LUT]
```

注記: スライスの IS_OCCUPIED プロパティをクエリしても、スライス内に既にオブジェクトが配置されている BEL があるかどうかはわかるだけで、スライスが完全に占有されているかどうかはわかりません。

このコマンドは、セルを配置する場合、およびセルをデバイス上のあるサイトから別のサイトに移動する場合に使用できます。配置されていないセルを配置する構文も、配置されている移動する構文も同じです。

配置されているセルを移動する際、サイトのスライスのみを指定した場合は、セルが現在配置されているのと同様に、新しいスライスの同じ BEL サイトに配置するよう試みられます。たとえば、B6LUT からセルを移動する場合に新しいスライスを指定すると、新しいスライスの B6LUT にセルを配置するよう試みられます。この BEL サイトに既にオブジェクトが配置されている場合は、エラーが返されます。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<cell_site_list> (必須): {<cell_name> <site>} という形式でセルとサイトを指定します。まずセル名を指定し、その後にセルを配置する BEL サイトまたはスライスを指定します。スライスを指定すると、そのスライス内の使用可能な BEL が選択されます。次のようにセル/サイトのペアを複数リストして、複数のセルを複数のサイトに配置できます。

```
{<cell_name1> <site1> <cell_name2> <site2> \
<cell_name3> <site3>... <cell_nameN> <siteN> }
```

例

次の例では、指定したセルを指定した BEL サイトに配置しています。

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y60/D6LUT
```

次の例では、指定したセルを指定したスライスに配置しています。

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y61
```

注記: ツールにより、適切な BEL サイトが選択されます。使用可能な BEL が存在しない場合は、エラーが返されます。

次の例では、複数のセルを複数のサイトに配置しています。

```
place_cell { \
  cpuEngine/cpu_iwb_adr_o/buffer_fifo/i_4810_17734 SLICE_X49Y60/A6LUT \
  cpuEngine/or1200_cpu/or1200_mult_mac/i_4775_15857 SLICE_X49Y60/B6LUT \
  cpuEngine/cpu_iwb_adr_o/buffer_fifo/xlnx_opt_LUT_i_4810_18807_2 \
  SLICE_X49Y60/C6LUT }
```

関連項目

- [create_cell](#)
- [remove_cell](#)
- [unplace_cell](#)

place_design

ポートと最下位インスタンスを自動配置します。

構文

```
place_design [-directive <arg>] [-no_timing_driven] [-timing_summary]
             [-unplace] [-post_place_opt] [-no_psisip] [-no_bufg_opt] [-quiet]
             [-verbose]
```

使用法

名前	説明
[-directive]	コマンドのモードを指定します。有効な値は、「引数」セクションを参照してください。デフォルトは Default です。
[-no_timing_driven]	タイミング ドリブン モードで実行しません。
[-timing_summary]	正確な配置後のタイミング サマリをイネーブルにします。
[-unplace]	制約で固定されていないすべてのインスタンスの配置を解除します。
[-post_place_opt]	配置後の最適化を実行します。
[-no_psisip]	配置の物理合成最適化 (PSIP) をディスエーブルにします。
[-no_bufg_opt]	配置中のグローバル バッファの挿入をディスエーブルにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Tools (ツール)

説明

現在のデザインに含まれる指定のポートおよびロジック インスタンスまたはすべてのポートおよびロジック セルを、ターゲット パーツのデバイス リソースに配置します。負のタイミング スラックを最小限に抑え、ワイヤの長さをなるべく短くし、同時に配置を分散させて配線が密集しないように、ツールにより配置が最適化されます。

配置はインプリメンテーションの 1 つの段階です。Vivado ツールをプロジェクト モードで実行している場合は、インプリメンテーションは `launch_runs` コマンドを使用してすべてを自動的に実行できます。

非プロジェクト モードでは、インプリメンテーション プロセスを `opt_design`、`place_design`、`phys_opt_design`、`power_opt_design`、および `route_design` の各コマンドを使用して手動で実行する必要があります。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

配置および配線はどちらも、インクリメンタル インプリメンテーション フローを使用して、デザイン チェックポイント ファイル (DCP) に保存されている結果に基づいて、インクリメンタルに実行できます。インクリメンタル配置および配線の詳細は、`read_checkpoint` コマンドまたは『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。



ヒント: `place_design` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメーターの設定に関する詳細は、`set_param` コマンドを参照してください。

また、`place_ports` を使用するか、またはセルに LOC プロパティを設定して、デザインの一部のエレメントを手動で配置しておいてから、残りの部分を `place_design` を使用して自動的に配置することも可能です。

このコマンドを実行するには、合成済みデザインを開いている必要があり、最適でないネットリストを配置するのを避けるため、`place_design` コマンドを実行する前に `opt_design` コマンドを実行しておくことをお勧めします。

引数

`-directive <arg>` (オプション): 特定のデザイン目標を達成するように配置します。1 つの `place_design` コマンドに対して 1 つのモードのみを指定できます。値では大文字/小文字が区別されます。有効な値は次のとおりです。

- `Explore`: 詳細配置および配置後の最適化の-effort を増加します。
- `EarlyBlockPlacement`: RAM および DSP ブロックをタイミング ドリブンに配置します。RAM および DSP ブロックの位置は、配置プロセスの早い段階で固められ、残りのロジックを配置するためのアンカーとして使用されます。
- `WLDriivenBlockPlacement`: RAM および DSP ブロックをワイヤ長に基づいて配置します。タイミング ドリブンの配置を無効にし、ブロックとの接続距離を最短にするよう配置します。
- `ExtraNetDelay_high`: ファンアウトが大きく距離の長いネットの見積もり遅延を増加します。high、medium、low の 3 つのレベルがサポートされます。ExtraNetDelay_high では増加量が最も大きくなります。
- `ExtraNetDelay_low`: ファンアウトが大きく距離の長いネットの見積もり遅延を増加します。high、medium、low の 3 つのレベルがサポートされます。ExtraNetDelay_low では増加量が最も小さくなります。
- `AltSpreadLogic_high`: 密集した領域が作成されないように、ロジックをデバイス全体に分散します。high、medium、low の 3 つのレベルがサポートされます。AltSpreadLogic_high では分散度が最も高くなります。
- `AltSpreadLogic_medium`: 密集した領域が作成されないように、ロジックをデバイス全体に分散します。high、medium、low の 3 つのレベルがサポートされます。AltSpreadLogic_medium では分散度が中程度になります。
- `AltSpreadLogic_low`: 密集した領域が作成されないように、ロジックをデバイス全体に分散します。high、medium、low の 3 つのレベルがサポートされます。AltSpreadLogic_low では分散度が最も低くなります。
- `ExtraPostPlacementOpt`: 配置後の最適化の-effort を増加します。
- `ExtraTimingOpt`: タイミング ドリブンの配置にタイミング エffort が増加された代替アルゴリズムを使用します。
- `SSI_SpreadLogic_high`: ロジックを SLR 間に分散します。SSI_SpreadLogic_high では分散度が最も高くなります。
- `SSI_SpreadLogic_low`: ロジックを SLR 間に分散します。SSI_SpreadLogic_low では分散度が中程度になり、配置の実行時間が短縮されます。
- `SSI_SpreadSLLs`: SLR 間で分割を実行し、接続の多い領域に追加のエリアを割り当てます。
- `SSI_BalanceSLLs`: SLR 間で SLL のバランスが取られるように SLR 間で分割を実行します。
- `SSI_BalanceSLRs`: SLR 間でセルの数のバランスが取られるように SLR 間で分割を実行します。
- `SSI_HighUtilSLRs`: 各 SLR でロジックを近くに配置するよう指定します。
- `RuntimeOptimized`: 反復回数を少なくし、デザイン パフォーマンスよりも実行時間を短縮することを優先します。
- `Quick`: 最も高速な、タイミング ドリブンでない、有効なデザインを得るために最低限の配置を実行します。
- `Default`: `place_design` をデフォルト設定で実行します。



重要: `-directive` オプションは全体的な配置ストラテジを制御するもので、一部の `place_design` オプションとは互換性がありません。`-no_psisip`、`-no_bufg_opt`、`-quiet`、および `-verbose` オプションと使用できます。再利用の高いデザインおよび `read_checkpoint -incremental` コマンドで定義されたインクリメンタル インプリメンテーション フローでは、`Explore`、`Quick`、および `Default` 指示子のみを使用できます。配置ストラテジおよび `-directive` オプションの使用法の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。

`-no_timing_driven` (オプション): デフォルトのタイミング ドリブン配置アルゴリズムをディスエーブルにします。このオプションを使用するとワイヤの長さに基づいて高速な配置が実行されますが、タイミング制約は無視されます。

`-timing_summary` (オプション): スタティックタイミング解析からの結果を使用して配置後のワースト ネガティブ スラック (WNS) をレポートします。WNS の値は、配置後のデザインに対して `report_timing_summary` コマンドを実行した場合と同じになります。デフォルトでは、配置レポートにはデザイン インプリメンテーション中のインクリメンタル配置アップデートに基づいて、見積もり WNS がレポートされます。`-timing_summary` オプションを使用すると、タイミング解析が実行されるため、実行時間が長くなります。

`-unplace` (オプション): 制約で固定されていないすべてのインスタンスの配置を解除します。配置が固定されているセル (`IS_LOC_FIXED` が `TRUE`) は変更されません。



ヒント: 固定されているセルの配置を解除するには、`set_property` コマンドを使用して `IS_LOC_FIXED` を `FALSE` に設定します。

`-post_place_opt` (オプション): 配置後に最適化を実行してクリティカル パスのタイミングを向上します。ただし、配置配線の実行時間が増加します。この最適化は、配置後のどの段階でも実行できます。この最適化では、ワースト ケース タイミング パスが解析され、遅延を削減するように配置の向上を試みます。



ヒント: 配置を変更すると未配線の接続が生成されるので、`-post_place_opt` オプションを使用した後 `route_design` コマンドを実行する必要があります。

`-no_psisip` (オプション): 配置の物理合成最適化 (PSIP) をディスエーブルにします。デフォルトでは、遅延を短縮するため、Vivado 配置によりファンアウトの大きいネットのドライバーおよび遠くに離れたロードのドライバーが複製されます。このオプションは、これらの最適化をディスエーブルにします。

`-no_bufg_opt` (オプション): デフォルトでは、ファンアウトの大きいネットを駆動するため配置中にグローバル バッファが挿入されます。このオプションを使用すると、タイミング クリティカルでないファンアウトの大きいネットにより消費される配線リソースの数を削減するため、グローバル バッファの挿入はディスエーブルになります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインを配置して最適化を実行し、デザインを配線して配置後の最適化を実行し、最後に配置後の最適化により生成された未接続のネットを接続するためデザインを再配線しています。

```
place_design  
phys_opt_design  
route_design  
place_design -post_place_opt  
phys_opt_design  
route_design
```

次の例では、よりよい配置結果を得るために、異なる配置アルゴリズムを試しています。

```
place_design -directive Explore
```

次の例では、現在のデザインの配置を解除しています。

```
place_design -unplace
```

関連項目

- [launch_runs](#)
- [opt_design](#)
- [place_ports](#)
- [phys_opt_design](#)
- [power_opt_design](#)
- [read_checkpoint](#)
- [route_design](#)
- [set_property](#)

place_pblocks

Pblock をスライスの容量に基づいてサイズ変更し、接続に基づいて再配置します。

構文

```
place_pblocks [-effort <arg>] [-utilization <arg>] [-quiet] [-verbose]  
<pblocks>...
```

使用法

名前	説明
[-effort]	Pblock ごとの配置ツールのエフォート レベルを指定します。有効な値は LOW、MEDIUM、HIGH で、デフォルトは HIGH です。
[-utilization]	Pblock ごとの使用率を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pblocks>	配置する Pblock を指定します。

カテゴリ

[Floorplan \(フロアプラン\)](#)

説明

Pblock を FPGA のファブリックに配置します。Pblock は `create_pblock` コマンドを使用して作成し、`add_cells_to_pblock` コマンドを使用してロジックを割り当てておく必要があります。

注記: 空の Pblock も指定どおり配置されますが、1 つの CLB タイル (2 つのスライス) が使用されます。

引数

`-effort <arg>` (オプション): 各 Pblock をファブリックに配置するのに Pblock 配置ツールで使用するエフォート レベルを指定します。有効な値は LOW、MEDIUM、HIGH で、デフォルトは HIGH です。

`-utilization <arg>` (オプション): Pblock に割り当てられたロジック エLEMENTで消費されるターゲット デバイスのスライス リソースの割合を指定します。たとえば 50% に指定した場合、Pblock エリア内のスライス リソースの半分を Pblock で使用するよう割り当て、残りの半分はデザインのその他のロジック用に残すように指定されます。使用率を大きくすると Pblock は小さくなり集積度が上がりますが、デザインの配置が困難になることがあります。

注記: Pblock の使用率はスライス ベースのデバイス リソースを考慮しており、Pblock に割り当てられた合成後のロジックに基づいて見積もられます。実際の配置結果は異なる可能性があり、`resize_pblock` コマンドを使用して Pblock のサイズを調整することが必要な場合があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されず。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<pblocks> (必須): FPGA のファブリックに配置する Pblock を 1 つ以上指定します。

例

次の例では、指定した Pblock を使用率を 75% に設定して配置しています。

```
place_pblocks -effort LOW -utilization 75 block1 block2 block3 block4  
block5
```

関連項目

- [add_cells_to_pblock](#)
- [create_pblock](#)
- [resize_pblock](#)

place_ports

ポートのセットを自動的に配置します。

構文

```
place_ports [-skip_unconnected_ports] [-check_only] [-iobank <args>]  
            [-quiet] [-verbose] [<ports>...]
```

使用法

名前	説明
<code>[-skip_unconnected_ports]</code>	接続のないポートは配置しません。
<code>[-check_only]</code>	I/O クロック配置 DRC のみをチェックします。
<code>[-iobank]</code>	配置を指定したバンクに制限します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<ports>]</code>	配置するポートを指定します。指定しない場合、すべてのポートが配置されます。ポート オブジェクトとパッケージ ピン オブジェクトを交互に指定すると、手動の配置が実行されます。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

ポートを自動または手動で配置し、ポートをサイリンクス FPGA パッケージのピンに割り当てます。

- ポートを使用可能な I/O またはクロック サイト、あるいは指定の I/O バンクに自動配置します。
- ポートとパッケージ ピンの両方を指定すると、ポートが指定のパッケージ ピン (package_pin) に手動で割り当てられます。

`place_ports` コマンドでは、ユーザーにより既に配置されているポート、配置および固定されているポートは変更されません。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-skip_unconnected_ports` (オプション): 接続されていないポートは配置しません。

`-check_only` (オプション): クロック配置 DRC を実行します。これは、`report_drc` コマンドで PLCK チェックとしても実行できます。このオプションを使用すると、ポートは配置されず、有効な配置がチェックされるのみです。

`-iobank <args>` (オプション): ポートを指定の I/O バンク オブジェクトに配置します。I/O バンク オブジェクトは、`get_iobanks` コマンドで返されます。

注記: ポートの配置を特定の I/O バンクに制限した場合、指定のポート数を配置するのに十分なサイトがないと、配置エラーが発生します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<ports>` (オプション): 配置するポートの名前を指定します。

- ポートを指定しない場合は、配置されていないすべてのポートが配置されます。
- ポートをポート名 (`<port_name>`) とパッケージ ピン (`<package_pin>`) のペアとして指定すると、ポートを指定のパッケージ ピンに手動で割り当てることができます。`<port_name>` と `<package_pin>` のペアを複数指定する場合は、次の形式を使用します。

```
{<port_name1> <package_pin1> <port_name2> <package_pin2>
<port_name2> <package_pin2>...<port_nameN> <package_pinN> }
```

注記: 既に配置されているポートを指定した場合、これらのポートは置き換えられたり移動されたりしません。

例

次の例では、`get_ports` コマンドで返されたポート オブジェクトを、`get_iobanks` コマンドで返されたデバイスの I/O バンク 13 に配置しています。

```
place_ports -iobank [get_iobanks 13] [get_ports DataOut_pad_1_o]
```

次の例では、`port_name` と `package_pin` のペアを指定して複数のポートを手動で配置しています。

```
place_ports {LEDS_n[2] AA11 LEDS_n[3] AA10 LEDS_n[0] Y11 LEDS_n[1] Y10}
```

次の例では、すべての入力ポートをデバイスの I/O バンク 12、13、14、および 15 に配置しています。

```
place_ports -iobank [get_iobanks {12 13 14 15}] [all_inputs]
```

関連項目

- [create_port](#)
- [get_iobanks](#)
- [make_diff_pair_ports](#)
- [remove_port](#)

platform_verify

ビット レベルでファイルを比較することによりプラットフォームを検証する Tcl タスク。

構文

```
platform_verify -reference <arg> [-untrusted <arg>] [-in_memory]
               [-cell <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
-reference	信頼できる参照 DCP への完全パスを指定します。
[-untrusted]	検証する未確認の DCP への完全パスを指定します。
[-in_memory]	-untrusted オプションを指定しない場合に、インメモリ デザインを比較に使用します。
[-cell]	検証するリコンフィギャラブル セルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

説明

2 つのデザイン チェックポイント ファイル (DCP) を読み込み、ファイルのバイナリ内容をビット レベルで比較します。

引数

-reference <arg> (必須): 比較時に参照する既知の良い DCP を指定します。

-untrusted <arg> (オプション): 検証する未確認の DCP を指定します。



重要: -untrusted および -in_memory の使用はどちらもオプションですが、いずれかを使用して未確認のデザイン ソースを指定する必要があります。

-in_memory (オプション): -untrusted オプションを指定しない場合に、現在のデザイン (current_design) を比較に使用します。

-cell <arg> (オプション): 2 つの DCP ファイルで比較するセルを指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、2 つのデザイン チェックポイントを比較しています。

```
platform_verify -reference C:/Data/design1/routed.dcp -untrusted C:/Data/  
design2/routed.dcp
```

次の例では、現在のデザインと指定したメモリ内のデザイン チェックポイントを比較しています。

```
platform_verify -reference C:/Data/design1/routed.dcp -in_memory
```

関連項目

- [current_design](#)
- [read_checkpoint](#)
- [write_checkpoint](#)

power_opt_design

高度なクロック ゲーティングを使用して、ダイナミック消費電力を最適化します。

構文

```
power_opt_design [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Power (電力)

説明

フリップフロップのクロック イネーブルを利用してクロック ゲーティングを変更することにより、デザインのダイナミック消費電力を最適化します。クロック ゲーティング最適化はデザイン全体に対して自動的に実行され、デザインの動作が変化する可能性のある既存のロジックやクロックを変更せずに消費電力を削減します。

`set_power_opt` コマンドを使用して、消費電力最適化に特定のセルを含めるか除外するかを指定できます。

注記: ブロック RAM の消費電力最適化は、`opt_design` コマンドでデフォルトで実行されます。ブロック RAM の消費電力最適化をディセーブルにするには、`opt_design` のデフォルトを変更するか、`set_power_opt` コマンドを使用して特定のセルを消費電力最適化から除外します。

`power_opt_design` コマンドの前に `read_saif` コマンドを使用して、デザインの最適化でアクティビティ データが考慮されるようにすることもできます。

消費電力最適化は、合成後または配置後に実行できます。このコマンドを配置前に実行すると、デザインの消費電力を削減するために最適化が実行されます。配置後に実行すると、タイミングを保持しながらデザインの消費電力を削減するために最適化が実行されます。配置後に実行すると、`power_opt_design` コマンドで実行可能な最適化は制限されます。最適な結果を得るには、このコマンドを配置前に実行してください。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、開いているデザインの消費電力最適化を実行しています。

```
power_opt_design
```

次の例では、実行する最適化を指定することによりブロック RAM の消費電力最適化をディスエーブルにしてデザインを最適化し、デザインの消費電力最適化を実行しています。

```
opt_design -retarget -propconst -sweep  
power_opt_design
```

関連項目

- [opt_design](#)
- [phys_opt_design](#)
- [read_saif](#)
- [report_power](#)
- [report_power_opt](#)
- [set_power_opt](#)

pr_recombine

階層パースシャル リコンフィギュレーション ソリューションを使用する場合に、親セルをリコンフィギャラブル パーティションとして再設定し、下位リコンフィギャラブル パーティションを削除します。

構文

```
pr_recombine [-cell <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-cell]	(必須) リコンフィギャラブル コンテナ モジュールの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

pr_subdivide

階層パースアル リコンフィギュレーション ソリューションを使用する場合に、リコンフィギャラブル パーティションを 1 つまたは複数の下位リコンフィギャラブル パーティションに分割します。

構文

```
pr_subdivide [-cell <arg>] [-subcells <arg>] [-quiet] [-verbose]  
[<from_dcp>]
```

使用法

名前	説明
[-cell]	(必須) 親リコンフィギャラブル パーティション モジュールの名前を指定します。
[-subcells]	(必須) 子リコンフィギャラブル パーティション モジュールの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<from_dcp>]	(必須) -cell オプションで指定したリコンフィギャラブル モジュールの OOC 合成済みチェックポイントのパスを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

pr_verify

デザイン チェックポイントがボード上で置換可能であるかどうかを検証します。このコマンドでサポートされるフォーマットは、(1) `pr_verify DCP1 DCP2 -full_check`、(2) `pr_verify -initial DCP1 -additional {DCP2 DCP3 DCP4 ...}`、および (3) `pr_verify -in_memory -additional {DCP2 DCP3 ...}` です。フォーマット (3) を使用する場合は、インメモリ デザインを開いている必要があります。

構文

```
pr_verify [-full_check] [-file <arg>] [-initial <arg>] [-additional <arg>]
          [-in_memory] [-quiet] [-verbose] [<file1>] [<file2>]
```

使用法

名前	説明
<code>[-full_check]</code>	デフォルトでは、最初の差異のみがレポートされます。このオプションを <code>true</code> に設定すると、配置または配線の差異がすべてレポートされます。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-initial]</code>	初期チェックポイント (.dcp) を指定します。
<code>[-additional]</code>	追加のチェックポイント (.dcp) を指定します。
<code>[-in_memory]</code>	-additional オプションと共に使用し、比較にインメモリ デザインを使用します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<file1>]</code>	デザイン チェックポイント (.dcp) ファイル 1 を指定します。
<code>[<file2>]</code>	デザイン チェックポイント (.dcp) ファイル 2 を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

デザイン チェックポイント ファイルを比較して、パースナル リコンフィギュレーションに使用できるかどうかを検証します。

パースナル リコンフィギュレーションが可能なデザインがハードウェアで機能するためには、スタティック ロジックの配置および配線がすべてのコンフィギュレーションで一貫している必要があります。また、プロキシ ロジックは同じ場所に配置し、クロック スパインの配線が一致している必要があります。`pr_verify` コマンドは、パースナル リコンフィギュレーション デザイン用に作成された配線済みデザイン チェックポイント ファイル (DCP) を比較し、インポートされたすべてのリソースが一致しているかどうかを検証します。詳細は、『Vivado Design Suite ユーザーガイド: Dynamic Function eXchange』 (UG909) を参照してください。

`pr_verify` コマンドには 2 つのモードがあり、比較する 2 つの DCP ファイルを指定するか、複数の DCP ファイルを最初の DCP ファイルと比較できます。これら 2 つのモードの構文は、次のとおりです。

- `pr_verify DCP1 DCP2`

- `pr_verify -initial DCP1 -additional {DCP2 DCP3 DCP4}`

2 つ目のモードは、次のように `pr_verify` コマンドを複数回実行して最初の DCP とそれ以外の各 DCP を比較すると同じですが、最初の DCP が開いたままになるので、追加の比較の実行時間を短縮できます。

```
pr_verify DCP1 DCP2
pr_verify DCP1 DCP3
pr_verify DCP1 DCP4
```

このコマンドを実行すると、比較結果が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-full_check` (オプション): 2 つのデザイン チェックポイントに対して完全なチェックを実行し、不一致のリソースをすべてレポートします。デフォルトでは、`pr_verify` コマンドは最初の不一致が検出されると停止します。不一致が 1 つでも検出された場合、2 つのデザイン チェックポイントはパーシャル リコンフィギュレーションに使用できません。

`-file <arg>` (オプション): 比較結果を保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-initial <arg>` (オプション): 比較する最初のデザイン チェックポイント ファイルを指定します。

`-additional <args>` (オプション): `-initial` オプションで指定したチェックポイント ファイルと比較する追加のチェックポイントを 1 つまたは複数指定します。複数のチェックポイントは、ダブル クォーテーション (") または波かっこ ({}) で囲む必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file1>`: 比較する最初のデザイン チェックポイント ファイルを指定します。

`<file2>`: 比較する 2 つ目のデザイン チェックポイント ファイルを指定します。

例

次の例では、完全なチェックを実行して、`-initial` オプションで指定した DCP を `-additional` オプションで指定した DCP と比較しています。

```
pr_verify -full_check -initial FastConfig.dcp \
  -additional {corner1.dcp corner2.dcp}
```

関連項目

- [read_checkpoint](#)

- [write_checkpoint](#)

program_hw_cfgmem

コンフィギュレーション メモリ オブジェクトをプログラムします。

構文

```
program_hw_cfgmem [-svf_file <arg>] [-force] [-append] [-quiet] [-verbose]
[<hw_cfgmem>...]
```

使用法

名前	説明
[-svf_file]	生成する SVF ファイルを指定します。
[-force]	-svf_file で指定したファイル名が既に存在する場合に上書きします。
[-append]	既存の SVF ファイルの最後に追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_cfgmem>]	ハードウェア コンフィギュレーション メモリを指定します。デフォルトは現在のハードウェア コンフィギュレーション メモリです。

カテゴリ

Hardware (ハードウェア)

説明

指定したハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトに対して消去およびブランク チェックを実行し、PROGRAM.FILE プロパティで定義されたメモリ コンフィギュレーション ファイルでプログラムした後、検証します。メモリ コンフィギュレーション ファイルは write_cfgmem コマンドを使用して作成し、set_property コマンドを使用して hw_cfgmem オブジェクトに関連付けます (例を参照)。

ザイリンクス FPGA にデザイン特定のデータを読み込む (プログラムする) プロセスは、コンフィギュレーションと呼ばれます。create_hw_cfgmem を使用して、FPGA デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成し、ハードウェア デバイス (hw_device) に関連付けたら、write_cfgmem コマンドで作成したメモリ コンフィギュレーション ファイルからのビットストリームおよびその他のデータでコンフィギュレーション メモリをプログラムできます。hw_cfgmem オブジェクトをプログラムするには、program_hw_cfgmem コマンドを使用します。

program_hw_cfgmem コマンドは複数の手順を実行します。まずコンフィギュレーション メモリ デバイスを消去し、ブランク チェックを実行してデバイスが空であることを確認して、デバイスをメモリ コンフィギュレーション ファイルでプログラムした後、デバイス上のプログラムを検証します。hw_cfgmem オブジェクトのプロパティにより、プログラム プロセスのどの手順を実行するかを指定します。次のプロパティがあります。

- PROGRAM.FILES: デバイスをプログラムするのに使用するメモリ コンフィギュレーション ファイルを指定します。メモリ コンフィギュレーション ファイルは、write_cfgmem コマンドで作成します。

- PROGRAM.ADDRESS_RANGE: プログラムするコンフィギュレーション メモリ デバイスのアドレス範囲を指定します。有効なアドレス範囲の値は、次のとおりです。
 - {use_file}: 消去、ブランク チェック、プログラム、および検証にメモリ コンフィギュレーション ファイルに必要なアドレス空間のみを使用します。
 - {entire_device}: デバイス全体を消去、ブランク チェック、プログラム、および検証します。
- PROGRAM.ERASE: フラッシュ メモリの内容を消去します。これはブール プロパティで、0 (false) または 1 (true) に指定します。
- PROGRAM.BLANK_CHECK: プログラムの前にデバイスにデータがないことを確認します。これはブール プロパティで、0 (false) または 1 (true) に指定します。
- PROGRAM.CFG_PROGRAM: 指定した PROGRAM.FILE でデバイスをプログラムします。これはブール プロパティで、0 (false) または 1 (true) に指定します。
- PROGRAM.VERIFY: プログラム後にデバイスを検証します。これはブール プロパティで、0 (false) または 1 (true) に指定します。

program_hw_cfgmem コマンドでは、インシステムの SVF ファイルを生成し、ザイリンクス デバイスをリモートでプログラムすることもできます。SVF は業界標準のファイル フォーマットで、デバイス チェーンにシフト入力する必要のある情報を記述することにより JTAG チェーンの操作を記述するために使用されます。SVF ファイルは ASCII ファイルであり、テキスト エディターで記述および変更できます。多くのサードパーティ ユーティリティで SVF ファイルを使用して JTAG チェーンのザイリンクス デバイスをプログラムできます。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスが返され、正常に実行されなかった場合はエラーが返されます。

引数

-svf_file <arg> (オプション): SPI および BPI フラッシュ コンフィギュレーション メモ리를プログラムするための SVF ファイルを作成します。SVF ファイルはサードパーティ ツールで使用でき、JTAG トランザクションのトレーシングをサポートし、ブルズアイ カバレッジを向上します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-force (オプション): 指定の SVF ファイルが存在する場合に上書きします。



ヒント: -force オプションと -append オプションを同時に使用すると、-append オプションは無視され、新しい SVF ファイルが作成されて既存のファイルが上書きされます。

-append (オプション): SVF 出力を指定したファイルに上書きするのではなく追加します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_cfgmem> (必須): プログラムする hw_cfgmem オブジェクトを指定します。hw_cfgmem オブジェクトは、名前で指定するのではなく、 `get_hw_cfgmems` または `current_hw_cfgmem` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、hw_cfgmem オブジェクトを作成して現在のハードウェア デバイス (current_hw_device) に関連付け、`write_cfgmem` コマンドを使用してビットストリームから作成されたメモリ コンフィギュレーション ファイルを定義し、プログラム プロセスのほかのプロパティを定義した後、hw_cfgmem オブジェクトをプログラムしています。

```
create_hw_cfgmem -hw_device [current_hw_device] [lindex $cfgParts 0 ]
set cfgMem [current_hw_cfgmem]
set_property PROGRAM.FILE {C:/Data/config_n25q128.mcs} $cfgMem
set_property PROGRAM.ADDRESS_RANGE {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK 1 $cfgMem
set_property PROGRAM.ERASE 1 $cfgMem
set_property PROGRAM.CFG_PROGRAM 1 $cfgMem
set_property PROGRAM.VERIFY 1 $cfgMem
program_hw_cfgmem $cfgMem
```

注記: 上記の例では、現在の hw_cfgmem オブジェクトが変数 \$cfgMem に割り当てられます。

次の例では、現在の hw_cfgmem オブジェクトをプログラムしています。

```
program_hw_cfgmem [current_hw_cfgmem]
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_cfgmem](#)
- [current_hw_device](#)
- [delete_hw_cfgmem](#)
- [get_cfgmem_parts](#)
- [get_hw_cfgmems](#)
- [get_property](#)
- [readback_hw_cfgmem](#)
- [set_property](#)
- [write_cfgmem](#)

program_hw_devices

ハードウェア デバイスをプログラムします。

構文

```
program_hw_devices [-key <arg>] [-clear] [-skip_program_keys]
  [-skip_program_rsa] [-user_efuse <arg>] [-user_efuse_128 <arg>]
  [-control_efuse <arg>] [-security_efuse <arg>] [-only_export_efuse]
  [-svf_file <arg>] [-efuse_export_file <arg>] [-disable_eos_check]
  [-skip_reset] [-force] [-append] [-type <arg>] [-quiet] [-verbose]
  [<hw_device>...]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
[-key]	暗号化プログラム用のキー オプション値を指定します。有効な値は efuse、bbr、none です。
[-clear]	BBR レジスタをクリアします。BBR でのみ有効です。
[-skip_program_keys]	NKY ファイルで指定されているキーのプログラムをスキップします。
[-skip_program_rsa]	NKY ファイルで指定されている RSA キーのプログラムをスキップします。
[-user_efuse]	暗号化プログラム用の 16 進ユーザー eFUSE 値を指定します。
[-user_efuse_128]	暗号化プログラム用の 128 ビットの 16 進ユーザー eFUSE 値を指定します。
[-control_efuse]	暗号化プログラム用の 16 進制御 eFUSE 値を指定します。
[-security_efuse]	暗号化プログラム用の 16 進セキュリティ eFUSE 値を指定します。
[-only_export_efuse]	eFUSE をプログラムせず、設定を -efuse_export_file で指定したファイルにエクスポートします。
[-svf_file]	デバイスのプログラムに使用する SVF ファイルを指定します。
[-efuse_export_file]	プログラムされた eFUSE 設定を保存する出力ファイルを指定します。
[-disable_eos_check]	プログラム後のスタートアップの終了チェックをディスエーブルにします。
[-skip_reset]	プログラムの前にデバイスをリセットしません。
[-force]	SVF ファイルを上書きし、空のファイルを作成します。
[-append]	既存の SVF ファイルの最後に追加します。
[-type]	プログラムに使用するビットストリーム ファイルタイプを指定します。有効な値は bit、bin、rbt です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

名前	説明
[<hw_device>]	ハードウェア デバイスを指定します。デフォルトは現在のハードウェア デバイスです。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア サーバーで開いているハードウェア ターゲット上の指定したハードウェア デバイス オブジェクトをプログラムします。

ハードウェア マネージャーを介してザイリンクス FPGA にアクセスするには、次の Tcl コマンド シーケンスを使用する必要があります。

1. `open_hw`: Vivado Design Suite でハードウェア マネージャーを開きます。
2. `connect_hw_server`: ローカルまたはリモートの Vivado ハードウェア サーバー アプリケーションに接続します。
3. `current_hw_target`: 接続されたサーバーのハードウェア ターゲットを定義します。
4. `open_hw_target`: ハードウェア ターゲットへの接続を開きます。
5. `current_hw_device`: プログラムおよびデバッグに使用するザイリンクス FPGA を指定します。

ターゲット ハードウェア デバイスに接続したら、`set_property` コマンドを使用して、デザインのビットストリーム ファイル (.bit、.rbt、.bin) をデバイスに関連付ける必要があります。

```
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
```

デバッグ プロセスでは、`PROBES.FILE` プロパティを使用してデバイスにプローブ ファイル (.ltx) を関連付けることもできます。

```
set_property PROBES.FILE {C:/Data/debug_nets.ltx} [current_hw_device]
```

ハードウェア デバイスにプログラム ファイルを関連付けたら、`program_hw_devices` コマンドを使用してハードウェア デバイスをプログラムし、ハードウェア マネージャー Tcl コマンドを使用してデバイスをデバッグできます。デバイスを対話的にデバッグするには、ハードウェア マネージャーを Vivado Design Suite IDE で開きます。

暗号化されたビットストリームを指定のハードウェア デバイス (hw_device) にプログラムすることもできます。これには、`write_bitstream` コマンドを使用してビットストリームを生成する前に、インプリメント済みデザインに暗号化プロパティを設定しておく必要があります。デザインに `ENCRYPTION` プロパティを追加するには、Vivado IDE で [Tools] → [Edit Device Properties] をクリックし、[Edit Device Properties] ダイアログ ボックスの [Encryption] ページで設定するのが最も簡単です。[Edit Device Properties] ダイアログ ボックスの詳細は、『Vivado Design Suite ユーザーガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

デバイスを暗号化されたビットストリームでプログラムするのは、`program_hw_devices` コマンドをまず暗号キーを BBR または eFUSE レジスタにプログラムするために実行し、もう一度暗号化されたビットストリームをデバイスにプログラムするために実行する 2 段階のプロセスです。

```
program_hw_devices -key bbr [current_hw_device]
program_hw_device [current_hw_device]
```



注意: eFUSE はハードウェア上の 1 回のみプログラム可能なセルであり、ファクトリでプログラムされる Device DNA、AES-GCM 暗号キー、およびユーザー指定の値を格納するために使用されます。eFUSE レジスタの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』 (UG570) または『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』 (UG470) を参照してください。

`program_hw_devices` コマンドでは、インシステムの SVF ファイルを生成し、ザイリンクス デバイスをリモートでプログラムすることもできます。SVF は業界標準のファイル フォーマットで、デバイス チェーンにシフト入力する必要のある情報を記述することにより JTAG チェーンの操作を記述するために使用されます。SVF ファイルは ASCII ファイルであり、テキスト エディターで記述および変更できます。多くのサードパーティ ユーティリティで SVF ファイルを使用して JTAG チェーンのザイリンクス デバイスをプログラムできます。

このコマンドを実行すると、実行されたアクションが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-key [bbr | efuse]` (オプション): 暗号キーをバッテリー バックアップ SRAM にプログラムするか、または指定したハードウェア デバイス上の eFUSE レジスタにプログラムするかを指定します。暗号キーは、暗号化された BIT ファイルに対して `write_bitstream` コマンドで生成される暗号キー (NKY または NKZ) ファイルで定義され、`ENCRYPTION.FILE` プロパティによりハードウェア ビットストリーム (`hw_bitstream`) オブジェクトに関連付けられます。暗号キーの値はこのファイルから抽出され、ハードウェア デバイスに関連付けられている `hw_bitstream` オブジェクトの `ENCRYPTION.KEY` プロパティに保存されます。



ヒント: ビットストリームの暗号化プロパティは、`write_bitstream` コマンドを使用してビットストリームを生成する前に定義しておく必要があります。

`-clear` (オプション): 現在バッテリー バックアップ SRAM (BBR) にプログラムされている暗号キーを消去します。 `-clear` オプションを `-key` オプションおよび `hw_device` と共に使用し、BBR から暗号キーを消去します。

`-disable_program_keys` (オプション): ハードウェア デバイスのプログラム中に `FUSE_KEY` をプログラムしないよう指定します。

`-disable_program_rsa` (オプション): ハードウェア デバイスのプログラム中に `FUSE_RSA` をプログラムしないよう指定します。

`-user_efuse <arg>` (オプション): `hw_device` の `FUSE_USER` eFUSE レジスタにプログラムする 32 ビット値を指定します。

`-user_efuse_128 <arg>` (オプション): `hw_device` の `FUSE_USER` eFUSE レジスタにプログラムする 128 ビット値を指定します。

`-control_efuse <arg>` (オプション): `hw_device` の `FUSE_CNTL` eFUSE レジスタにプログラムする値を指定します。指定可能な値は、7 シリーズ デバイスでは 14 ビット値、UltraScale デバイスでは 21 ビット値です。

`-security_efuse <arg>` (オプション): `hw_device` のセキュリティ eFUSE レジスタにプログラムする 256 ビット値を指定します。

`-only_export_efuse` (オプション): eFUSE レジスタをプログラムせず、`-efuse_export_file` オプションで指定したファイルに eFUSE 設定をエクスポートします。これにより、eFUSE レジスタを確定する前にデバイスの eFUSE プログラムを確認できます。

`-svf_file <arg>` (オプション): デバイスをプログラム中に SVF ファイルを作成します。SVF ファイルはサードパーティ ツールで使用でき、JTAG トランザクションのトレーシングをサポートし、ブルズアイ カバレッジを向上します。`write_hw_svf` コマンドを使用して、プログラムされた `hw_device` から SVF ファイルを生成することも可能です。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-efuse_export_file <arg>` (オプション): 現在の eFUSE 設定を記述する出力ファイル (.nkz) を指定します。このオプションを指定しない場合、現在の eFUSE 設定は `export_<FUSE_DNA>.nkz` (<FUSE_DNA> はデバイスの DNA 値) という名前のファイルに記述されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-disable_eos_check` (オプション): プログラム後のスタートアップの終了 (EOS) チェックをディスエーブルにします。

`-force` (オプション): 指定した SVF または eFUSE ファイルが存在する場合に上書きします。



ヒント: `-force` オプションと `-append` オプションを同時に使用すると、`-append` オプションは無視され、新しい SVF ファイルが作成されて既存のファイルが上書きされます。

`-append` (オプション): SVF 出力を指定したファイルに上書きするのではなく追加します。

`-type [bit | bin | rbt]` (オプション): ビットストリーム ファイルのタイプを標準ビットストリーム (.bit)、ヘッダーなしのバイナリ ビットストリーム (.bin)、または ASCII 形式のロー ビット ファイル (.rbt) に指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): プログラムされた hw_device の eFUSE レジスタ値を表示します。

<hw_device> (オプション): プログラムする hw_device オブジェクトを 1 つ以上指定します。hw_device は、`get_hw_devices` コマンドを使用してオブジェクトとして指定する必要があります。デバイスを指定しない場合、現在のハードウェア デバイス (current_hw_device) がプログラムされます。

例

次の例では、現在の hw_device オブジェクトを設定し、そのデバイスの PROGRAM.FILE プロパティを設定して、デバイスをプログラムしています。

```
current_hw_device [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Data/design.bit} [current_hw_device]
program_hw_devices [current_hw_device]
```

次の例では、暗号キーをバッテリー バックアップ RAM (BBR) にプログラムし、現在の hw_device にプログラムしています。

```
program_hw_devices -key bbr [current_hw_device]
program_hw_devices [current_hw_device]
```

次の例では、プログラムされたデバイスから SVF ファイルを生成しています。

```
program_hw_devices -force -svf_file {C:/Data/test1.svf} [current_hw_device]
```

次の例では、BBR 暗号キーのプログラムを消去しています。

```
program_hw_devices -key bbr -clear [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [create_hw_device](#)
- [create_hw_target](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)
- [verify_hw_devices](#)
- [write_bitstream](#)
- [write_hw_svf](#)

ptrace

シミュレーションする HDL プロセス名の表示のオン/オフを切り替えます。

構文

```
ptrace [-quiet] [-verbose] <value>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><value></code>	有効な値は、on、true、yes、off、false、no です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレーション デバッグ用にプロセス トレースをイネーブルにします。

シミュレーション中、評価される HDL プロセスの名前と、そのプロセスに関連付けられている行番号が Tcl コンソールに表示されます。



ヒント: プロセス トレースでは、`ltrace` コマンドによる行トレースよりも詳細な情報が表示されます。

この機能をイネーブルにするには、次のように現在のシミュレーション オブジェクトの `PROCESS_TRACING` プロパティを設定します。

```
set_property PROCESS_TRACING on [current_sim]
```

このコマンドを実行すると、プロセス トレースの設定が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<value> (必須): シミュレーション中のプロセストレースをイネーブル/ディスエーブルにします。イネーブルにする場合は true、ディスエーブルにする場合は false に設定します。

例

次の例では、プロセストレースをイネーブルにしています。

```
ptrace true
```

関連項目

- [current_sim](#)
- [ltrace](#)
- [set_property](#)

read_bd

1 つまたは複数の IP インテグレーター デザイン ファイルを読み込みます。

構文

```
read_bd [-quiet] [-verbose] <files>...
```

戻り値

追加された IP インテグレーター デザイン ファイル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	IP インテグレーター デザイン ファイル名を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[IPIntegrator \(IP インテグレーター\)](#)

説明

指定した IP サブシステム デザイン ファイル (ブロック デザイン) を現在のプロジェクトまたはインメモリ デザイン に読み込みます。このコマンドは `add_files` コマンドと似ており、ブロック デザイン ファイルを読み込むと、ソース ファイルセットに追加されます。



推奨: ファイルは現在の場所から読み込まれて参照され、ローカル プロジェクト ディレクトリには移動されません。ファイルをローカル プロジェクトに組み込むには、`import_files` コマンドを使用してください。

このコマンドを使用すると、Vivado ツールを非プロジェクト モードで実行しており、プロジェクト ソース ファイルを管理するプロジェクト ファイルがない場合に、ブロック デザインをインメモリ デザインに読み込むことができます。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

このコマンドを実行すると、読み込まれた IP サブシステム デザイン ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>`: 現在のプロジェクトまたはインメモリ デザインに読み込む IP サブシステム ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、現在のプロジェクトに指定の IP サブシステム デザインを読み込んでいます。

```
read_bd C:/Data/block_designs/design1.bd
```

関連項目

- [add_files](#)
- [import_files](#)
- [open_bd_design](#)
- [save_bd_design](#)

read_checkpoint

デザイン チェックポイントを読み込みます。

構文

```
read_checkpoint [-cell <arg>] [-incremental] [-directive <arg>]
               [-reuse_objects <args>] [-fix_objects <args>] [-dcp_cell_list <args>]
               [-quiet] [-verbose] [<file>]
```

使用法

名前	説明
[-cell]	指定のセルをチェックポイントに置き換えます。セルはブラックボックスである必要があります。
[-incremental]	インプリメンテーションを再利用するデザイン チェックポイント ファイルを指定します。
[-directive]	コマンドのモードを指定します。有効な値は、「引数」セクションを参照してください。デフォルトは RuntimeOptimized です。
[-reuse_objects]	指定したセル、クロック領域、SLR のみを再利用します。
[-fix_objects]	指定したセル、クロック領域、SLR、またはデザインを固定します。
[-dcp_cell_list]	セルと DCP のペアを {<cell1> <dcp1> <cell2> <dcp2>} のように指定します。値は波かっこで囲む必要があります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	デザイン チェックポイント ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ネットリスト、制約、およびオプションでインプリメント済みデザインの配置配線情報を含むデザイン チェックポイント ファイル (DCP) を読み込みます。write_checkpoint コマンドを使用すると、デザインのどの段階でもデザイン チェックポイントでも保存できます。

read_checkpoint コマンドは、デザインまたはプロジェクトをメモリに読み込まずに、チェックポイント ファイルを読み込みます。インポートしたチェックポイントからプロジェクトを作成するには、read_checkpoint コマンドではなく open_checkpoint コマンドを使用するか、read_checkpoint コマンドの後に link_design コマンドを使用して読み込んでいるチェックポイントまたはチェックポイント ファイルからインメモリ デザインを開きます。

注記: Vivado ツールで複数のデザイン チェックポイントを開いている場合、current_project コマンドを使用してデザインを切り替える必要があります。current_design を使用して、どのチェックポイントがアクティブ デザインかを確認できます。

引数

`-cell <arg>` (オプション): 読み込むチェックポイント ファイルからのネットリスト データを挿入するブラック ボックス セルを指定します。このオプションは、`-incremental` または関連のオプションと共に使用することはできません。

`-incremental` (オプション): 既に開いているデザインにチェックポイント ファイルを読み込み、インクリメンタル インプリメンテーション デザイン フローをイネーブルにします。`<file>` には、インクリメンタル デザイン チェックポイント (DCP) ファイルのパスとファイル名を指定します。インクリメンタル インプリメンテーション フローでは、インクリメンタル DCP からの配置および配線が現在のデザインの一致するネットリスト オブジェクトに適用され、既存の配置および配線が再利用されます。インクリメンタル インプリメンテーションの詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。



重要: `-incremental` オプションは、2 つの DCP ファイルを 1 つのデザインに統合するのではなく、インクリメンタル チェックポイントの配置および配線を現在のデザインのネットリスト オブジェクトに適用します。

インクリメンタル デザイン チェックポイントを読み込んだら、`report_incremental_reuse` コマンドを使用してインクリメンタル チェックポイントから現在のデザインに再利用される物理データの割合を確認できます。

`place_design` および `route_design` コマンドでインクリメンタル配置配線が実行され、再利用された配置配線情報が保持されて、デザイン ソリューションに組み込まれます。

`-incremental` オプションでデザイン チェックポイントを読み込むと、物理データが現在のメモリ内のデザインに読み込まれます。インクリメンタル デザイン データをクリアするには、現在のデザインを再読み込みするか、`open_run` を使用してインスタンスの合成 run を開くか、新しいインクリメンタル チェックポイントを読み込んで既に読み込まれているものを上書きします。

`-directive [RuntimeOptimized | TimingClosure | Quick]` (オプション): `-incremental` プロセスの操作モードを指定します。

- **RuntimeOptimized:** インクリメンタル DCP の WNS をターゲットとします。タイミングよりも実行時間を短縮することが優先されます。
- **TimingClosure:** ターゲット WNS は 0 です。実行時間が長くなっても、タイミングを満たすことが試みられます。
- **Quick:** 低いエフォートでタイミングドリブンでないインプリメンテーション モードを使用します。実行時間は最短になります。

`-reuse_objects <args>` (オプション): `-incremental` オプションと共に使用し、インクリメンタル チェックポイントの指定したセル、クロック領域、および SLR の配置配線データのみを再利用します。



ヒント: このオプションを指定しない場合は、デザイン全体が再利用されます。`-reuse_objects` オプションを複数回使用して、異なるオブジェクト タイプを再利用できます。例を参照してください。

`-fix_objects` (オプション): `-incremental` を指定した場合に、指定したセルの配置場所を固定 (IS_LOC_FIXED) としてマークし、`place_design` コマンドで変更されないようにします。このオプションで指定したセル、クロック領域、SLR、または現在のデザイン (`current_design`) の配置が固定されます。

`-dcp_cell_list <arg>` (オプション): セルと DCP のペアを指定します。`read_checkpoint` コマンドの 1 回の実行で、指定したセルの複数の DCP ファイルを読み込むことができます。{<cell1> <dcp1> <cell2> <dcp2>} のように、セルと DCP のペアのリストを波かっこで囲んで指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 読み込むチェックポイント ファイルのパスとファイル名を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定したチェックポイント ファイルをインポートし、さまざまなデザイン エlementをリンクして、指定した名前のインメモリ デザインを作成しています。

```
read_checkpoint C:/Data/checkpoint.dcp
link_design -name Test1
```

次の例では、現在のデザインにインクリメンタル配置配線用のデザイン チェックポイントを読み込んでいます。

```
read_checkpoint -incremental C:/Data/routed.dcp
```

次の例では、DSP およびブロック RAM の配置配線のみを再利用および固定しています。

```
read_checkpoint -incremental C:/Data/routed.dcp \
-reuse_objects [all_rams] -reuse_objects [all_dsps] -fix_objects
[current_design]
```



ヒント: 上記の `-reuse_objects` オプションは、次のように記述することもできます。

```
-reuse_objects [get_cells -hier -filter {PRIMITIVE_TYPE =~ BMEM.*.* ||
PRIMITIVE_TYPE =~ MULT.dsp.* }]
```

次の例では、階層セル `cpuEngine` 内のセルの配置配線を再利用し、DSP セルの配置を固定しています。

```
read_checkpoint -incremental C:/Data/routed.dcp -reuse_objects [get_cells
cpuEngine] -fix_objects [all_dsps]
```

関連項目

- [all_dsps](#)
- [config_implementation](#)
- [current_design](#)
- [current_project](#)
- [get_cells](#)
- [link_design](#)
- [open_checkpoint](#)
- [report_config_implementation](#)
- [write_checkpoint](#)

read_csv

パッケージ ピンとポート配置情報をインポートします。

構文

```
read_csv [-quiet_diff_pairs] [-quiet] [-verbose] <file>
```

使用法

名前	説明
<code>[-quiet_diff_pairs]</code>	I/O ポートをインポートする際に、差動ペアの推論に関する警告メッセージを表示しないようにします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ピン プランニング CSV ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

CSV (Comma Separated Value) ファイルから、ポート定義およびパッケージ ピン配置情報をインポートします。

CSV ファイルのポート定義は、I/O ピン プランニング プロジェクトにインポートできます。ピン プランニング プロジェクトでは、CSV ファイルをインポートすると、現在のポート定義が置き換えられます。インポートした CSV ファイルに存在しないポートは削除されます。

その他のプロジェクトでは、ポート定義はソース デザイン データに定義されていますが、パッケージ ピンの割り当ておよびポート属性は指定の CSV ファイルから読み込むことができます。

CSV ファイルのポート名にスペースを含めることはできません。CSV ファイルから読み込まれたポート名にスペースが含まれていると、エラー メッセージが表示されます。CSV ファイルの形式および要件は、『Vivado Design Suite ユーザー ガイド: I/O およびクロックの配置』(UG899) を参照してください。

引数

`-quiet_diff_pairs` (オプション): CSV ファイルをインポートする際、差動ペアと認識されたピンに関するメッセージが表示されます。このオプションを指定すると、差動ペアの推論に関するメッセージは表示されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): インポートする CSV ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、開いているプロジェクトに CSV ファイルをインポートしています。

```
read_csv C:/Data/pinList.csv
```

次の例では、新しい I/O ピン プランニング プロジェクトを設定してから、指定した CSV ファイルをインポートし、CSV ファイルの差動ペアを推論しています。

```
create_project myPinPlan C:/Data/myPinPlan -part xc7v285tffg1157-1
set_property design_mode PinPlanning [current_filesset]
open_io_design -name io_1
read_csv C:/Data/import.csv
infer_diff_pairs -filetype csv C:/Data/import.csv
```

注記: プロジェクトの特性は、ソース ファイルセットの `design_mode` プロパティで指定します。

関連項目

- [create_project](#)
- [infer_diff_pairs](#)
- [open_io_design](#)
- [set_property](#)
- [write_csv](#)

read_edif

1 つまたは複数の EDIF または NGC ファイルを読み込みます。

構文

```
read_edif [-quiet] [-verbose] <files>
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	EDIF または NGC ファイルの名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

EDIF または NGC ネットリスト ファイルを現在のプロジェクトのデザイン ソース ファイルセットにインポートします。



重要: Vivado Design Suite では、UltraScale デバイスに対して NGC フォーマットのファイルはサポートされていません。Vivado Design Suite で IP を再生成し、ネイティブ出力ファイルを使用することをお勧めします。NGC ファイルは、NGC2EDIF コマンドで EDIF に変換してインポートすることもできます。詳細は、『ISE から Vivado Design Suite への移行ガイド』(UG911) を参照してください。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): インポートする EDIF または NGC ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、開いているプロジェクトに EDIF ファイルをインポートしています。

```
read_edif C/Data/bft_top.edf
```

関連項目

- [write_edif](#)

read_hw_ila_data

ファイルからハードウェア ILA データを読み出します。

構文

```
read_hw_ila_data [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ハードウェア ILA データ ファイル名を指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

指定のファイルから ILA デバッグ コア データを読み出し、ハードウェア ILA データ (hw_ila_data) オブジェクトを作成します。

ILA デバッグ サンプル データは、`upload_hw_ila_data` コマンドを使用して動作中のデバイスから取得します。これによりハードウェア ILA データ (hw_ila_data) オブジェクトが作成され、`write_hw_ila_data` コマンドを使用してファイルとしてディスクに書き込むことができます。このコマンドは、その ILA データ ファイルを読み込みます。

`read_hw_ila_data` で作成される hw_ila_data オブジェクトの名前は、データを読み出すファイルの名前 (<file>) になります。同じ名前の hw_ila_data オブジェクトが既に存在する場合は、オブジェクト名の後に 1 から開始する番号が追加されます (<file>_1 など)。

新しい hw_ila_data オブジェクトは、デザインの ILA に接続または関連付けられていません。

ディスクから読み出した ILA デバッグ データは、`display_hw_ila_data` コマンドを使用して Vivado ロジック解析機能の波形ビューアーで表示できます。

このコマンドを実行すると、ILA データ オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 読み出す ILA データ ファイルの名前を指定します。拡張子を指定しない場合は、`.ila` であると想定されます。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した ILA データ ファイルを読み出し、そのファイル名の `hw_ila_data` オブジェクトを作成しています。

```
read_hw_ila_data C:/Data/hw_ila_data_2.ila
```

関連項目

- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [display_hw_ila_data](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [run_hw_ila](#)
- [write_hw_ila_data](#)

read_hw_sio_scan

ファイルからハードウェア SIO スキャン データを読み出します。ハードウェア SIO スキャン データを指定しない場合は、作成されます。

構文

```
read_hw_sio_scan [-quiet] [-verbose] <file> [<hw_sio_scan>]
```

戻り値

ハードウェア SIO スキャン オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ハードウェア SIO スキャン ファイル名を指定します。
<code>[<hw_sio_scan>]</code>	ハードウェア SIO スキャン データ オブジェクトを指定します。デフォルトはありません。

カテゴリ

Hardware (ハードウェア)

説明

Vivado Design Suite のハードウェア マネージャー機能にハードウェア SIO スキャン データ ファイルを読み込み、ハードウェア SIO スキャン (hw_sio_scan) オブジェクトを作成します。

SIO スキャン データは、run_hw_sio_scan コマンドを使用してスキャンを実行した後、write_hw_sio_scan コマンドを使用してディスクに保存できます。このコマンドは、そのデータ ファイルを読み込みます。

hw_sio_scan オブジェクトを指定しない場合、新しい hw_sio_scan オブジェクトが作成され、既存の hw_sio_scan オブジェクトに続いて順に名前が付けられます。SIO スキャン データをディスクから読み出した後、display_hw_sio_scan コマンドを使用して Vivado シリアル I/O 解析機能の波形ビューアーでプロットして表示できます。

このコマンドを実行すると、hw_sio_scan オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 読み出す SIO スキャン データ ファイルの名前を指定します。拡張子を指定しない場合は、`.csv` であると想定されます。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

`<hw_sio_scan>` (オプション): スキャン データ ファイルを読み込む既存のハードウェア SIO スキャン オブジェクトを指定します。`hw_sio_scan` オブジェクトは、名前で指定するか、または `get_hw_sio_scans` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、指定した SIO スキャン データ ファイルを既存の `hw_sio_scan` オブジェクトに読み込んでいます。

```
read_hw_sio_scan C:/Data/LoopBack1.csv SCAN_0
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [display_hw_sio_scan](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [run_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

read_hw_sio_sweep

ディレクトリからハードウェア SIO スweep データを読み出します。ハードウェア SIO スweep データを指定しない場合は、作成されます。

構文

```
read_hw_sio_sweep [-quiet] [-verbose] <directory> [<hw_sio_sweep>]
```

戻り値

ハードウェア SIO スweep オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><directory></code>	ハードウェア SIO スweep のディレクトリ名を指定します。
<code>[<hw_sio_sweep>]</code>	ハードウェア SIO スweep データ オブジェクトを指定します。デフォルトはなしです。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

Vivado Design Suite のハードウェア マネージャー機能にハードウェア SIO スweep データ ディレクトリを読み込み、ハードウェア SIO スweep (hw_sio_sweep) オブジェクトを作成します。

SIO スweep データは、`run_hw_sio_sweep` コマンドを使用してスweep を実行した後、`write_hw_sio_sweep` コマンドを使用してディスクに保存できます。このコマンドは、複数の SIO スキャン データ ファイルを含むスweep ディレクトリを読み込みます。

hw_sio_sweep オブジェクトを指定しない場合、新しい hw_sio_sweep オブジェクトが作成され、既存の hw_sio_sweep オブジェクトに続いて順に名前が付けられます。SIO スweep の SIO スキャンのいずれかをディスクから読み出した後、`display_hw_sio_scan` コマンドを使用して Vivado シリアル I/O 解析機能の波形ビューアーでプロットして表示できます。

このコマンドを実行すると、hw_sio_sweep オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<directory>` (必須): 読み込む SIO スイープ データを含むディレクトリの名前を指定します。

`<hw_sio_sweep>` (オプション): スイープのスキャン データ ファイルを読み込む既存の `hw_sio_sweep` オブジェクトを指定します。 `hw_sio_sweep` オブジェクトは、名前で指定するか、または `get_hw_sio_sweeps` コマンドでオブジェクトとして返すことにより指定します。

例

次の例では、指定したスイープ ディレクトリを読み込み、新しい `hw_sio_sweep` オブジェクトを作成して、そのスイープのハードウェア SIO スキャン (`hw_sio_scan`) の 1 つを表示しています。

```
read_hw_sio_sweep C:/Data/SWEEP_1/
display_hw_sio_scan [get_hw_sio_scans {SCAN_86}]
```

関連項目

- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [display_hw_sio_scan](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

read_ip

1 つまたは複数の IP ファイルを読み込みます。

構文

```
read_ip [-quiet] [-verbose] <files>
```

戻り値

追加された IP ファイル オブジェクトのリスト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><files></code>	IP ファイル名を指定します。

カテゴリ

FileIO (ファイル入力および出力)、IPFlow (IP フロー)

説明

指定した IP ファイル (XCI) を読み込み、デザインおよび現在のファイルセットに追加します。add_files コマンドと同様に、ファイルは参照で現在のプロジェクトに追加されます。

Vivado ツールを非プロジェクト モードで実行しており、プロジェクト ソース ファイルを管理するプロジェクト ファイルがない場合に、このコマンドを使用してソース ファイルの内容をメモリのデザインに読み込むことができます。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

read_ip コマンドを使用すると、デザイン チェックポイント (DCP) を含む IP コアに関連するすべての出力ファイルがメモリに読み込まれます。



ヒント: プロジェクト ベースのデザイン フローでは、IP コアに関連する必要な出力ファイルが自動的に生成されます。非プロジェクト フローでは、synth_ip または generate_target コマンドを使用して必要な出力ファイルを生成する必要があります。IP での作業に関する詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』(UG896) を参照してください。

IP コアを追加し、ファイルをローカル プロジェクト ディレクトリにインポートするには、import_ip コマンドを使用します。

このコマンドを実行すると、読み込まれたファイルのリストが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): 現在のプロジェクトに読み込む IP ファイルを指定します。XCI および XCO ファイル フォーマットの両方がサポートされています。XCI ファイルは、IP のパラメーター情報を含む IP-XACT 形式のファイルです。XCO ファイルは、IP コアを生成する際に使用されるカスタマイズ パラメーターとプロジェクト オプションがすべて記述された CORE Generator ファイルです。

例

次の例では、指定した IP ファイルを読み込んでいます。

```
read_ip C:/test_ip/char_fifo.xci
```

関連項目

- [add_files](#)
- [import_ip](#)

read_iphys_opt_tcl

iPhysOpt スクリプトを読み込み、実行します。

構文

```
read_iphys_opt_tcl [-fanout_opt] [-critical_cell_opt] [-placement_opt]
                  [-rewire] [-dsp_register_opt] [-bram_register_opt] [-uram_register_opt]
                  [-shift_register_opt] [-auto_pipeline] [-critical_pin_opt]
                  [-include_skipped_optimizations] [-place] [-insert_negative_edge_ffs]
                  [-hold_fix] [-slr_crossing_opt] [-quiet] [-verbose] [<input>]
```

使用法

名前	説明
[-fanout_opt]	ファンアウト最適化を実行します。
[-critical_cell_opt]	タイミングクリティカル ネットをセルを複製して最適化します。
[-placement_opt]	タイミングクリティカル ネットの遅延を削減するようセルを移動します。
[-rewire]	配線を再実行します。
[-dsp_register_opt]	DSP レジスタの最適化を実行します。
[-bram_register_opt]	BRAM レジスタの最適化を実行します。
[-uram_register_opt]	UltraRAM レジスタの最適化を実行します。
[-shift_register_opt]	シフト レジスタの最適化を実行します。
[-auto_pipeline]	自動パイプライン処理を実行します。
[-critical_pin_opt]	ピン スワップ最適化を実行します。
[-include_skipped_optimizations]	スキップされた最適化を含めます。
[-place]	配置を再実行します。
[-insert_negative_edge_ffs]	ホールドが大きくなるのを回避するため、立ち下がりエッジでトリガーされるフリップフロップを挿入します。
[-hold_fix]	ホールド修正最適化用にバッファを挿入します。
[-slr_crossing_opt]	SLR をまたぐネットを最適化します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<input>]	iPhysOpt.tcl ファイルを指定します。

カテゴリ

Tools (ツール)

説明

インタラクティブ物理最適化は、次の 2 つの方法で使用できます。

- 全体的な配置結果とデザイン パフォーマンスを向上するため、配置前のネットリストに配置後の物理最適化を適用する。
- 物理最適化を必要に応じて繰り返すことができるように Tcl スクリプトに保存する。

配置後の最適化を配線前のネットリストに適用するには、インプリメンテーション run をリセットし、合成済みデザインまたは `opt_design` チェックポイントを開いて、`iphys_opt` Tcl スクリプトを読み込んで物理最適化を適用します。

`iphys_opt` Tcl スクリプトからの最適化をすべて適用するか、`read_iphys_opt_tcl` コマンドのオプションを使用して特定の最適化を適用できます。定義されていたが物理最適化中にスキップされた最適化を含めることもできます。

`iphys_opt` Tcl スクリプトに配置データが含まれる場合は、そのデータを使用して最適化されたセルをデザインに配置できます。

`iphys_opt` Tcl スクリプトを読み込んで最適化されたセルを配置したら、デザイン全体の配置を再実行します。複製によるファンアウトの大きいネットの削減、ブロック RAM 出力からの長いパスの削減など、`phys_opt_design` 最適化の効果を配置前に組み込むことができます。ネットリストの最適化を前もって適用することにより、配置およびデザイン パフォーマンスが向上するはずです。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-fanout_opt` (オプション): 指定のインタラクティブ物理最適化 Tcl スクリプトで定義されているファンアウト最適化を適用します。

`-critical_cell_opt` (オプション): 指定の Tcl スクリプトで定義されているセル複製最適化を適用します。

`-placement_opt` (オプション): 指定の Tcl スクリプトで定義されているセル配置最適化を適用します。

`-rewire` (オプション): 指定の Tcl スクリプトで定義されているロジック コーンの再構築を適用します。

`-dsp_register_opt` (オプション): 指定のインタラクティブ物理最適化 Tcl スクリプトで定義されている DSP の最適化を適用します。

`-bram_register_opt` (オプション): 指定の Tcl スクリプトで定義されているブロック RAM の最適化を適用します。

`-shift_register_opt` (オプション): 指定の Tcl スクリプトで定義されているシフト レジスタの最適化を適用します。

`-critical_pin_opt`: 指定の Tcl スクリプトで定義されているピンのスワップを適用します。

`-include_skipped_optimization` (オプション): 入力 Tcl スクリプトで定義されているスキップされた最適化と、標準最適化を適用します。これらは、`phys_opt_design` により最適化されたロジックに適切な場所が見つからなかったためにスキップされた最適化です。このオプションを指定した場合、`iphys_opt_design` コマンドは配置前のネットリストにスキップされた最適化を適用しようとします。

`-place` (オプション): 入力 Tcl スクリプトで定義されている配置を復元します。入力 `iphys_opt` Tcl スクリプトに Tcl スクリプトが記述されたときに指定された配置データが含まれている場合は、このオプションによりその配置データが適用されます。入力スクリプトに配置データが含まれない場合は、`-place` オプションは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<input> (必須): 読み込むインタラクティブ物理最適化 Tcl ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定のインタラクティブ物理最適化 Tcl スクリプトで定義されているブロック RAM の最適化を適用し、最適化されたセルの配置データを適用しています。

```
open_checkpoint C:/Data/opt_design.dcp
read_iphys_opt_tcl -shift_register_opt -place C:/Data/my_iphys_opt.tcl
```

関連項目

- [iphys_opt_design](#)
- [phys_opt_design](#)
- [report_phys_opt](#)
- [write_iphys_opt_tcl](#)

read_mem

1 つまたは複数のデータ ファイル (.mem、.mif、.dat) を読み込みます。

構文

```
read_mem [-quiet] [-verbose] <files>...
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	データ ファイル (.mem、.coe、.dat) の名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ビヘイビアー シミュレーション、合成、および合成後のシミュレーションで BRAM メモリを初期化するため、メモリ ファイル (MEM、DAT、COE) を読み込んでインメモリ デザインまたは現在のプロジェクトに追加します。

デザインでメモリが初期化されない場合、メモリはすべて 0 に初期化されます。

このコマンドを実行すると、読み込まれたファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<files> (必須): 読み込む MEM、DAT、または COE ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次に例を示します。

```
read_mem C:/Data/design1.mem
```

関連項目

- [generate_mem_files](#)
- [write_bmm](#)

read_qor_suggestions

指定したファイルから QoR 推奨項目を読み込みます。

構文

```
read_qor_suggestions [-quiet] [-verbose] <file>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	QoR 推奨項目ファイルを指定します。QoR 推奨項目ファイルへのパスを .rqs を付けて指定します。

カテゴリ

FileIO (ファイル入力および出力)、Feasibility (設計実現可能性)、Timing (タイミング)

説明

指定したファイルから QoR 推奨項目を読み込みます。

このコマンドを実行すると、読み込まれた QoR ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): QoR 推奨項目を読み出すファイルを指定します。拡張子 `.rqs` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定したファイルから QoR 推奨項目を読み込み、開いているデザインの QoR 推奨項目をレポートしています。

```
read_qor_suggestions C:/Data/qor_results.rqs  
report_qor_suggestions -of_objects [get_qor_suggestions]
```

関連項目

- [get_qor_suggestions](#)
- [report_qor_suggestions](#)
- [write_qor_suggestions](#)

read_saif

SAIF フォーマットのシミュレーション データをインポートします。

構文

```
read_saif [-strip_path <arg>] [-no_strip] [-out_file <arg>] [-quiet]
          [-verbose] <file>
```

使用法

名前	説明
<code>[-strip_path]</code>	現在のデザインのインスタンス名を SAIF ファイルに現れるものと同時に指定します。
<code>[-no_strip]</code>	SAIF ファイルから階層の最初の 2 つのレベルを削除しません。
<code>[-out_file]</code>	不一致のネットを含む出力ファイルの名前を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	読み込む SAIF ファイルの名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Power \(電力\)](#)、[Simulation \(シミュレーション\)](#)

説明

`report_power` コマンドでの消費電力解析または `power_opt_design` コマンドでの消費電力最適化に使用する SAIF (Switching Activity Interchange Format) ファイルを読み込みます。`read_saif` コマンドを実行すると、SAIF ファイルからのアクティビティがデザイン ノードにアノートされ、消費電力が正しく見積もられます。

SAIF ファイルを読み込んだ後に `report_power` または `power_opt_design` コマンドを実行すると、そのファイルからのアクティビティ レートが解析および最適化に使用されます。

引数

`-strip_path <arg>` (オプション): SAIF ファイルのエレメントから指定のインスタンス パス接頭辞を削除し、エレメントが現在のデザインのインスタンスに適切にマップされるようにします。



ヒント: スラッシュ (/) で開始するインスタンス パスを指定することはできません。`read_saif` コマンドでは、デザイン ネット名 (スラッシュで開始しない) が検索されます。

`-no_strip` (オプション): SAIF ファイルから階層の最初の 2 つのレベルを削除しません。

`-out_file <arg>` (オプション): 不一致のネットおよびその他のメッセージがレポートされる出力ファイルの名前を指定します。このファイルは、SAIF ファイルのインポート中に作成されます。`-out_file` オプションを指定しない場合、これらの情報はファイルに保存されません。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 読み込む SAIF ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次に例を示します。

```
read_saif -strip_path design/top/F1 C:/Data/design1.saif
```

関連項目

- [power_opt_design](#)
- [report_power](#)

read_schematic

回路図をインポートします。

構文

```
read_schematic [-name <arg>] [-quiet] [-verbose] <file>
```

戻り値

以前にエクスポートされたファイルの名前

使用法

名前	説明
[-name]	[Schematic] ウィンドウのタイトルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	入力ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

Vivado Design Suite から `write_schematic` コマンドを使用して以前にエクスポートされたネイティブ回路図ファイルをインポートします。

引数

`-name <arg>` (オプション): 回路図ファイルを読み込んだときに開く [Schematic] ウィンドウの名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): 読み込む回路図ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した回路図ファイルを読み込んで、Vivado IDE で Sheet_1 という [Schematic] ウィンドウを開いています。

```
read_schematic C:/Data/mySchematic.txt -name Sheet_1
```

関連項目

- [write_schematic](#)

read_twx

TRACE スタティック タイミング解析ツールからタイミング結果を読み込みます。

構文

```
read_twx [-cell <arg>] [-pblock <arg>] [-quiet] [-verbose] <name> <file>
```

使用法

名前	説明
[-cell]	指定したセルに対してレポート ファイルの名前を解釈します。
[-pblock]	指定した Pblock に対してレポート ファイルの名前を解釈します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	結果のセット名を指定します。
<file>	TRACE インポート ファイルの名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ザイリンクス TRACE (Timing Reporter And Circuit Evaluator) ツールで生成された TWX 形式のタイミング レポートをインポートします。TWX は最上位にインポートするか (デフォルト)、特定のセル レベルまたは特定の Pblock に相対的にインポートできます。

TWX ファイルをインポートすると、タイミング結果が GUI の [Timing] ウィンドウに表示されます。

引数

-cell <arg> (オプション): TWX ファイルをインポートする現在のデザインの階層セルの名前を指定します。タイミング パスは、指定したセルに適用されます。

-pblock <arg> (オプション): 現在のデザインの Pblock の名前を指定します。タイミング パスは、指定したブロックに対してインポートされます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<name> (必須): TWX ファイルのタイミング パスをインポートする際に作成する [Timing Results] ウィンドウの名前を指定します。

注記: <name> と <file> の両方を指定し、<name> を先に指定する必要があります。

<file> (必須): インポートする TWX ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した TWX ファイルをデザインの最上位に読み込んでいます。

```
read_twx C:/Data/timing_files/bft.twx
```

関連項目

- [report_timing](#)

read_verilog

1 つまたは複数の Verilog ファイルを読み込みます。

構文

```
read_verilog [-library <arg>] [-sv] [-quiet] [-verbose] <files>...
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
[-library]	ライブラリ名を指定します。Vivado 合成では無視されます。デフォルトは xil_defaultlib です。
[-sv]	SystemVerilog コンパイルをイネーブルにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	Verilog ファイル名を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

Verilog または SystemVerilog ソース ファイルを読み込みます。このコマンドは `add_files` コマンドと似ており、Verilog ファイルを読み込むと、ソース ファイルセットに追加されます。-library オプションを指定すると、ファイルはライブラリ プロパティを設定して追加されます。

Vivado ツールを非プロジェクト モードで実行しており、プロジェクト ソース ファイルを管理するプロジェクト ファイルがない場合に、このコマンドを使用してソース ファイルの内容をメモリのデザインに読み込むことができます。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

SystemVerilog には Verilog が含まれるので、`read_verilog` コマンドで両方のファイル タイプを読み込むことができますが、SystemVerilog ファイルでは SystemVerilog モードのコンパイルをイネーブルにするため、`read_verilog` コマンドで -sv オプションを指定する必要があります。このモードでは、ツールで SystemVerilog のキーワードおよび構文が認識されます。

1 つのプロジェクトに、Verilog ファイル (.v) と SystemVerilog ファイル (.sv) の両方、および VHDL ファイル (`read_vhdl` を使用) を含めることができます。ツールで合成用にこれらのファイルをコンパイルする際に、各ファイル タイプにそれぞれコンパイル ユニットが作成され、同じタイプのファイルと一緒にコンパイルされます。

引数

`-library <arg>` (オプション): Verilog ファイルが参照するライブラリを指定します。デフォルトの Verilog ライブラリは `xil_defaultlib` です。Vivado 合成では、ライブラリ名は無視されます。

`-sv` (オプション): ファイルを SystemVerilog コンパイル グループとして読み込みます。

注記: SystemVerilog には Verilog が含まれるので、Verilog ソース ファイルにユーザー定義名として SystemVerilog の予約キーワードが使用されていないければ、`-sv` オプションを使用して Verilog ファイルを読み込んでも SystemVerilog コンパイル モードがイネーブルになります。ただし、SystemVerilog ファイルを `-sv` を使用せずに Verilog コンパイル ユニットとして追加することはできません。

`<files>` (必須): 読み込む Verilog ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、指定した Verilog ファイルを読み込んで、ソース ファイルセットに追加しています。

```
read_verilog C:/Data/FPGA_Design/new_module.v
```

次の例では、SystemVerilog ファイルと Verilog ファイルに 1 つずつコンパイル ユニットを作成しています。

```
read_verilog -sv { file1.sv file2.sv file3.sv }
read_verilog { file1.v file2.v file3.v }
```

関連項目

- [add_files](#)
- [read_vhdl](#)
- [remove_files](#)

read_vhdl

1 つまたは複数の VHDL ファイルを読み込みます。

構文

```
read_vhdl -library <arg> [-vhdl2008] [-quiet] [-verbose] <files>
```

戻り値

追加されたファイル オブジェクトのリスト

使用法

名前	説明
-library	VHDL library
[-vhdl2008]	VHDL ファイルがバージョン 2008 であることを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	VHDL ファイル名を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

VHDL ソース ファイルを読み込みます。このコマンドは `add_files` コマンドと似ており、VHDL ファイルを読み込むと、ソース ファイルセットに追加されます。-library オプションを指定すると、ファイルはライブラリ プロパティを設定して追加されます。

Vivado ツールを非プロジェクト モードで実行しており、プロジェクト ソース ファイルを管理するプロジェクト ファイルがない場合に、このコマンドを使用してソース ファイルの内容をメモリのデザインに読み込むことができます。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

引数

-library <arg> (オプション): VHDL ファイルが参照するライブラリを指定します。デフォルトの VHDL ライブラリは `xil_defaultlib` です。

-vhdl2008 (オプション): ファイルをバージョン 2008 の VHDL ファイルとして読み込みます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): 読み込む VHDL ファイルの名前を 1 つまたは複数指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、指定した VHDL ファイルを読み込んで、ソース ファイルセットに追加しています。

```
read_vhdl C:/Data/FPGA_Design/new_module.vhdl
```

次の例では、指定した複数の VHDL 2008 ファイルを読み込んでいます。

```
read_vhdl -vhdl2008 {file1.vhd file2.vhd file3.vhd}
```

関連項目

- [add_files](#)
- [read_verilog](#)
- [remove_files](#)

read_xdc

1 つまたは複数のファイルから物理制約とタイミング制約を読み込みます。

構文

```
read_xdc [-cells <args>] [-ref <arg>] [-quiet_diff_pairs] [-mode <arg>]
         [-unmanaged] [-no_add] [-quiet] [-verbose] <files>
```

戻り値

ファイルのリスト

使用法

名前	説明
[-cells]	制約をインポートするセルを指定します。
[-ref]	制約をインポートする参照を指定します。
[-quiet_diff_pairs]	I/O ポートをインポートする際に、差動ペアの推論に関する警告メッセージを表示しないようにします。
[-mode]	制約ファイルを最上位から独立させてインポートするかどうかを指定します。有効な値は default、out_of_context で、デフォルトは default です。
[-unmanaged]	指定したファイルをツールで管理されない制約ファイルとして処理します。
[-no_add]	ファイルを制約ファイルセットに追加しません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	読み込む入力ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ザイリンクス デザイン制約 (XDC) から物理制約およびタイミング制約をインポートします。XDC をデザイン階層の `current_instance` レベル (デフォルトではデザインの最上位) にインポートするか、指定のセルにインポートできます。最上位にインポートすると、指定した XDC ファイルがアクティブな制約ファイルセットに追加されます。



重要: XDC からの制約は、同じ名前の現在の制約を上書きします。このため、XDC を読み込む際には重要な制約が上書きされないよう注意が必要です。

このコマンドは `add_files` コマンドと似ており、XDC はローカルプロジェクト ディレクトリにインポートされるのではなく、参照されて追加されます。

Vivado ツールを非プロジェクト モードで実行しており、プロジェクト ソース ファイルを管理するプロジェクト ファイルがない場合に、このコマンドを使用してソース ファイルの内容をメモリのデザインに読み込むことができます。非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

引数

`-cells <args>` (オプション): XDC ファイルの制約を指定のインスタンスに適用します。制約は指定したセル インスタンスにのみ適用され、XDC ファイルはアクティブな制約ファイルセットには追加されません。



ヒント: `-cells` オプションを指定する場合は、デザインを開いておく必要があります。

`-ref <arg>` (オプション): XDC ファイルから制約を読み込み、現在のデザインのどこにインスタンス化されているかにかかわらず、このオプションで指定したモジュールのすべてのインスタンスに適用します。

`-quiet_diff_pairs` (オプション): I/O 制約をインポートする際に、差動ペアの推論に関する警告メッセージを表示しないようにします。

`-mode [default | out_of_context]` (オプション): 指定した制約ファイルを最上位デザインのコンテキストでインポートするか、階層モジュールまたは IP コアの出力ファイルを生成する際の最上位から独立させてインポートするかを指定します。アウト オブ コンテキスト デザイン フローの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。



重要: 最上位から独立させたアウト オブ コンテキスト制約は、指定のセルまたはセル インスタンスに追加する必要があります。

`-unmanaged` (オプション): 追加されたファイルをツールで管理されない Tcl 制約ファイルとして処理します。制約を変更した場合、ツールでこれらのファイルに変更が自動的に保存されることはありません。ツールで管理されない Tcl 制約の詳細は、『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903) を参照してください。

`-no_add` (オプション): ファイルから制約を読み込み、インメモリ デザインに統合しますが、XDC ファイルを現在の制約セットには追加しません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): インポートする XDC ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例では、XDC ファイルを読み込み、現在のデザインに適用しています。

```
read_xdc file_1.xdc
```

次の例では、XDC ファイルを読み込み、現在のデザインに含まれる指定したモジュールのすべてのインスタンスに適用しています。

```
read_xdc -ref hex2led file_2.xdc
```

次の例では、XDC ファイルを読み込み、指定したモジュール内の指定したインスタンスのみに適用しています。

```
read_xdc -ref sixty -cells lsbcount file_3.xdc
```

次の例では、XDC ファイルを読み込み、異なるモジュールのインスタンスであっても、指定したインスタンスに適用しています。

```
read_xdc -cells {one_decode sixty/msbcount} file_4.xdc
```

注記: 複数のセルは、ダブル クォーテーション (") または波かっこ ({}) で囲む必要があります。

関連項目

- [add_files](#)
- [current_instance](#)
- [infer_diff_pairs](#)
- [write_xdc](#)

readback_hw_cfgmem

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトからデータをリードバックします。

構文

```
readback_hw_cfgmem [-checksum] [-force] [-all] [-offset <arg>] -file <arg>
                    [-format <arg>] [-datacount <arg>] [-quiet] [-verbose] [<hw_cfgmem>...]
```

使用法

名前	説明
[-checksum]	チェックサムをリードバックおよび計算します。-file オプションと共に使用することはできません。
[-force]	ファイルを強制的に記述します。
[-all]	すべてのメモリ ロケーションのリードバックを指定します。
[-offset]	メモリのオフセット値を指定します。デフォルトは 0x0 です。
-file	リードバックされたデータを書き込むファイルを指定します。
[-format]	リードバック ファイルのフォーマットを指定します。
[-datacount]	リードバックするデータ ユニットの数を指定します。デフォルトは 0x0 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_cfgmem>]	ハードウェア コンフィギュレーション メモリを指定します。デフォルトは現在のハードウェア コンフィギュレーション メモリです。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

hw_cfgmem オブジェクトとして指定されたハードウェア コンフィギュレーション メモリ デバイスからプログラムデータをリードバックします。

このコマンドは、program_hw_cfgmem コマンドでフラッシュ メモリ デバイスにプログラムされたメモリ コンフィギュレーション ファイル データをリードバックし、指定のファイルに書き込みます。メモリ コンフィギュレーション ファイルは、write_cfgmem コマンドにより作成され、ビットストリーム (.bit) ファイルおよび指定のデータ ファイルをメモリ コンフィギュレーション ファイル フォーマットに統合します。

リードバックは、ビットストリームおよび追加のデータ ファイルがフラッシュ メモリ デバイスに正しくプログラムされていることを検証するために、コンフィギュレーション メモリ デバイスからデータを読み出すプロセスです。

引数

-checksum (オプション): デバイスのビットストリームのチェックサムを計算します。

-force (オプション): 指定したファイルと同じ名前のファイルが存在する場合に上書きします。

-all (オプション): コンフィギュレーション メモリ デバイスのすべてのアドレス ロケーションをリードバックします。



ヒント: デフォルトでは、指定した hw_cfgmem オブジェクトのコンフィギュレーション メモリ ファイル (PROGRAM.FILE) で定義されたアドレスのみがリードバックされます。ただし、これは -offset および -datacount オプションに影響されます。

-offset <arg> (オプション): リードバックを開始するメモリ アドレス オフセット値を指定します。デフォルト オフセットは 0x0 です。

-file <arg> (必須): hw_cfgmem オブジェクトからリードバックされたデータを保存するファイルを指定します。リードバック ファイルは、write_cfgmem コマンドで作成される MCS ファイルと類似しています。ファイル名の拡張子は、その内容のフォーマットに応じて .mcs または .bin にする必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-format [mcs | bin] (オプション): 作成するリードバック ファイルのフォーマットを指定します。デフォルト フォーマットは MCS です。

-datacount <arg> (オプション): リードバックするバイト数を指定します。デフォルトでは、-offset オプションで指定されたアドレスからすべてのデータがリードバックされます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_cfgmem> (オプション): データをリードバックする hw_cfgmem オブジェクトを指定します。HW_CFGMEM は、get_hw_cfgmems または current_hw_cfgmem コマンドを使用してオブジェクトとして指定する必要があります。hw_cfgmem オプションを指定しない場合、current_hw_cfgmem が使用されます。

例

次の例では、hw_cfgmem オブジェクトを作成して現在のハードウェア デバイス (current_hw_device) に関連付け、write_cfgmem コマンドを使用してビットストリームから作成されたメモリ コンフィギュレーション ファイル (PROGRAM.FILE) を定義するプロパティを設定し、リードバック プロセスで使用する hw_cfgmem オブジェクトのほかのプロパティを設定し、現在の hw_device を cfgmem オブジェクトでプログラムしています。

```
create_hw_cfgmem -hw_device [current_hw_device] \
    [lindex [get_cfgmem_parts {n25q128-3.3v-spi-x1-x2-x4}] 0]
set_cfgMem [current_hw_cfgmem]
set_property PROGRAM.FILE {C:/Data/config_n25q128.mcs} $cfgMem
set_property PROGRAM.ADDRESS_RANGE {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK 1 $cfgMem
set_property PROGRAM.ERASE 1 $cfgMem
```

```
set_property PROGRAM.CFG_PROGRAM 1 $cfgMem
set_property PROGRAM.VERIFY 1 $cfgMem
create_hw_bitstream -hw_device [current_hw_device] \
    [get_property PROGRAM.HW_CFGMEM_BITFILE [current_hw_device]]
program_hw_devices [current_hw_device]
```

注記: 上記の例では、hw_cfgmem オブジェクトが変数 \$cfgMem に割り当てられます。

次の例では、現在の hw_cfgmem オブジェクトを PROGRAM.FILE プロパティで定義されたアドレスを使用してリードバックします。

```
readback_hw_cfgmem -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

次の例では、現在の hw_cfgmem オブジェクトから、アドレス 0 から最大メモリ ワード数までのすべてのアドレスをリードバックしています。

```
readback_hw_cfgmem -all -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

次の例では、現在の hw_cfgmem オブジェクトから選択した範囲のアドレスをリードバックしています。

```
readback_hw_cfgmem -offset 0x084 -datacount 100 -format mcs \
    -file C:/Data/design1.mcs [current_hw_cfgmem]
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_cfgmem](#)
- [delete_hw_cfgmem](#)
- [get_cfgmem_parts](#)
- [get_hw_cfgmems](#)
- [get_property](#)
- [program_hw_cfgmem](#)
- [write_cfgmem](#)

readback_hw_device

ハードウェア デバイスをリードバックします。

構文

```
readback_hw_device [-force] [-capture] [-readback_file <arg>]  
                   [-bin_file <arg>] [-quiet] [-verbose] [<hw_device>...]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
[-force]	ファイルを強制的に記述します。
[-capture]	コンフィギュレーション リードバック データをキャプチャします (UltraScale デバイスのみ)。
[-readback_file]	リードバック ファイル (RBD) を記述します。
[-bin_file]	バイナリ ファイル (BIN) を記述します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_device>]	ハードウェア デバイスを指定します。デフォルトは現在のハードウェア デバイスです。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア デバイスからビットストリーム データを読み出し、指定したリードバックまたはバイナリ ファイルに記述します。

Vivado デバイス プログラマにより、開いているターゲットを介してサイリンクス デバイスからビットストリーム データがリードバックされます。



重要: ハードウェア デバイス (hw_device) のビットストリームが暗号化されている場合は、リードバックできません。

このコマンドを実行すると、作成されたリードバック ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-force (オプション): 指定したファイルと同じ名前のファイルが存在する場合に上書きします。

`-capture` (オプション): UltraScale デバイスのみでコンフィギュレーション データのリードバック キャプチャをイネーブルにします。リードバック キャプチャは、CLB レジスタ、ブロック RAM、分散 RAM、SRL などのユーザー ステート エレメントの内容を判断するために使用できます。

`-readback_file <arg>` (オプション): 指定の `hw_device` から読み出したビットストリームの ASCII 出力であるリードバック ファイル (RDB) を記述します。このファイルは `write_bitstream -raw_bitfile` コマンドで生成された ASCII ファイルと似ていますが、ヘッダーはありません。



重要: `-readback_file` および `-bin_file` はどちらもオプションですが、いずれかまたは両方を指定する必要があります。そうしないと、結果を記述する出力ファイルがありません。

`-bin_file <arg>` (オプション): 指定の `hw_device` からリードバックされたプログラム データを含むバイナリ ファイル (BIN) を記述します。このファイルは `write_bitstream -bin_file` コマンドで生成されたバイナリ ファイルと似ています。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_device>` (オプション): プログラム データを読み出す `hw_device` を指定します。`hw_device` は、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア デバイスを指定しない場合、現在のハードウェア デバイス (`current_hw_device`) が使用されます。

例

次の例では、現在のハードウェア デバイスからリードバックされたビットストリームの ASCII ファイルを記述しています。

```
readback_hw_device -readback_file C:/Data/readback_1.rdb
[current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [get_property](#)
- [open_hw_target](#)
- [program_hw_devices](#)
- [write_bitstream](#)

redo

取り消されたコマンドをやり直します。

構文

```
redo [-list] [-quiet] [-verbose]
```

戻り値

やり直し可能なタスクのリスト (-list を使用した場合)

使用法

名前	説明
<code>[-list]</code>	やり直し可能なタスクのリストを返します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明



重要: UNDO および REDO コマンドは Vivado IDE で使用するためのもので、Tcl スクリプトでデザインの以前のステートを復元するために使用することはお勧めしません。デザインを特定の状態に戻すには、`write_checkpoint` コマンドを使用してデザイン チェックポイントを保存し、`read_checkpoint` コマンドを使用して復元できるようにします。

取り消されたコマンドをやり直します。このコマンドを繰り返し使用し、一連のコマンドをやり直すことができます。

`startgroup` および `endgroup` コマンドを使用してコマンド グループを作成した場合、`redo` コマンドでコマンド グループがシーケンスとしてやり直されます。

引数

`-list` (オプション): やり直すことができるコマンドのリストを取得します。`undo` コマンドを使用すると、コマンドのリストを順にさかのぼってコマンドが取り消されます。`redo` コマンドを使用すると、これらのコマンドをやり直すことができます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、やり直し可能なコマンドのリストが返されます。

```
redo -list
```

関連項目

- [undo](#)
- [startgroup](#)
- [endgroup](#)

refresh_design

現在のデザインを最新情報に更新します。

構文

```
refresh_design [-part <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-part]	ターゲット パーツを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

ハード ドライブのプロジェクト データから現在のデザインを読み込み直します。これにより、メモリのデザインが上書きされ、保存されていない変更は失われます。

引数

-part <arg> (オプション): デザインを読み込み直したときに使用する新しいターゲット パーツを指定します。これにより、ハード ドライブのプロジェクト データで指定されている制約ファイルのパーツが上書きされます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、ハード ディスクのプロジェクト データから現在のデザインを読み込み直します。これにより、メモリにある未保存の変更が上書きされます。

```
refresh_design
```

注記: デザインに対する一連の変更を取り消して、デザインを保存されている状態に戻します。

次の例では、指定した Virtex-6 パーツをターゲット デバイスとして使用して現在のデザインを更新しています。2 つ目のコマンドは、選択したパーツをアクティブなインプリメンテーション run のターゲット デバイスにするために必要です。

```
refresh_design -part xc6vcx75tff784-1  
set_property part xc6vcx75tff784-1 [get_runs impl_6]
```

注記: ターゲット パーツを変更しない場合は、2 つ目のコマンドは必要ありません。

関連項目

- [set_property](#)

refresh_hw_axi

ハードウェア AXI オブジェクトのステータスを更新します。

構文

```
refresh_hw_axi [-quiet] [-verbose] [<hw_axis>...]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_axis>]	ハードウェア AXI オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

ハードウェア AXI (hw_axi) の STATUS プロパティを現在のハードウェア デバイス (hw_device) からの値で更新します。

ハードウェア デバイス上にある JTAG to AXI MASTER のステータス レジスタからの値を読み出し、その値をハードウェア マネージャーの hw_axi オブジェクトの適切なプロパティに入力します。

指定の hw_axi オブジェクトの STATUS プロパティを更新します。STATUS プロパティには、STATUS.AXI_READ_BUSY、STATUS.AXI_READ_DONE、STATUS.AXI_WRITE_BUSY、STATUS.AXI_WRITE_DONE、STATUS.BRESP、および STATUS.RRESP が含まれます。

このコマンドを実行すると、hw_axi オブジェクトのプロパティがアップデートされますが、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_axis> (必須): 更新する hw_axi オブジェクトを指定します。hw_axi は、get_hw_axis コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、hw_axi オブジェクトの STATUS プロパティを更新しています。

```
refresh_hw_axi [get_hw_axis]
```

関連項目

- [delete_hw_axi_txn](#)
- [get_hw_axis](#)
- [get_hw_axi_txns](#)
- [reset_hw_axi](#)

refresh_hw_ddrmc

現在のハードウェア オブジェクトのステータスを更新します。DDRMC オブジェクトを指定する必要があります。少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

構文

```
refresh_hw_ddrmc [-regexp] [-properties <args>] [-quiet] [-verbose]
                  <hw_objects>
```

使用法

名前	説明
[-regexp]	プロパティを正規表現を使用して指定します。
[-properties]	更新するプロパティを指定します。デフォルトは、オブジェクトのすべてのプロパティです。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_objects>	ハードウェア DDRMC オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

すべてのプロパティ、指定したプロパティ、または指定した Versal DDR メモリ コントローラー (DDRMC) オブジェクトのインメモリの内容を、現在のハードウェア デバイスからの値で更新します。

ハードウェア デバイス上にインプリメントされている DDRMC オブジェクトから値を読み出し、その値を Vivado ハードウェア マネージャーの hw_ddrmc オブジェクトの適切なプロパティに入力します。プロパティを指定した場合、それらのプロパティがハードウェア デバイスからの値でアップデートされます。

少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

このコマンドを実行すると、hw_ddrmc オブジェクトのプロパティがアップデートされますが、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 更新するプロパティを正規表現を使用して指定します。

-properties <args> (オプション): hw_ddrmc オブジェクトの指定したプロパティを更新します。デフォルトでは、プロパティを指定しない場合は指定したオブジェクトのすべてのプロパティがハードウェア デバイス上の現在の値で更新されます。



ヒント: 存在しないプロパティを指定した場合は、そのプロパティは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_objects> (必須): コマンドを実行する DDRMC オブジェクト (hw_ddrmc) を指定します。

例

次の例は、Vivado ハードウェア マネージャーの DDRMC オブジェクトのすべてのプロパティを現在のハードウェアデバイスからのプロパティ値で更新します。

```
refresh_hw_ddrmc -properties {PHY_RANKS} [lindex [get_hw_ddrmcs] 3]
```

関連項目

- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_ddrmcs](#)
- [report_hw_ddrmc](#)

refresh_hw_device

ハードウェア デバイスを更新します。デバイスからデバイスおよびコア情報を読み込みます。

構文

```
refresh_hw_device [-update_hw_probes <arg>] [-disable_done_check]
                  [-force_poll] [-quiet] [-verbose] [<hw_device>]
```

使用法

名前	説明
[-update_hw_probes]	プローブ ファイルから読み込んでハードウェア プローブ情報をアップデートします。
[-disable_done_check]	デバイス更新の DONE チェックをディスエーブルにします。
[-force_poll]	すべてのターゲットを強制的にポーリングします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_device>]	ハードウェア デバイスを指定します。デフォルトは、現在のハードウェア デバイスです。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア デバイス オブジェクト上のデバッグ コアおよび IBERT コアをスキャンすることによりデバイスのインメモリ表示を更新し、指定した場合はプローブ ファイルも読み込みます。

Vivado Design Suite のハードウェア マネージャーは、デバイスにあるコアの情報および ILA および VIO デバッグ コアの場合はプローブ ファイルからの情報に基づいて、ハードウェア ILA、ハードウェア VIO、ハードウェア SIO、ハードウェア AXI オブジェクトを作成、削除、またはアップデートします。

program_hw_devices コマンドの後に refresh_hw_device コマンドを使用し、インメモリのハードウェア デバッグ オブジェクトが物理デバイス上の実際のコアのステートと一致するようにします。

引数

-update_hw_probes <arg> (オプション): 指定のプローブ ファイルを読み込むんで、指定のデバイスに関連付けられているプローブ ファイル(.ltx) をアップデートします。

注記: プローブ ファイルは、set_property コマンドを使用して PROBES.FILE プロパティを定義することにより、ハードウェア デバイス オブジェクトに関連付けます。

-disable_done_check (オプション): DONE チェックをディスエーブルにし、DONE ピンが Low の場合でもハードウェア マネージャーをイネーブルにして現在のデバイスのデバッグ コアがスキャンされるようにします。コンフィギュレーション シーケンスが完了すると、DONE ピンが High になります。DONE チェックは、コンフィギュレーションの完了を確認するのに DONE ピンが High になるのを待機します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_device>` (オプション): 更新するハードウェア デバイス オブジェクトを指定します。`hw_device` は、`get_hw_devices` または `current_hw_device` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア デバイスを指定しない場合、現在のハードウェア デバイス (`current_hw_device`) が返されます。

例

次の例では、指定したハードウェア デバイスを更新しています。

```
refresh_hw_device [lindex [get_hw_devices] 0]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [program_hw_devices](#)

refresh_hw_hbm

現在のハードウェア オブジェクトのステータスを更新します。HBM またはデバイス ハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

構文

```
refresh_hw_hbm [-regexp] [-properties <args>] [-quiet] [-verbose]
               <hw_objects>
```

使用法

名前	説明
<code>[-regexp]</code>	プロパティを正規表現を使用して指定します。
<code>[-properties]</code>	更新するプロパティを指定します。デフォルトは、オブジェクトのすべてのプロパティです。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定の HBM コントローラー オブジェクトのすべてのプロパティまたは指定したプロパティのインメモリ ビューを、現在のハードウェア デバイスからの値で更新します。

ハードウェア デバイス上にインプリメントされている HBM コントローラーから値を読み出し、その値を Vivado ハードウェア マネージャーの hw_hbm オブジェクトの適切なプロパティに入力します。プロパティを指定した場合、それらのプロパティがハードウェア デバイスからの値でアップデートされます。

少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

このコマンドを実行すると、hw_hbm オブジェクトのプロパティがアップデートされますが、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 更新するプロパティを正規表現を使用して指定します。

`-properties <args>` (オプション): hw_hbm オブジェクトの指定したプロパティを更新します。デフォルトでは、プロパティを指定しない場合は指定したオブジェクトのすべてのプロパティがハードウェア デバイス上の現在の値で更新されます。



ヒント: 存在しないプロパティを指定した場合は、そのプロパティは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_objects> (必須): 実行する HBM アクティビティ モニター オブジェクト (hw_hbm) を指定します。

例

次の例は、Vivado ハードウェア マネージャーの HBM コントローラーのすべてのプロパティを現在のハードウェア デバイスからのプロパティ 値で更新します。

```
refresh_hw_hbm [lindex [get_hw_hbms] 1]
```

次の例は、HBM コントローラーの指定したプロパティを現在のハードウェア デバイスからの値で更新します。

```
refresh_hw_hbm -properties {MC2.INIT.AM.REPEAT_EN} [get_hw_hbms *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)
- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

refresh_hw_mig

現在のハードウェア オブジェクトのステータスを更新します。任意の MIG、デバイス、ターゲット、またはサーバーハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

構文

```
refresh_hw_mig [-regexp] [-properties <args>] [-quiet] [-verbose]  
               <hw_objects>
```

使用法

名前	説明
<code>[-regexp]</code>	プロパティを正規表現を使用して指定します。
<code>[-properties]</code>	更新するプロパティを指定します。デフォルトは、オブジェクトのすべてのプロパティです。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア MIG (hw_mig) オブジェクトのすべてのプロパティまたは指定したプロパティのインメモリ ビューを、現在のハードウェア デバイスからの値で更新します。

ハードウェア デバイス上にインプリメントされているメモリ コントローラーからの値を読み出し、Vivado ロジック解析またはスタンドアロンの Vivado Lab Edition のその値を hw_mig デバッグ コアの適切なプロパティに入力します。

少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

このコマンドを実行すると、hw_mig オブジェクトのプロパティがアップデートされますが、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 更新するプロパティを正規表現を使用して指定します。

`-properties <args>` (オプション): hw_mig オブジェクトの指定したプロパティを更新します。デフォルトでは、プロパティを指定しない場合は指定したオブジェクトのすべてのプロパティがハードウェア デバイス上の現在の値で更新されます。



ヒント: 存在しないプロパティを指定した場合は、そのプロパティは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (オプション): `hw_mig`、ハードウェア デバイス (`hw_device`)、ハードウェア ターゲット (`hw_target`)、またはハードウェア サーバー (`hw_server`) オブジェクトを指定します。

注記: オブジェクトは、名前で指定するのではなく、`get_hw_migs` などの `get_hw_XXX` コマンドを使用して指定する必要があります。

例

次の例では、Vivado ロジック解析のメモリ IP のプロパティを現在のハードウェア デバイスからのプロパティで更新しています。

```
refresh_hw_mig [lindex [get_hw_migs] 0]
```

関連項目

- [commit_hw_mig](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_migs](#)
- [implement_mig_cores](#)
- [set_property](#)

refresh_hw_server

ハードウェア サーバーへの接続を更新します。

構文

```
refresh_hw_server [-force_poll] [-quiet] [-verbose] [<hw_server>]
```

使用法

名前	説明
<code>[-force_poll]</code>	すべてのターゲットを強制的にポーリングします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_server>]</code>	ハードウェア サーバー

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

現在または指定のハードウェア サーバーへの接続を更新または再度開きます。

このコマンドを実行すると、ハードウェア サーバーからの接続メッセージが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_server>` (オプション): 更新するハードウェア サーバー (`hw_server`) を指定します。ハードウェア サーバーを指定しない場合、現在のハードウェア サーバー (`current_hw_server`) が更新されます。ハードウェア サーバーは、名前で指定するか、あるいは `get_hw_servers` または `current_hw_server` コマンドで 1 つの `hw_server` オブジェクトを返すことにより指定します。

例

次の例では、current_hw_server を更新しています。

```
refresh_hw_server
```

次の例では、接続されているすべてのハードウェア サーバーを更新しています。サーバーを更新する前に、ホスト名が表示されます。

```
foreach x [get_hw_server] {puts "Refreshing Host $x"; refresh_hw_server $x}
```

関連項目

- [connect_hw_server](#)
- [current_hw_server](#)
- [disconnect_hw_server](#)
- [get_hw_servers](#)

refresh_hw_sio

指定のハードウェア オブジェクトのステータスを更新します。任意のシリアル I/O (スキャンまたはスイープ以外)、デバイス、ターゲット、またはサーバー ハードウェア オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

構文

```
refresh_hw_sio [-regexp] [-properties <args>] [-quiet] [-verbose]
               <hw_objects>
```

使用法

名前	説明
<code>[-regexp]</code>	プロパティを正規表現を使用して指定します。
<code>[-properties]</code>	更新するプロパティを指定します。デフォルトは、オブジェクトのすべてのプロパティです。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア SIO (hw_sio) オブジェクトのすべてのプロパティまたは指定したプロパティのインメモリ ビューを、ハードウェア デバイス上の実際のオブジェクトからの値で更新します。

オブジェクトには、GT、RX、TX、PLL、または Common など、ハードウェア SIO スキャン (hw_sio_scan) およびハードウェア SIO スイープ (hw_sio_sweep) オブジェクトを除く任意のシリアル I/O オブジェクトを指定できます。SIO オブジェクトには、デバイス、ターゲット、またはサーバー ハードウェア オブジェクトも含まれます。

refresh_hw_sio コマンドは、デバイス上にある指定したオブジェクトの値を読み出し、その値をハードウェア マネージャーの IBERT コアに関連するプロパティに適用します。

このコマンドを実行すると、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-regexp` (オプション): 更新するプロパティを正規表現を使用して指定します。

`-properties <args>` (オプション): hw_sio オブジェクトの指定したプロパティを更新します。デフォルトでは、プロパティを指定しない場合は指定したオブジェクトのすべてのプロパティがハードウェア デバイス上の現在の値で更新されます。



ヒント: 存在しないプロパティを指定した場合は、そのプロパティは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_objects> (必須): 更新するハードウェア SIO (`hw_sio`) オブジェクトを 1 つ以上指定します。`hw_sio` は、`get_hw_sio_*` コマンドのいずれかを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したハードウェア SIO GT (`hw_sio_gt`) オブジェクトのすべてのプロパティを更新しています。

```
refresh_hw_sio [get_hw_sio_gts *MGT_X0Y11]
```

次の例では、現在のハードウェア デバイス上にあるすべてのハードウェア SIO (`hw_sio`) オブジェクトのすべてのプロパティを更新しています。

```
refresh_hw_sio [current_hw_device]
```

関連項目

- [current_hw_device](#)
- [get_hw_devices](#)
- [get_hw_servers](#)
- [get_hw_sio_commons](#)
- [get_hw_sio_gts](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_plls](#)
- [get_hw_sio_txs](#)
- [get_hw_targets](#)
- [report_property](#)

refresh_hw_sysmon

現在のハードウェア オブジェクトのステータスを更新します。ハードウェア サーバー (hw_server)、ハードウェア ターゲット (hw_target)、ハードウェア デバイス (hw_device)、またはハードウェア システム モニター (hw_sysmon) オブジェクトを入力できます。少なくとも 1 つのオブジェクトが必要です。オブジェクトに存在しないプロパティを指定した場合、そのプロパティは更新されません。

構文

```
refresh_hw_sysmon [-regexp] [-properties <args>] [-quiet] [-verbose]
                  <hw_objects>
```

使用法

名前	説明
[-regexp]	プロパティを正規表現を使用して指定します。
[-properties]	更新するプロパティを指定します。デフォルトは、オブジェクトのすべてのプロパティです。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_objects>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

ハードウェア システム モニター (hw_sysmon) のプロパティを現在のハードウェア デバイス (hw_device) 上のシステム モニター (XADC) からの値で更新します。

ハードウェア デバイス上にあるシステム モニターのステータス レジスタからの値を読み出し、その値をハードウェア マネージャーの hw_sysmon オブジェクトの適切なプロパティに入力します。



ヒント: hw_sysmon オブジェクトは、オブジェクトの SYSMON_REFRESH_RATE_MS で指定されたレートで自動的に更新されます。

このコマンドを実行すると、hw_sysmon オブジェクトのプロパティがアップデートされますが、正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-regexp (オプション): 更新するプロパティを正規表現を使用して指定します。

-properties <args> (オプション): hw_sysmon オブジェクトの指定したプロパティを更新します。デフォルトでは、プロパティを指定しない場合は指定したオブジェクトのすべてのプロパティがハードウェア デバイス上の現在の値で更新されます。



ヒント: 存在しないプロパティを指定した場合は、そのプロパティは無視されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_objects> (オプション): プロパティを `hw_device` からの値で更新するシステム モニターを指定します。システム モニター オブジェクトは `hw_sysmon` オブジェクトとして指定するか、関連付けられているハードウェア デバイス (`hw_device`)、ハードウェア ターゲット (`hw_target`)、またはハードウェア サーバー (`hw_server`) オブジェクトによりシステム モニターとして指定できます。

注記: オブジェクトは、名前で指定するのではなく、`get_hw_sysmon` などの `get_hw_XXX` コマンドを使用して指定する必要があります。

例

次の例では、ハードウェア システム モニターの TEMPERATURE プロパティを現在のデバイスの実際の温度で更新しています。

```
refresh_hw_sysmon -properties {TEMPERATURE} [lindex [get_hw_sysmons] 0]
```

関連項目

- [commit_hw_sysmon](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_sysmons](#)
- [set_hw_sysmon_reg](#)
- [set_property](#)

refresh_hw_target

ハードウェア ターゲットを更新します。

構文

```
refresh_hw_target [-force_poll] [-quiet] [-verbose] [<hw_target>]
```

使用法

名前	説明
<code>[-force_poll]</code>	すべてのターゲットを強制的にポーリングします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_target>]</code>	ハードウェア ターゲット

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア サーバー上の指定したハードウェア ターゲットへの接続を更新し、Vivado Design Suite のハードウェア マネージャーにハードウェア ターゲット (hw_target) オブジェクトを読み込み直します。hw_target オブジェクトを指定しない場合、現在のハードウェア ターゲット (current_hw_target) オブジェクトが更新されます。

ハードウェア ターゲットとは、ビットストリーム ファイルを使用してプログラム、またはデザインをデバッグするために使用する、1 つ以上のサイリンクス デバイスから構成される JTAG チェーンを含むシステム ボードです。システム ボード上のハードウェア ターゲットと Vivado Design Suite との接続は、connect_hw_server コマンドを使用して開き、サイリンクス ハードウェア サーバー アプリケーションで制御します。サポートされる JTAG ダウンロード ケーブルおよびデバイスのリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

各ハードウェア ターゲットには、プログラムまたはデバッグ目的で使用するサイリンクス デバイスを 1 つまたは複数含めることができます。current_hw_device コマンドは、現在のデバイスを指定するか、返します。現在のハードウェア ターゲットを指定したら、open_hw_target コマンドを使用して、そのハードウェア ターゲットを介してサイリンクス FPGA への接続を開くことができます。

refresh_hw_target コマンドはハードウェア ターゲット上のデバイスをスキャンし、ターゲットを介して使用可能なハードウェア デバイス (hw_device) を作成、削除、またはアップデートします。使用可能なデバイスを取得するには、get_hw_devices コマンドを使用します。

このコマンドを実行すると、更新プロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_target>` (オプション): 接続を更新する `hw_target` オブジェクトを指定します。`hw_target` は、`get_hw_targets` または `current_hw_target` コマンドを使用してオブジェクトとして指定する必要があります。ターゲットを指定しない場合は、現在のハードウェア ターゲット (`current_hw_target`) への接続が更新されます。

例

次の例では、現在のハードウェア ターゲットを更新しています。

```
refresh_hw_target
```

関連項目

- [connect_hw_server](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [get_hw_servers](#)
- [open_hw_target](#)

refresh_hw_vio

ハードウェア プロブの INPUT_VALUE および ACTIVITY_VALUE プロパティをハードウェア VIO コアから読み出した値でアップデートします。

構文

```
refresh_hw_vio [-update_output_values] [-quiet] [-verbose] <hw_vios>...
```

使用法

名前	説明
[-update_output_values]	ハードウェア プロブの OUTPUT_VALUE プロパティを VIO コアから読み出した値でアップデートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_vios>	ハードウェア VIO オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定した VIO デバッグ コアの入力プロブの INPUT_VALUE および ACTIVITY_VALUE プロパティを、ハードウェア デバイス上のハードウェア VIO (hw_vio) コアから読み出された値にアップデートします。

VIO (Virtual Input/Output) デバッグ コアは、サイリンクス デバイスにプログラムされている内部信号をリアルタイムで監視および駆動できます。VIO コアは、ハードウェア プロブ (hw_probe) オブジェクトを使用してデバイス上の信号を監視および駆動します。入力プロブは、VIO コアへの入力として信号を監視します。出力プロブは、VIO コアから信号を指定の値に駆動します。

refresh_hw_vio コマンドは、デバイス上にある VIO デバッグ コアの入力プロブの信号値を読み出し、hw_probe オブジェクトの INPUT_VALUE プロパティにその値を適用して、そのプロブの ACTIVITY_VALUE プロパティをアップデートします。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-update_output_values (オプション): ハードウェア プロブの OUTPUT_VALUE プロパティを指定した VIO デバッグ コアの信号から読み出した値でアップデートします。これはブール値のオプションであり、使用するとイネーブルになります。このオプションを使用するとプロブの OUTPUT_VALUE が hw_vio デバッグ コアの信号値と一致するよう更新されます。



ヒント: これは、プロブの OUTPUT_VALUE をデバイス上の hw_vio デバッグ コアに適用する commit_hw_vio コマンドとは逆です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_vios> (必須): 更新する `hw_vio` オブジェクトを 1 つ以上指定します。`hw_vio` は、`get_hw_vios` コマンドを使用してオブジェクトとして指定するか、名前で指定できます。

例

次の例では、名前で指定した `hw_vio` デバッグ コアを更新しています。ハードウェア プローブ (`hw_probe`) オブジェクトの `OUTPUT_VALUE` プロパティもアップデートしています。

```
refresh_hw_vio -update_output_values hw_vio_1
```

関連項目

- [commit_hw_vio](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [program_hw_devices](#)
- [reset_hw_vio_activity](#)
- [reset_hw_vio_outputs](#)
- [set_property](#)

refresh_meminit

ブロック RAM の初期化文字列を ELF ファイルの内容でアップデートおよび初期化します。

構文

```
refresh_meminit [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Project \(プロジェクト\)](#)

regenerate_bd_layout

レイアウトを再生成します。

構文

```
regenerate_bd_layout [-hierarchy <arg>] [-layout_file <arg>] [-routing]
                     [-quiet] [-verbose]
```

使用法

名前	説明
[-hierarchy]	ウィンドウへの階層パスを指定します。
[-layout_file]	write_bd_layout を使用してネイティブ フォーマットでエクスポートされたレイアウト ファイルを指定します。
[-routing]	ブロックの配置を保持して配線を再生成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

現在の IP サブシステム デザインのレイアウトを開いているキャンバスに再生成します。このコマンドは、Vivado IDE でサブシステム デザインのグラフィカル エlementをアップデートおよび再描画します。

引数

-hierarchy <arg> (オプション): 再生成する階層モジュールを指定します。階層モジュールをオブジェクトとして指定するには、get_bd_cells コマンドを使用します。

-layout_file <arg> (オプション): write_bd_layout コマンドを使用して作成したネイティブ フォーマットのブロック デザインのレイアウト ファイルを指定します。

-routing (オプション): IP インテグレーター キャンバスの配線を更新しますが、オブジェクトの配置は更新しません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、Vivado IDE の IP インテグレーター キャンバスを更新しています。

```
regenerate_bd_layout
```

次の例では、現在のブロック デザインの指定した階層モジュールを更新しています。

```
regenerate_bd_layout -hierarchy [get_bd_cell myHier1]
```

関連項目

- [current_bd_design](#)
- [open_bd_design](#)
- [start_gui](#)
- [write_bd_layout](#)

register_proc

Tcl プロシージャを Vivado に登録します。

構文

```
register_proc [-quiet] [-verbose] <proc> [<tasknm>]
```

戻り値

なし

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<proc>	登録するプロシージャの名前を指定します。プロシージャは、Tcl に対して既知のものである必要があります。
[<tasknm>]	プロシージャのラッパーとなる Tcl タスクの名前を指定します。デフォルトでは、プロシージャのルート名を使用してプロシージャが登録されます (名前空間なし)。

カテゴリ

[Tools \(ツール\)](#)

説明

Tcl プロシージャ (proc) を Vivado Tcl コマンド インタープリターに登録し、Vivado Design Suite ヘルプ システムに組み込みます。

次に、Vivado Design Suite で使用できるように定義された Tcl プロシージャの例を示します。

```
proc findCommand {option} {
    # Summary:
    # Searches through all Vivado Tcl commands for commands implementing
    # the specified argument.
    # Argument Usage:
    # option: Specifies the argument to search for.
    # Return Value:
    # Returns a list of Tcl commands that implement the option.
    # Categories: personal

    foreach cmd [lsort [info commands *]]
    {
        catch {
            if {[regexp "$option" [help -syntax $cmd]]}
        }
        puts $cmd
    }
}
```

```

    }
  }
} :
# End

```

で始まる行は、Vivado Design Suite ヘルプ システムに登録されたコマンドのヘルプ テキストを定義します。

- # Summary: コマンドの簡単な説明を示します。
- # Argument Usage: プロシージャの引数とその説明をリストします。
- # Return Value: プロシージャを実行したときに返される値を示します。
- # Categories: 登録されたプロシージャのカテゴリを指定します。

プロシージャを Tcl コマンドとして登録すると、次のコマンドを実行したときに指定したヘルプ テキストが返されます。

```

tasknm -help
-or-
help tasknm

```

このコマンドを実行すると、登録されたプロシージャの名前が返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<proc> (必須): 現在の Vivado Design Suite セッションに読み込む Tcl プロシージャの名前を指定します。登録する前に、Tcl プロシージャを定義して Vivado Design Suite に読み込んでおく必要があります。

<tasknm> (オプション): 次に、Vivado Design Suite で使用する、プロシージャのラッパーとなる Tcl コマンド名を指定します。デフォルトでは、プロシージャのルート名を使用してプロシージャが登録されます。

例

次の例では、findCommand という Tcl プロシージャを findCmd という名前の Tcl コマンドとして登録しています。

```

register_proc findCommand findCmd

```

関連項目

- [unregister_proc](#)

reimport_files

最新でないファイルを再インポートします。

構文

```
reimport_files [-force] [-quiet] [-verbose] [<files>...]
```

戻り値

インポートされたファイル オブジェクトのリスト

使用法

名前	説明
[-force]	ローカル ファイルの方が新しい場合でも再インポートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<files>]	再インポートするファイルを指定します。ファイルを指定しない場合は、プロジェクトの最新でないファイルがすべて再インポートされます。

カテゴリ

[Project \(プロジェクト\)](#)

説明

プロジェクト ファイルを再インポートします。これにより、参照先のソース ファイルからローカルのプロジェクト ファイルがアップデートされます。

引数

-force (オプション): ローカルのプロジェクト ファイルの方が参照先のソース ファイルよりも新しい場合でも、ファイルを再インポートします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<files> (オプション): 再インポートするファイルを指定します。ファイルを指定しない場合は、プロジェクトの最新でないファイルがすべて再インポートされます。-force を使用してファイルを指定しない場合は、プロジェクトのすべてのファイルが再インポートされます。

例

次の例では、ファイルが最新でないか、ローカル ファイルが参照先ソース ファイルよりも新しいかどうかにかかわらず、すべてのプロジェクト ファイルが再インポートされます。

```
reimport_files -force
```

注記: 上書きされるローカル ファイルの方が新しくても、警告メッセージは表示されません。

次の例では、元のソース ファイルがローカル プロジェクト ファイルよりも新しい場合にのみ、指定のファイルをプロジェクトに再インポートします。

```
reimport_files C:/Data/FPGA_Design/source1.v \  
C:/Data/FPGA_Design/source2.vhdl
```

関連項目

- [add_files](#)
- [import_files](#)

relaunch_sim

コンパイル オプションを変更せずにデザインを再コンパイルし、現在のシミュレーションを再開します。

構文

```
relaunch_sim [-quiet] [-verbose]
```

戻り値

現在のシミュレーション オブジェクト

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレータを再起動し、アップデートされたデザインを解析および検証します。

`relaunch_sim` コマンドは現在のシミュレーションを一時停止し、現在のデザインを新しいシミュレーション スナップショットに再コンパイルして、現在のシミュレーションを新しいスナップショットに接続し、シミュレーションを再開します。

通常の HDL デバッグ サイクルでは、デザインをシミュレーション スナップショットにコンパイルし、シミュレーションを起動して、Vivado シミュレータ IDE の波形ビューアーに表示する信号、スコープ、オブジェクトを設定します。コードまたはテストベンチに問題が見つかった場合は、デザインを修正し、再コンパイルしてシミュレータを再起動します。

このコマンドを使用すると、開いている波形、コード、スコープ、オブジェクト ウィンドウの設定などの Vivado シミュレータの設定を保持しながら、デザインを再コンパイルしてシミュレータを再起動できます。



重要: `relaunch_sim` コマンドは Vivado Design Suite IDE で実行中のシミュレーションに対してのみ適用され、スタンドアロンまたはバッチ Vivado シミュレータ実行には適用されません。

このコマンドを実行すると、実行されたプロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のシミュレーションを再実行しています。

```
relaunch_sim
```

関連項目

- [close_sim](#)
- [current_sim](#)
- [launch_simulation](#)
- [xsim](#)

remove_bps

シミュレーションからブレークポイントを削除します。

構文

```
remove_bps [-all] [-file <arg>] [-line <arg>] [-quiet] [-verbose]  
[<BreakPointObjsOrIds>...]
```

使用法

名前	説明
[-all]	すべてのブレークポイントを削除します。
[-file]	ブレークポイントを削除するファイルを指定します。
[-line]	ブレークポイントを削除する行番号を指定します。デフォルトは -1 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<BreakPointObjsOrIds>]	削除するブレークポイント オブジェクトまたはブレークポイント オブジェクト ID を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションから指定のブレークポイントを削除します。このコマンドを使用するには、シミュレーションを開いている必要があります。

ブレークポイントは、ユーザーの指定するソース コード内の停止地点で、デザインをデバッグする際に使用できます。ブレークポイントを含むデザインをシミュレーションすると、各ブレークポイントでシミュレーションが停止し、値および動作を確認できます。

現在のシミュレーションで設定されているブレークポイントは、`report_bps` コマンドを使用してレポートできます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-all` (オプション): 現在のシミュレーションのブレークポイントをすべて削除します。



重要: このオプションを使用すると、ほかのオプションを指定していても、警告なしですべてのブレークポイントが削除されます。

`-file <arg>` (オプション): 指定した HDL ソース ファイルの既存のブレークポイントを削除します。

`-line <arg>` (オプション): HDL ソース ファイルの指定した行番号に設定されている既存のブレークポイントを削除します。

注記: 既存の 1 つのブレークポイントを定義するには、`-file` および `-line` を両方使用する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<BreakPointObjsOrIds>` (オプション): 現在のシミュレーションの 1 つまたは複数のブレークポイントを指定します。ブレークポイント オブジェクトは、`add_bp` コマンドでシミュレーションにブレークポイントを追加したときに返されます。

例

次の例では、現在のシミュレーションからすべてのブレークポイントを削除しています。

```
remove_bps -all
```

関連項目

- [report_bps](#)

remove_cell

現在のデザインからセルを削除します。

構文

```
remove_cell [-quiet] [-verbose] <cells>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><cells></code>	削除するセルの名前を指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

開いている合成済みデザインまたはインプリメント済みデザインの現在のネットリストからセルを削除します。

注記: セルをライブラリ マクロ (マクロ プリミティブとも呼ばれる) から削除することはできません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cells>` (必須): 削除するセルを指定します。インスタンス名は、デザインの最上位からの階層名で指定できます。この場合、階層インスタンス名に階層区切り文字を使用する必要があります。現在の階層区切り文字を確認するには、`get_hierarchy_separator` コマンドを使用します。

例

次の例では、現在のデザインのメモリにあるネットリストから fftEngine を削除します。

```
remove_cell fftEngine  
remove_cell usbEngine0/usb_out
```

関連項目

- [create_cell](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

remove_cells_from_pblock

Pblock からセルを削除します。

構文

```
remove_cells_from_pblock [-quiet] [-verbose] <pblock> <cells>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><pblock></code>	セルを削除する Pblock を指定します。
<code><cells></code>	削除するセルを指定します。

カテゴリ

Floorplan (フロアプラン)、XDC

説明

指定したロジック インスタンスを Pblock から削除します。Pblock にセルを追加するには、`add_cells_to_pblock` コマンドを使用します。

注記: Pblock からセルを削除しても、配置が解除されるわけではありません。LOC 制約はそのまま残ります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pblock>` (必須): 指定のインスタンスを削除する Pblock の名前を指定します。

`<cells>` (必須): 指定の Pblock から削除する 1 つまたは複数のセル オブジェクトを指定します。

例

次の例では、`pb_cpuEngine` Pblock から指定したセルを削除しています。

```
remove_cells_from_pblock pb_cpuEngine [get_cells cpuEngine/cpu_dwb_dat_o/*]
```

関連項目

- [add_cells_to_pblock](#)

remove_conditions

シミュレーションから条件を削除します。名前は Tcl glob パターンを使用して指定できます。

構文

```
remove_conditions [-all] [-quiet] [-verbose] [<ConditionObjs>]
```

使用法

名前	説明
[-all]	すべての条件を削除します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<ConditionObjs>]	条件オブジェクトを ID または名前で指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションから指定の条件を削除します。このコマンドを使用するには、シミュレーションを開いている必要があります。

条件はシミュレーションの開始前に定義できます。条件を追加すると、シミュレータで信号の変化が検出されるたびに条件式が評価されます。指定の条件が真になると、条件コマンドが実行されます。

現在のシミュレーションで設定されている条件は、`report_conditions` コマンドを使用してレポートできます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-all` (オプション): 現在のシミュレーションの条件をすべて削除します。



重要: このオプションを使用すると、ほかのオプションを指定していても、警告なしですべての条件が削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<ConditionObjs>: 現在のシミュレーションから削除する条件の識別子を 1 つ以上指定します。条件識別子は、`add_condition` コマンドで条件を定義したときに返されます。

例

次の例では、現在のシミュレーションから指定の条件を削除しています。

```
remove_conditions condition3
```

関連項目

- [add_condition](#)
- [report_conditions](#)

remove_drc_checks

ユーザー ルール デックから DRC ルール チェック オブジェクトを削除します。

構文

```
remove_drc_checks [-of_objects <args>] [-regexp] [-nocase] [-filter <arg>]
                  -ruleddeck <arg> [-quiet] [-verbose] [<patterns>]
```

戻り値

drc_check オブジェクト

使用法

名前	説明
[-of_objects]	指定した drc_ruledeck の rule_check オブジェクトを追加します。
[-regexp]	検索パターンを正規表現で指定します。
[-nocase]	検索パターンの大文字/小文字を区別せずに検索します (-regexp を指定した場合のみ有効)。
[-filter]	式を使用してリストをフィルター処理します。
-ruleddeck	変更する DRC ルール デックを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<patterns>]	rule_check オブジェクトを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[DRC、Object \(オブジェクト\)](#)

説明

DRC ルール デック (drc_ruledeck) オブジェクトから指定のデザイン ルール チェックを削除します。

ルール デックとは、デザイン ルール チェックのグループで、I/O プランニングなどの FPGA デザイン フローの異なる段階で report_drc コマンドにより実行されます。ツールには定義済みのルール デックが含まれていますが、create_drc_ruledeck コマンドを使用して新しいユーザー定義のルール デックを作成できます。

ルール デックにチェックを追加するには、add_drc_checks コマンドを使用します。

DRC ルール チェック オブジェクトには IS_ENABLED プロパティがあり、set_property コマンドを使用して true または false に設定できます。新しいルール チェックを作成すると、IS_ENABLED プロパティはデフォルトで true に設定されます。IS_ENABLED プロパティを false に設定すると、ルール デックからルールを削除せずに、report_drc を実行したときにルール チェックが使用されないようにすることができます。



ヒント: DRC ルールとそのプロパティをデフォルト設定に戻すには、reset_drc_check コマンドを使用します。

このコマンドを実行すると、指定のルール デックから削除されたデザイン ルール チェックのリストが返されます。

引数

`-of_objects <arg>` (オプション): 指定したルール デックから指定の `drc_ruledeck` オブジェクトのルール チェックを削除します。このオプションを使用すると、ターゲット ルール デックからの 1 つのルール デックからすべてのルール チェックが削除されます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-regexp` (オプション): 検索パターンが正規表現で記述されていることを指定します。このオプションを使用する場合、検索パターン (`<patterns>`) および `-filter` オプションの式は正規表現で記述する必要があります。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-nocase` (オプション): 検索パターンの大文字/小文字を区別せずに検索します。このオプションは、`-regexp` オプションを使用した場合にのみ適用されます。

`-filter <args>` (オプション): 結果のリストに指定した式のフィルターを適用します。`-filter` オプションを使用すると、検索パターンにより返されるオブジェクトのリストに、指定したプロパティ 値に基づくフィルターを適用できます。オブジェクトに設定されているプロパティは、`report_property` または `list_property` コマンドで確認できます。

フィルター検索パターンにはクォーテーションを使用し、ネット、ピン、セル名、またはほかのプロパティに含まれる特殊文字のエスケープ処理を回避してください。文字列比較では大文字と小文字が区別され、常に検索文字列の冒頭および末尾にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、プロパティ 値の部分文字列を含めるよう検索を拡張できます。

注記: フィルターでは、オブジェクトに指定のプロパティが存在し、指定の検索パターンがオブジェクトのプロパティ 値に一致すると、そのオブジェクトが返されます。ワイルドカード (*) を使用すると、定義値が "" のプロパティに一致します。

文字列比較では、フィルター式に使用できる演算子は等価 (==)、不等価 (!=)、一致 (==)、不一致 (!~) です。数値比較演算子 <、>、<=、および >= も使用できます。複数のフィルター式を AND (&&) および OR (||) で組み合わせることもできます。次の例では、名前に文字列「RESET」が含まれない入力ピンを取得しています。

```
get_pins * -filter {DIRECTION == IN && NAME !~ "RESET"}
```

ブール型 (bool) プロパティでは、フィルター式が真か偽かを直接評価できます。

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

`-ruledeck <arg>` (必須): 指定のデザイン ルール チェックを削除するルール デックの名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<patterns>` (オプション): ルール デックから削除するデザイン ルール チェックを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのルール チェックが指定のルール デックから削除されます。複数の検索パターンを指定して、異なる検索条件に基づいてルール チェックを削除できます。

注記: 複数の検索パターンは波かっこ {} またはダブルクォーテーション (") で囲み、1 つのエレメントとして指定します。

例

次の例では、指定した検索パターンに一致するルール チェックを `my_rules` ルール デックから削除しています。

```
remove_drc_checks -filter {GROUP == AVAL} -ruleddeck my_rules
```

次の例では、指定の DRC チェックが実行されないように指定しています。

```
set_property IS_ENABLED FALSE [get_drc_checks RAMW-1]
```

次の例では、指定したルール デックからすべてのルール チェックを削除しています。

```
remove_drc_checks -ruleddeck my_rules
```

関連項目

- [add_drc_checks](#)
- [create_drc_check](#)
- [create_drc_ruleddeck](#)
- [get_drc_checks](#)
- [get_drc_ruledecks](#)
- [list_property](#)
- [report_drc](#)
- [report_property](#)
- [reset_drc_check](#)

remove_files

ファイルセットからファイルまたはディレクトリを削除します。

構文

```
remove_files [-fileset <arg>] [-quiet] [-verbose] <files>...
```

戻り値

削除されたファイルのリスト

使用法

名前	説明
[-fileset]	ファイルセット名を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	削除するファイルの名前を指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[Simulation \(シミュレーション\)](#)

説明

現在のファイルセットまたは指定のファイルセットから指定のファイル オブジェクトを削除します。ファイルは現在のプロジェクトから削除されますが、ディスクからは削除されません。

ファイルはファイル名の文字列で指定するか、`get_files` コマンドを使用してファイル オブジェクトとして指定します。文字列として指定すると、現在のファイルセットまたは別のファイルセットでファイルが検索されます。`get_files` コマンドでファイル オブジェクトを指定すると、ファイルセットはそのオブジェクトにより定義され、`-fileset` オプションは無視されます。

このコマンドが正常に実行された場合は、何も返されません。指定のファイルが見つからない場合はエラーが返されます。

引数

`-fileset <arg>` (オプション): 指定のファイルを検索するファイルセットの名前を指定します。デフォルトでは、`current_fileset` コマンドで定義された現在のファイルセットからファイルが削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<files> (必須): プロジェクトから削除するファイルの名前を指定します。

注記: ファイルを指定しない場合、ファイルは削除されません。

例

次の例では、制約セット `constrs_1` から `C:/Design/top.xdc` というファイルを削除しています。

```
remove_files -fileset constrs_1 C:/Design/top.xdc
```

複数のファイルを削除する場合は、次のように指定します。

```
remove_files -fileset sim_1 top_tb1.vhdl top_tb2.vhdl
```

次の例では、現在のプロジェクトのファイル オブジェクトをすべて取得し、削除しています。

```
remove_files [get_files]
```



注意: これにより、デザインからすべてのファイルが削除されます。

関連項目

- [add_files](#)
- [current_fileset](#)
- [get_files](#)

remove_forces

add_force コマンドを使用して信号、ワイヤ、またはレジスタに適用している強制的な値の設定を解除します。

構文

```
remove_forces [-all] [-quiet] [-verbose] [<ForceObj>...]
```

使用法

名前	説明
[-all]	すべての force オブジェクトを削除します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<ForceObj>]	force オブジェクトまたは ID を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションの指定の force オブジェクトまたは force ID を削除します。

指定の HDL オブジェクトへの強制的な値の設定は、add_forces コマンドを使用して適用します。このコマンドは、現在のシミュレーションからこれらの値の強制を解除します。



重要: テストベンチまたはモジュールの HDL オブジェクトに force/release 文がある場合、それらの文より add_force コマンドが優先されます。remove_force コマンドを使用してこれらのオブジェクトを解除すると、通常の操作が再開し、Verilog force/release 文も有効になります。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-all (オプション): 現在のシミュレーションのすべての値の強制を解除します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<ForceObj> (オプション): 指定の force オブジェクトのみを削除します。add_force コマンドで値の強制を設定すると、force ID が返されます。

例

次の例では、`add_force` コマンドを使用して force オブジェクトを作成し、force ID を Tcl 変数に保存して、その force オブジェクトを削除しています。

```
set f10 [ add_force reset 1 300 ]  
remove_forces $f10
```

次の例では、現在のシミュレーションからすべての force オブジェクトを削除しています。

```
remove_forces -all
```

関連項目

- [get_objects](#)
- [add_force](#)

remove_gui_custom_command_args

1 つまたは複数のカスタム コマンド引数を削除します。

構文

```
remove_gui_custom_command_args -command_name <arg> [-quiet] [-verbose]
<names>...
```

使用法

名前	説明
-command_name	引数を削除するカスタム コマンドの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<names>	削除する 1 つまたは複数のカスタム コマンド引数の名前を指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

特定のカスタム GUI コマンドの 1 つまたは複数のカスタム GUI コマンド引数を削除します。

`get_gui_custom_commands` を使用すると、定義済みのカスタム コマンドを取得できます。
`get_gui_custom_command_args` を使用すると、特定の GUI カスタム コマンドの GUI カスタム コマンド引数のリストを取得できます。

引数

-command_name (必須): 引数を削除する GUI カスタム コマンドの名前を指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<names> (必須): 削除する引数の名前を 1 つ以上指定します。

例

次の例は、cmd_1 というカスタム GUI コマンドの arg1 および arg2 という引数を削除します。

```
remove_gui_custom_command_args -command_name cmd_1 {arg1 arg2}
```

関連項目

- [create_gui_custom_command](#)
- [create_gui_custom_command_arg](#)
- [get_gui_custom_command_args](#)
- [get_gui_custom_commands](#)

remove_gui_custom_commands

1 つまたは複数のカスタム コマンドを削除します。

構文

```
remove_gui_custom_commands [-quiet] [-verbose] <names>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><names></code>	削除する 1 つまたは複数のカスタム コマンドの名前を指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

1 つまたは複数の GUI カスタム コマンドを削除します。

`get_gui_custom_commands` を使用すると、定義済みのカスタム コマンドを取得できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<names>` (必須): 削除するコマンドの名前を 1 つ以上指定します。

例

次の例は、`abc` および `xyz` という名前の GUI カスタム コマンドを削除します。

```
remove_gui_custom_commands {abc xyz}
```

関連項目

- [create_gui_custom_command](#)

- [create_gui_custom_command_arg](#)
- [get_gui_custom_command_args](#)
- [get_gui_custom_commands](#)

remove_hw_hbm_pc

指定したハードウェア HBM のアクティビティ モニターから擬似チャネルの選択を解除します。まずメモリ コントローラーの番号を指定し、その後に擬似チャネル番号を指定します。

構文

```
remove_hw_hbm_pc [-quiet] [-verbose] <mc_num> <pc_num> <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><mc_num></code>	選択を解除するメモリ コントローラーの番号 (0 ～ 7) を指定します。
<code><pc_num></code>	選択を解除する擬似チャネルの番号 (0 または 1) を指定します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

`remove_hw_hbm_pc` コマンドは、`add_hw_hbm_pc` コマンドを使用して HBM アクティビティ モニターに追加したメモリ チャネル (mc)/擬似チャネル (pc) を削除します。



ヒント: 擬似チャネルを追加または削除するときには、HBM アクティビティ モニターが実行されていないことを確認してください。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<mc_num>` (必須): HBM コントローラーの 8 つのメモリ チャネル (0 ～ 7) のいずれかを指定します。

`<pc_num>` (必須): 2 つの擬似チャネル (0 または 1) のいずれかを指定します。

<hw_objects> (必須): 擬似チャネルを削除する hw_hbm を指定します。

例

次の例では、HBM アクティビティ モニターから追加されていた擬似チャネルを削除し、アクティビティ モニターを実行しています。

```
remove_hw_hbm_pc 2 0 [get_hw_hbms *HBM_2]  
run_hw_hbm_amon [get_hw_hbms *HBM_2]
```

関連項目

- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

remove_hw_probe_enum

hw_probe の列挙から列挙名と値のペアを削除します。

構文

```
remove_hw_probe_enum [-no_gui_update] [-list <args>] [-remove_all] [-quiet]
                    [-verbose] <hw_probe>
```

使用法

名前	説明
[-no_gui_update]	GUI をアップデートしません。
[-list]	削除する列挙名のリストを指定します。
[-remove_all]	ハードウェア プローブの列挙すべてを削除します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_probe>	ILA ハードウェア プローブ オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

指定したハードウェア プローブ (hw_probe) オブジェクトから列挙名と値のペアを削除します。

列挙名 (ENUM プロパティ) は、add_hw_probe_enum コマンドを使用して hw_probe オブジェクトに追加されます。このコマンドは、それらの定義済みプロパティを削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-no_gui_update (オプション): プローブの列挙値を削除するために Vivado ロジック解析の GUI をアップデートしません。

-list <args> (オプション): 指定した <hw_probe> オブジェクトから指定した列挙名のリストを削除します。



ヒント: 名前のリストは、リスト オブジェクトとして指定するか、単純な名前のリストとして指定できます。

-remove_all (オプション): 指定した <hw_probe> オブジェクトで定義されているすべての ENUM プロパティを削除します。このオプションを -list オプションと共に使用することはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_probe>` (必須): ENUM プロパティを削除する `hw_probe` オブジェクトを指定します。

例

次の例では、指定した `hw_probe` オブジェクトから列挙名のリストを削除しています。

```
remove_hw_probe_enum -list {WHITE YELLOW GREY} \
[get_hw_probes op1 -of_objects [current_hw_ila]]
```

関連項目

- [add_hw_probe_enum](#)
- [current_hw_device](#)
- [current_hw_ila](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [report_property](#)

remove_hw_sio_link

既存のハードウェア SIO リンクを削除します。

構文

```
remove_hw_sio_link [-quiet] [-verbose] <hw_sio_links>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_links></code>	ハードウェア SIO リンク

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア デバイスで定義されている IBERT デバッグ コアの GT 上の TX オブジェクトと RX オブジェクト間の指定した通信リンクを削除します。

Vivado シリアル I/O 解析はリンク ベースの解析機能であり、IBERT デザイン内の任意のトランスミッターとレシーバーをリンクできます。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。このコマンドは、これらのリンクを削除します。

このコマンドを実行すると、IBERT デバッグ コアのリンク オブジェクトのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_links>` (必須): 削除する `hw_sio_link` オブジェクトを 1 つ以上指定します。`hw_sio_link` は、`create_hw_sio_link` または `get_hw_sio_links` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、IBERT デバッグ コアの指定した通信リンクを削除しています。

```
remove_hw_sio_link [get_hw_sio_links -filter {DESCRIPTION == "Link2"}]
```

関連項目

- [create_hw_sio_link](#)
- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_linkgroups](#)

remove_hw_sio_linkgroup

既存のハードウェア SIO リンク グループを削除します。

構文

```
remove_hw_sio_linkgroup [-quiet] [-verbose] <hw_sio_linkgroups>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_linkgroups></code>	ハードウェア SIO リンク グループを指定します。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア デバイスで定義されている IBERT デバッグ コアの GT 上の TX オブジェクトと RX オブジェクト間の通信リンクを関連付ける指定したグループを削除します。

Vivado シリアル I/O 解析機能はリンク ベースです。リンクは、デバイス上のギガビット トランシーバーのトランスミッターとレシーバー間の通信パスとプロトコルを定義します。リンク グループ (hw_sio_linkgroup) オブジェクトを使用すると、複数のリンクにまとめてプロパティを設定したりスキャンを実行するため、リンクをグループとして関連付けることができます。



ヒント: remove_hw_sio_linkgroup コマンドは、指定した関連性を削除するだけで、通信リンク自体は削除しません。通信リンク オブジェクトを削除するには、remove_hw_sio_link コマンドを使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

`<hw_sio_linkgroups>` (必須): 削除する hw_sio_linkgroup オブジェクトを 1 つ以上指定します。hw_sio_linkgroup は、create_hw_sio_linkgroup または get_hw_sio_linkgroups コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したリンク グループを削除しています。

```
remove_hw_sio_linkgroup [get_hw_sio_linkgroups {LINKGROUP_0}]
```

関連項目

- [create_hw_sio_link](#)
- [create_hw_sio_linkgroup](#)
- [current_hw_device](#)
- [get_hw_sio_iberts](#)
- [get_hw_sio_links](#)
- [get_hw_sio_linkgroups](#)

remove_hw_sio_scan

既存のハードウェア SIO スキャンを削除します。

構文

```
remove_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_scans></code>	ハードウェア SIO スキャン

カテゴリ

Hardware (ハードウェア)

説明

指定したシリアル I/O 解析スキャン オブジェクトを削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_scans>` (必須): 削除するハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトを 1 つ以上指定します。`hw_sio_scan` は、`create_hw_sio_scan` または `get_hw_sio_scans` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定した SIO スキャンを削除しています。

```
remove_hw_sio_scan [get_hw_sio_scans {SCAN_2}]
```

関連項目

- [create_hw_sio_scan](#)
- [get_hw_sio_scans](#)
- [run_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

remove_hw_sio_sweep

既存のハードウェア SIO スイープを削除します。

構文

```
remove_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_sweeps></code>	ハードウェア SIO スイープ

カテゴリ

Hardware (ハードウェア)

説明

指定したシリアル I/O 解析スイープ スキャン オブジェクトを削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_sweeps>` (必須): 削除するハードウェア SIO スイープ (`hw_sio_sweep`) オブジェクトを 1 つ以上指定します。`hw_sio_sweep` は、`create_hw_sio_sweep` または `get_hw_sio_sweeps` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したスイープ スキャン オブジェクトを削除しています。

```
remove_hw_sio_sweep [get_hw_sio_sweeps {SWEEP_3}]
```

関連項目

- [create_hw_sio_sweep](#)
- [get_hw_sio_sweeps](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

remove_net

現在のデザインからネットを削除します。

構文

```
remove_net [-prune] [-quiet] [-verbose] <nets>...
```

使用法

名前	説明
<code>[-prune]</code>	ネットの削除を実行する際に、remove_net コマンドの実行により未接続となったピンおよびポートを削除します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><nets></code>	削除するネットを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

開いている合成済みデザインまたはインプリメント済みデザインのネットリストから指定のネットを削除します。

注記: ネットをライブラリ マクロ (マクロ プリミティブとも呼ばれる) から削除することはできません。

バスを削除する場合は、バス インデックスではなくバス名を指定する必要があります。これにより、バスに関連付けられている一部のビットだけではなく、バス全体を確実に削除できます。バスのビットを削除してサイズを変更するには、resize_net_bus コマンドを使用します。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、write_checkpoint コマンドを使用してデザイン チェックポイントとして保存するか、write_* コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

-prune (オプション): 指定したネットの削除により未接続になった階層ピン、ポート、またはネットを削除します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<nets> (必須): 現在のデザインのネットリストから削除するネットを指定します。

例

次のような接続ネットワークがあるとします。

```
leaf_cell11/pin1 > net1 > block1/pin1 >
  topnet
< block2/pin1 < net2 < leaf_cell12/pin1
```

次の例では、`block1/pin1`、`block2/pin1`、`net1`、および `net2` を削除しますが、最下位セルのピンは削除しません。

```
remove_net topnet -prune
```

次の例では、バス ネットの 1 ビットを削除しようとした場合に表示される警告メッセージを示し、その後にバスのルート名を指定してバス全体を削除しています。

```
remove_net DataIn_pad_1_i[0]
WARNING: [Coretc1-82] No nets matched 'DataIn_pad_1_i[0]'.
remove_net DataIn_pad_1_i
```

関連項目

- [create_net](#)
- [disconnect_net](#)
- [resize_net_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

remove_pin

現在のデザインからピンを削除します。

構文

```
remove_pin [-quiet] [-verbose] <pins>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><pins></code>	削除するピンを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

開いている合成済みデザインまたはインプリメント済みデザインの現在のネットリストからピンを削除します。

注記: ピンをライブラリ マクロまたはマクロ プリミティブから削除することはできません。

バス ピンを削除する場合は、バス インデックスではなくプライマリ ピン名を指定する必要があります。これにより、バスに関連付けられている一部のビットだけではなく、バス ピン全体を確実に削除できます。バス ピンのビットを削除してサイズを変更するには、`resize_pin_bus` コマンドを使用します。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<pins> (必須): ネットリストから削除するピンを指定します。ピンは、ピンが存在するセル インスタンスを基準に階層で指定する必要があります。

例

次の例では、現在のデザインのメモリにあるネットリストから指定ピンを削除しています。

```
remove_pin cpuEngine/inPin
```

関連項目

- [create_pin](#)

remove_port

ネットリストから指定した最上位ポートを削除します。

構文

```
remove_port [-quiet] [-verbose] <ports>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><ports></code>	削除するポートまたはバス ポートを指定します。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

指定したポートまたはバスを削除します。

バス ポートを削除する場合は、バス インデックスではなくプライマリ ポート名を指定する必要があります。これにより、バスに関連付けられている一部のビットだけではなく、バス ポート全体を確実に削除できます。バス ポートのビットを削除してサイズを変更するには、`resize_port_bus` コマンドを使用します。

`remove_port` コマンドでは `create_port` コマンドで追加されたポートは削除可能ですが、RTL またはネットリスト デザインで定義されているポートは削除できません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<ports>: 削除するポートの名前を 1 つ以上指定します。

例

次の例では、指定したポートが削除されます。

```
remove_port PORT0
```

次の例では、バスの指定した 2 つのポートが削除されます。

```
remove_port BUS[1] BUS[2]
```

次の例では、差動ペア ポートの N 側と P 側の両方が削除されます。

```
remove_port D_BUS_P[0]
```

注記: 差動ペアの N 側または P 側のみを削除した場合、もう一方も削除されます。

関連項目

- [create_cell](#)
- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [create_interface](#)
- [place_ports](#)
- [resize_port_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

remove_wave

現在の波形設定から波形オブジェクトを削除します。

構文

```
remove_wave [-of <args>] [-quiet] [-verbose] <items>...
```

使用法

名前	説明
[-of]	波形オブジェクトを検索する波形設定、グループ、または仮想バスを指定します。デフォルトは現在の波形設定です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<items>	削除する波形オブジェクトを指定します。

カテゴリ

[Waveform \(波形\)](#)

rename_cell

セルの名前を変更します。

構文

```
rename_cell -to <arg> [-quiet] [-verbose] <cell>...
```

使用法

名前	説明
-to	新しい名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<cell>	名前を変更するセルを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

現在の合成済みデザインまたはインプリメント済みデザインに含まれる 1 つの階層セルまたは最下位セルの名前を変更します。



ヒント: DONT_TOUCH プロパティが TRUE に設定されているセルの名前を変更することはできません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、write_checkpoint コマンドを使用してデザイン チェックポイントとして保存するか、write_* コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

セル、ネット、ピン、およびポートの名前を変更すると、インメモリ デザインで定義されているデザイン制約にも影響します。制約は新しいオブジェクト名をターゲットとするよう自動的に変更されますが、ソース XDC ファイルには記述されません。write_checkpoint コマンドを使用して変更したインメモリ デザインを保存すると、変更されたオブジェクト名および制約の両方が保存されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-to <arg> (必須): 指定のセルに割り当てる新しい名前を指定します。名前には、Tcl の特殊文字 (' \" \{ } ; \$ #) を含めることはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cell>` (必須): 名前を変更するセルのインスタンス名を指定します。

例

次の例では、`or1200_cpu` という階層セルの名前を変更しています。

```
rename_cell -to or1200_gpu or1200_cpu
```

関連項目

- [connect_net](#)
- [create_cell](#)
- [create_net](#)
- [remove_cell](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

rename_net

ネットの名前を変更します。

構文

```
rename_net -to <arg> [-quiet] [-verbose] <net>...
```

使用法

名前	説明
-to	新しい名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<net>	名前を変更するネットを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

現在の合成済みデザインまたはインプリメント済みデザインに含まれるネットの名前を変更します。

ネットの名前を変更する場合、次の制限があります。

- DONT_TOUCH または MARK_DEBUG プロパティが TRUE に設定されているネットの名前を変更することはできません。
- バス ネットの個々のビットの名前を変更することはできませんが、バス全体の名前を変更することは可能です。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

セル、ネット、ピン、およびポートの名前を変更すると、インメモリ デザインで定義されているデザイン制約にも影響します。制約は新しいオブジェクト名をターゲットとするよう自動的に変更されますが、ソース XDC ファイルには記述されません。`write_checkpoint` コマンドを使用して変更したインメモリ デザインを保存すると、変更されたオブジェクト名および制約の両方が保存されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-to <arg>` (必須): 指定のネットに割り当てる新しい名前を指定します。名前には、Tcl の特殊文字 (' '\{\};\$#) を含むことはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<net>` (必須): 名前を変更するネットを指定します。

例

次の例では、指定したバス信号の名前を変更しています。

```
rename_net -to dataOut dout
```

関連項目

- [connect_net](#)
- [create_cell](#)
- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [remove_cell](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

rename_pin

ピンの名前を変更します。

構文

```
rename_pin -to <arg> [-quiet] [-verbose] <pin>...
```

使用法

名前	説明
-to	新しい名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pin>	名前を変更するピンを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

現在の合成済みデザインまたはインプリメント済みデザインに含まれる階層セルの指定のピンの名前を変更します。

ピンの名前を変更する場合、次の制限があります。

- プリミティブ セルのピンの名前を変更することはできません。
- DONT_TOUCH プロパティが設定されている階層セルのピンの名前を変更することはできませんが、DONT_TOUCH プロパティが設定されているセルに含まれる階層セルのピンの名前は変更可能です。
- バス ピンの個々のビットの名前を変更することはできませんが、バス全体の名前を変更することは可能です。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

セル、ネット、ピン、およびポートの名前を変更すると、インメモリ デザインで定義されているデザイン制約にも影響します。制約は新しいオブジェクト名をターゲットとするよう自動的に変更されますが、ソース XDC ファイルには記述されません。`write_checkpoint` コマンドを使用して変更したインメモリ デザインを保存すると、変更されたオブジェクト名および制約の両方が保存されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-to <arg>` (必須): 指定のピンに割り当てる新しい名前を指定します。ピンの階層名ではなく、ピン名のみを指定します。名前には、Tcl の特殊文字 (' " \ { } ; \$ #) を含めることはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<pin> (必須): 名前を変更するピンの階層名 (セルのインスタンス名から開始) を指定します。

例

次の例では、指定したピンの名前を変更しています。

```
rename_pin -to in1 egressLoop[0].egressFifo/I1
```

次の例では、バスの 1 つのビットを変更しようとした場合に表示されるエラー メッセージを示し、その後にバス ピン全体の名前を変更しています。

```
rename_pin -to din[0] egressLoop[0].egressFifo/buffer_fifo/dataInput[0]
WARNING: [Coretc1 2-1480] rename_pin can not rename bits of a bus, \
use resize_pin_bus instead.
rename_pin -to dataInput egressLoop[0].egressFifo/buffer_fifo/din
```

関連項目

- [connect_net](#)
- [create_cell](#)
- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [remove_pin](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

rename_port

ポートの名前を変更します。

構文

```
rename_port -to <arg> [-quiet] [-verbose] <port>...
```

使用法

名前	説明
-to	新しい名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<port>	名前を変更するポートを指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

現在の合成済みデザインまたはインプリメント済みデザインに含まれる 1 つのポートの名前を変更します。



ヒント: バス ポートの個々のビットの名前を変更することはできませんが、バス全体の名前を変更することは可能です。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイル セットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

セル、ネット、ピン、およびポートの名前を変更すると、インメモリ デザインで定義されているデザイン制約にも影響します。制約は新しいオブジェクト名をターゲットとするよう自動的に変更されますが、ソース XDC ファイルには記述されません。`write_checkpoint` コマンドを使用して変更したインメモリ デザインを保存すると、変更されたオブジェクト名および制約の両方が保存されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-to <arg> (必須): 指定のポートに割り当てる新しい名前を指定します。名前には、Tcl の特殊文字 (' '\{\};\$#) を含むことはできません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<port>` (必須): 名前を変更するポートを指定します。

例

次の例では、指定したバス ポートの名前を変更しています。

```
rename_port -to wbInputData wbInDat
```

関連項目

- [connect_net](#)
- [create_net](#)
- [create_port](#)
- [remove_port](#)
- [set_hierarchy_separator](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

rename_ref

セル参照の名前を変更します。

構文

```
rename_ref [-ref <arg>] [-to <arg>] [-prefix_all <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-ref]	名前を変更するセル参照を指定します。
[-to]	新しい名前を指定します。
[-prefix_all]	現在のデザインで該当する階層セル参照すべての名前を変更します。新しい名前は、元の名前に指定の接頭辞を付けたものになります。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

1 つのプリミティブでないセルの参照名を変更するか、現在の合成済みデザインまたはインプリメント済みデザインのプリミティブでないすべてのセルに 参照接頭辞を付けます。

このコマンドを使用すると、現在のデザインのプリミティブでない参照名を変更し、別のデザインの参照名と競合しないようにすることができます。これにより、2 つのモジュールまたはデザインを、名前の競合を回避して、一緒に合成またはシミュレーションできます。

このコマンドを実行すると、1 つのセルの参照名が変更された場合は何も返されず、-prefix_all を使用した場合は名前が変更されたセルの数が返されます。コマンドが正常に実行できなかった場合は、エラーが返されます。

引数

-ref <arg> (オプション): プリミティブでないセルの現在の参照名を指定します。

-to <arg> (オプション): 参照名を指定した値に名前します。

-prefix_all <arg> (オプション): 最上位モジュールを除く現在のデザインのプリミティブでないセルすべての参照名に、指定の接頭辞を付けます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定の参照名を指定の値に変更しています。

```
rename_ref -ref usbf_top -to MOD1_usbf_top
```

次の例では、現在のデザインのプリミティブでないセルすべてに 指定の参照名接頭辞を付けています。

```
rename_ref -prefix_all MOD1_
```

関連項目

- [synth_design](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

reorder_files

アクティブなファイルセットでソース ファイルの順序を変更します。

構文

```
reorder_files [-fileset <arg>] [-before <arg>] [-after <arg>] [-front]
              [-back] [-auto] [-disable_unused] [-quiet] [-verbose] <files>...
```

使用法

名前	説明
[-fileset]	ファイルの順序を並べ替えるファイルセットを指定します。
[-before]	指定したファイルをこのファイルの前に移動します。
[-after]	指定したファイルをこのファイルの後に移動します。
[-front]	指定したファイルを一番前に移動します (デフォルト)。
[-back]	指定したファイルを一番後に移動します。
[-auto]	指定したファイルセットを自動的に並べ替えます。
[-disable_unused]	TOP デザイン ユニットに関連していないファイルをすべてデイスエーブルにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<files>	移動するファイルを指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

指定したファイルセットのソース ファイルを並べ替えます。指定したファイルを、ファイルセット内で一番前または一番後に移動したり、ほかのファイルの前後に移動したりできます。このコマンドには自動並べ替え機能もあり、現在の最上位モジュールの要件に基づいてファイルを並べ替えることもできます。

引数

-fileset <arg> (オプション): ファイルを並べ替えるファイルセットを指定します。デフォルトでは、sources_1 ソース ファイルセットが指定されます。

-before <arg> (オプション): 指定したファイルをファイルセットのこのファイルの前に配置します。ファイルは、そのファイルセットへの完全パスで指定する必要があります。

-after <arg> (オプション): 指定したファイルをファイルセットのこのファイルの後に配置します。ファイルは、そのファイルセットへの完全パスで指定する必要があります。

-front (オプション): 指定したファイルをファイルセットのファイル リストの一番前に配置します。

-back (オプション): 指定したファイルをファイルセットのファイル リストの一番後に配置します。

`-auto` (オプション): プロジェクトの現在の最上位モジュールの階層要件に基づいて、ファイルを自動的に並べ替えます。このオプションは、最上位モジュールを `set_property top` コマンドで変更した後によく使用されます。

`-disable_unused` (オプション): 最上位モジュールに基づく階層で現在使用されないファイルをすべてディスエーブルにします。このオプションは、最上位モジュールを `set_property top` コマンドで変更した後によく使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<files>` (必須): ファイルセット内で移動するファイルを 1 つまたは複数指定します。ファイルは、ファイルセットの完全パス名で指定する必要があり、指定した順に並べ替えられます。

例

次の例では、指定したファイルをソース ファイルセットの一番前に移動しています。

```
reorder_files -front {C:/Data/FPGA/file1.vhdl C:/Data/FPGA/file2.vhdl}
```

注記: `-fileset` オプションが指定されていないので、デフォルトのソース ファイルセットが使用されます。

次の例では、デザインで最上位モジュールを設定し、この最上位モジュールの階層に基づいてファイルを自動的に並べ替え、未使用のファイルをディスエーブルにしています。

```
set_property top block1 [current_fileset]
reorder_files -auto -disable_unused
```

関連項目

- [add_files](#)
- [create_fileset](#)
- [current_fileset](#)
- [remove_files](#)

replace_bd_cell

cell1 への接続を解除して cell2 に置き換え、解除した接続を cell2 に接続します。

構文

```
replace_bd_cell [-preserve_name] [-preserve_configuration] [-quiet]
                [-verbose] [<cell1>] [<cell2>...]
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-preserve_name]	cell2 の名前を cell1 の名前に変更し、cell1 の名前を cell1name_old に変更します。
[-preserve_configuration]	cell2 上の cell1 の設定を保持します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<cell1>]	接続を解除するセルを指定します。
[<cell2>]	cell1 から解除した接続を接続するセルを指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

1 つの IP インテグレーターに現在割り当てられている接続を、現在のデザインの別の IP インテグレーター セルに移動します。このコマンドは、ソース セルからターゲット セルに接続を移動することにより、セルを別のセルにすばやく置き換えるためのものです。

現在のセルまたは既存のセルがブロック デザイン内の現在の位置から移動され、その場所に新しいセルが配置されます。セル上のピンおよびインターフェイス ピンへの接続は可能な限り保持され、接続が削除された場合はクリティカル警告が表示されます。



重要: このコマンドは、UNDO コマンドでサポートされていません。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は TCL_ERROR が返されます。

引数

-preserve_name (オプション): 現在のセルの名前を、それを置換する新しいセルに適用します。既存のセルの名前は <instance>_old に変更されます。

`-preserve_configuration` (オプション): 現在のセルの設定を、それを置換する新しいセルに適用します。

注記: 元のセルの設定オプションで新しいセルに適用できないものは警告としてレポートされ、スキップされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cell1>` (必須): 接続を削除し、新しいセルに置き換える IP インテグレーター セルを指定します。セルは、名前で指定するか、または `get_bd_cells` コマンドで 1 つのブロック デザインのセル (`bd_cell`) オブジェクトを返すことにより指定します。

注記: セルは IP インテグレーター サブシステム デザインからは削除されるのではなく、新しいセルをそこに配置するために移動されるだけです。

`<cell2>`: 既存のセル (`<cell1>`) を置き換え、既存の接続を適用する IP インテグレーター セルを指定します。セルは、名前で指定するか、または `get_bd_cells` コマンドで 1 つのブロック デザインのセル (`bd_cell`) オブジェクトを返すことにより指定します。

例

次の例では、セル `lmb_v10_1` からの接続を、セル `lmb_bram_cntlr_1` 上の同じ名前のピンおよびインターフェイス ピンに移動します。

```
replace_bd_cell [get_bd_cells /lmb_v10_1] [get_bd_cells \
    /lmb_bram_if_cntlr_1]
CRITICAL WARNING: [BD 41-1164] The interface pin 'LMB_S1_0' with
bus definition 'xilinx.com:interface:lmb:1.0' is not found on the
cell '/lmb_bram_if_cntlr_1'. Its connection to the interface
net 'Conn' has been removed.
CRITICAL WARNING: [BD 41-1166] The pin 'SYS_Rst' is not found
on the cell '/lmb_bram_if_cntlr_1'. Its connection to the net
'sys_rst_1' has been removed.
0
```

注記: ピンの不一致により既存の接続が削除された場合は、クリティカル警告が返されます。

関連項目

- [create_bd_cell](#)
- [get_bd_cells](#)

report_bd_diffs

2 のブロック デザインの違いをレポートします。この Tcl コマンドでは、スタンドアロンの diffbd 実行ファイルではレポートされない IP .xii ファイルからのバス インターフェイス パラメーターも比較できます。

構文

```
report_bd_diffs [-format <arg>] [-file <arg>] [-open_html] [-brief]
                [-strict] [-fast] [-return_string] [-depth <arg>] [-crossprobe]
                [-repository <arg>] [-take_snapshot] [-diff_snapshot] [-quiet]
                [-verbose] <design1> <design2>
```

戻り値

機能的な違いがない場合は 0、違いがある場合は > 0、エラーがある場合は -1

使用法

名前	説明
[-format]	レポートのフォーマットを HTML (html) またはテキスト (text) に指定します。デフォルトは text です。
[-file]	出力ファイル名を指定します。*.htm* は、HTML フォーマットを指定します。
[-open_html]	HTML レポートをブラウザで開きます。フォーマットを HTML に設定します。
[-brief]	ファイルが異なるかどうかのみを出力します。レポートは生成しません。
[-strict]	機能的でない変更を機能的な変更として扱います。
[-fast]	可能な場合にインメモリの BD を作成する代わりに BD JSON を直接読み込みます。.XII ファイルからのバス インターフェイス パラメーターを比較しません (スタンドアロン diffbd と同じ)。
[-return_string]	(テキストのみ) レポートを文字列として返します。
[-depth]	等しいアイテムの HTML 表示深さを指定します。HTML のファイル サイズを制限するために使用します。等しくないアイテムには適用されません。HTML フォーマットを指定します。デフォルトは 4 です。
[-crossprobe]	HTML レポートで Vivado オブジェクトを選択するリンクをイネーブルにします。open_html を指定します。
[-repository]	ディスク上のデザインのユーザー リポジトリを指定します。
[-take_snapshot]	現在のブロックデザインのスナップショットを作成します。
[-diff_snapshot]	現在のブロック デザインとスナップショットを比較します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<design1>	比較する最初のブロック デザインの名前またはファイル パスを指定します。
<design2>	比較する 2 番目のブロック デザインの名前またはファイル パスを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

2 のブロック デザイン ファイル (.bd) の違いを解析してレポートします。



ヒント: このコマンドにはスタンドアロン バージョン (diffbd) もあり、コマンド ラインから実行できます。スタンドアロン バージョンの詳細は、コマンド ラインで「diffbd -h」と入力すると表示されます。詳細は、『Vivado Design Suite ユーザー ガイド: IP インテグレーターを使用した IP サブシステムの設計』(UG994) を参照してください。

このコマンドは、2 つのブロック デザインのグラフィックでない比較を実行します。リビジョン制御システム内からブロック デザインのリビジョン間の比較を実行できます。ブロック デザインは、current_bd_design または get_bd_designs コマンドを使用して BD オブジェクトとして指定する必要があります。デザイン オブジェクトの名前は同じでもかまいませんが、異なる .bd ファイルから返されたものである必要があります。指定した BD オブジェクトが同じデザインを参照している場合は、エラーが返されます。

レポートされる違いは、追加、ブロック図で使用されている IP の変更、デザイン プロパティまたはパラメーターの変更、デザイン階層の変更、接続の変更、およびメモリ アドレス指定の変更です。

このコマンドを実行すると、指定したブロック デザインの違いを示すレポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-format [text | html] (オプション): 違いレポートのフォーマットをテキスト (text) または HTML (html) に指定します。-file オプションを指定しない場合、テキスト フォーマットはデフォルトで Tcl コンソールに表示され、HTML フォーマットは現在の作業ディレクトリの diff.html に記述されます。デフォルト フォーマットは text です。

-file <arg> (オプション): 違いレポートを保存するファイルを指定します。-format オプションで text を指定していない場合、ファイルの拡張子を .htm にすると HTML 出力が生成されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-open_html (オプション): 指定した HTML レポートをデフォルトのウェブ ブラウザーで開きます。このオプションを指定すると、レポートのフォーマットは HTML に指定されるので、-format text オプションと共に使用することはできません。

注記: このオプションでは、Internet Explorer または Edge ウェブ ブラウザーはサポートされません。HTML レポートを生成後にこれらのブラウザーで開くことはできますが、ツールで自動的に開くことはできません。

-brief (オプション): 2 つのデザインが異なるかどうかだけを示すメッセージを表示し、どのように異なるかの詳細は示しません。スクリプト作成のための事前チェックとして使用できます。

-strict (オプション): すべての変更を機能的な変更として解析してレポートし、変更がない場合は 0 を返します。機能的な変更は、オブジェクト、接続、パラメーターなど、デザイン自体に関するものです。機能的でない変更は、ツール バージョンなど BD のメタデータに関するもので、デザインの機能的な違いとは認識されません。デフォルトでは機能的な違いのみがレポートされます。-strict コマンドを指定すると、すべての変更がレポートに含まれます。

-fast (オプション): インメモリ ブロック デザインを開く代わりに、BD JSON ファイルを読み出すことにより比較をすばやく実行します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-depth <arg>` (オプション): デザイン アイテムが等しい場合に、レポートするデザイン階層の深さを指定します。このオプションを指定した場合でも、デザインの違いは常にレポートされます。

`-crossprobe` (オプション): HTML レポートにクロスプローブ リンクを追加します。これにより、レポートで選択したアイテムが開いているブロック デザインで選択されます。このオプションを指定すると HTML フォーマットと `-open_html` が指定されます。`-format text` オプションと共に使用することはできません。

`-repository <arg>` (オプション): ブロック デザインにサイリンクスの標準 IP カタログ以外からの IP が含まれている場合に、IP リポジトリを指定します。IP リポジトリの場所は、デザインで使用されている非標準 IP への参照を解決するために必要です。

`-take_snapshot` (オプション): 現在開いているブロック デザインのスナップショットを作成します。このオプションでは、デザイン名および追加のオプションは必要ありません。

`-diff_snapshot` (オプション): 現在開いているブロック デザインをブロック デザインのインメモリ スナップショットと比較します。スナップショットが作成されてからブロック デザインに加えられた変更を確認できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<design1>` (オプション): 2 番目のブロック デザインと比較するブロック デザインを指定します。ブロック デザインは、`current_bd_design` または `get_bd_designs` Tcl コマンドを使用して指定するか、パスおよびファイル名で指定できます。

注記: パスをファイル名の一部として指定しない場合は、現在のプロジェクトまたは現在の作業ディレクトリからファイルが検索されます。

`<design2>` (オプション): 最初のブロック デザインと比較するブロック デザインを指定します。ブロック デザインは、`current_bd_design` または `get_bd_designs` Tcl コマンドを使用して指定するか、パスおよびファイル名で指定できます。

例

次の例では、ファイル パスで指定したブロック デザインと現在開いているブロック デザインを比較し、HTML レポートを指定した深さで作成して開きます。

```
report_bd_diffs C:/Data/Base_Zynq_MPSoC.bd [current_bd_design] \
-file C:/Data/diffs5.htm -open_html -depth 5
```

次の例では、現在開いているブロック デザインのスナップショットを作成します。

```
report_bd_diffs -take_snapshot
```

関連項目

- [current_bd_design](#)
- [get_bd_designs](#)

report_bps

指定したブレークポイント オブジェクトの詳細を表示します。

構文

```
report_bps [-quiet] [-verbose] [<BreakPointObjs>...]
```

戻り値

ブレークポイント ID、ファイル名、行番号

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<BreakPointObjs>]</code>	レポートするブレークポイント オブジェクトを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

指定のブレークポイント オブジェクトをレポートするか、現在のシミュレーションのブレークポイントをすべてレポートします。このコマンドを使用するには、シミュレーションを開いている必要があります。

ブレークポイントは、ユーザーの指定するソース コード内の停止地点で、デザインをデバッグする際に使用できます。ブレークポイントを含むデザインをシミュレーションすると、各ブレークポイントでシミュレーションが停止し、値および動作を確認できます。

このコマンドを実行すると、指定のブレークポイントのファイル名と行番号が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<BreakPointObjs>`: 現在のシミュレーションの 1 つまたは複数のブレークポイントを指定します。ブレークポイント オブジェクトは、`add_bp` コマンドでシミュレーションにブレークポイントを追加したときに返されます。

例

次の例では、現在のシミュレーションのブレイクポイントをすべてレポートしています。

```
report_bps
```

次の例では、現在のシミュレーションの指定したブレイクポイントをレポートしています。

```
report_bps bp1 bp2 bp5
```

関連項目

- [add_bp](#)
- [remove_bps](#)

report_bus_skew

タイミング パスをレポートします。

構文

```
report_bus_skew [-delay_type <arg>] [-setup] [-hold] [-no_detailed_paths]
  [-max_paths <arg>] [-nworst <arg>] [-unique_pins] [-path_type <arg>]
  [-sort_by_slack] [-input_pins] [-no_header] [-significant_digits <arg>]
  [-file <arg>] [-append] [-return_string] [-warn_on_violation]
  [-rpx <arg>] [-cells <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-delay_type]	パス遅延のタイプを指定します。有効な値は max、min、min_max で、デフォルトは min_max です。
[-setup]	最大遅延終点タイミング パスをレポートします (-delay_type max と同じ)。
[-hold]	最小遅延終点タイミング パスをレポートします (-delay_type min と同じ)。
[-no_detailed_paths]	最上位サマリ表のみをレポートします。
[-max_paths]	バス スキュー制約ごとにレポートするバスの最大数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
[-nworst]	終点までのワーストパスを制約ごとに N 個までリストします。1 以上の値を指定します。デフォルト値は 1 です。
[-unique_pins]	ピンの各セットに対して、バス スキュー制約ごとに最大 1 つのパスを表示します。
[-path_type]	パス レポートのフォーマットを指定します。有効な値は short、full、full_clock、full_clock_expanded で、デフォルトは full_clock_expanded です。
[-sort_by_slack]	サマリおよび制約ごとのセクションをスラック値順に並べ替えます。
[-input_pins]	バスの入力ピンを表示します。
[-no_header]	ヘッダーなしのレポートを生成します。
[-significant_digits]	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 3 です。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-warn_on_violation]	レポートにタイミング違反が含まれる場合にクリティカル警告を表示します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-cells]	report_bus_skew コマンドを指定の階層セルに対して実行します。
[-quiet]	コマンド エラーを表示しません。

名前	説明
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

`set_bus_skew` コマンドで制約されている信号間で算出されたバス スキューをレポートします。

バス スキュー要件は、スロー コーナーとファースト コーナーの両方に適用されます。Vivado ツールにより、バスの信号すべての中で最も早く到着するものと遅く到着するものが判断され、スローおよびファースト プロセス コーナーの両方に対してバス スキューが算出されて、ワースト ケース スキューがレポートされます。バスの各信号は、同じバスからの基準信号に対してレポートされます。基準信号は、その信号のワースト バス スキューとなるのがどの信号かによるので、バスの各信号で異なることがあります。

バス スキュー レポートは、Tcl コンソールまたはコマンド シェルに出力するか、文字列に返すように指定、あるいはファイルに保存できます。

このコマンドを実行すると、バス スキュー レポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-delay_type <arg>` (オプション): バス スキュー レポートを実行する際に解析に使用する遅延のタイプを指定します。有効な値は `min`、`max`、`min_max`、`max_rise`、`max_fall`、`min_rise`、`min_fall` です。`-delay_type` オプションのデフォルト値は `max` です。

`-setup` (オプション): セットアップ違反がないかどうかをチェックします。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): ホールド違反がないかどうかをチェックします。これは `-delay_type min` を指定するのと同じです。



ヒント: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` を指定するのと同じになります。

`-no_detailed_paths` (オプション): デフォルトでは、バス スキュー レポートにレポートされた信号の詳細パス解析が含まれます。このオプションを使用すると、バス スキュー レポートに詳細パスは含まれません。

`-max_paths <arg>` (オプション): 制約ごとにレポートするバスの最大数を指定します。1 以上の値を指定します。デフォルトでは、`report_bus_skew` コマンドでバス スキュー制約ごとにワースト タイミング パスが 1 つレポートされます。

`-nworst <arg>` (オプション): バス スキュー レポートに出力する終点ごとのタイミング パスの数を指定します。レポートには、指定した数のワースト パスが表示されます。1 以上の値を指定します。デフォルト値は 1 です。

`-unique_pins` (オプション): ピンの各セットに対して 1 つのタイミング パスのみを表示します。

`-path_type <arg>` (オプション): バス スキュー レポートに出力するパス データを指定します。デフォルト フォーマットは `full_clock_expanded` です。有効なパス タイプは、次のとおりです。

- `short`: 始点と終点をタイミング値と共に表示します。

- `full`: 始点、通過点、終点を含む完全なタイミング パスを表示します。
- `full_clock`: 完全なタイミング パスに加えて完全なクロック パスを表示します。
- `full_clock_expanded`: `full_clock` で表示されるタイミング パスに加え、マスター クロックと生成クロック間の完全なクロック パスを表示します。これがデフォルト設定です。

`-sort_by_slack` (オプション): レポートのサマリおよび制約ごとのセクションをスラック値順に並べ替えます。

`-input_pins` (オプション): タイミング パス レポートに入力ピンを表示します。`-path_type` を `full`、`full_clock`、および `full_clock_expanded` に設定した場合に使用します。

`-no_header` (オプション): レポートにヘッダーを含めません。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 3 です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-warn_on_violation` (オプション): レポートに違反が含まれる場合に Vivado Design Suite でクリティカル警告が生成されるよう指定します。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-cells <arg>` (オプション): 指定した階層セルのレポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインの各バス スキュー制約に対してワースト信号 32 個のバス スキューをレポートしています。バスのビットごとに 1 つのパスがレポートされ、入力ピンを含む完全なタイミング パスとそのタイミング値が示されます。

```
report_bus_skew -max 32 -nworst 1 -path_type full -input_pins
```

関連項目

- [open_report](#)
- [set_bus_skew](#)

report_carry_chains

キャリー チェーンをレポートします。

構文

```
report_carry_chains [-file <arg>] [-append] [-return_string] [-cell <args>]  
                    [-max_chains <arg>] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	既存のファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-cell]	指定したセルのキャリー チェーンのみをレポートします。
[-max_chains]	レポートするチェーンの最大数を指定します。デフォルトは 1 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在開いているデザインで使用されているキャリー チェーンの詳細をレポートします。レポートには、すべてのキャリー チェーンの平均深さと、各キャリー チェーンの深さがレポートされます。

デフォルトでは最長のキャリー チェーンがレポートされますが、レポートするチェーン数は指定できます。

このコマンドを実行すると、キャリー チェーン レポートが返されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): レポートの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-cell <arg>` (オプション): キャリー チェーンを解析する際に使用するセルを指定します。このオプションを指定すると、指定した階層セルおよびそのセルのサブモジュールのみで解析が実行されます。

`-max_chains <arg>` (オプション): レポートするチェーン数を指定します。デフォルトでは、最長のキャリー チェーンがレポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインのキャリー チェーンを長い方から 10 個レポートしています。

```
report_carry_chains -max_chains 10
```

report_cdc

現在のデザインにおけるクロック乗せ換え (CDC) パスをレポートします。

構文

```
report_cdc [-from <args>] [-to <args>] [-cells <args>] [-details]
           [-summary] [-all_checks_per_endpoint] [-severity <arg>] [-no_header]
           [-show_waiver] [-no_waiver] [-waived] [-file <arg>] [-append]
           [-return_string] [-name <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-from]	開始クロックを指定します。
[-to]	終了クロックを指定します。
[-cells]	report_cdc を指定のセルに対して実行します。
[-details]	安全にタイミング制約を適用することができない CDC タイミングパスの詳細をレポートします。
[-summary]	CDC のクロックごとのサマリをレポートします。
[-all_checks_per_endpoint]	終点ごとにすべてのチェックをレポートします。
[-severity]	指定した重要度 (情報、警告、またはクリティカル) のみをレポートします。
[-no_header]	ヘッダーなしのレポートを生成します。
[-show_waiver]	除外されているパスを表示します。
[-no_waiver]	除外を無視します。
[-waived]	除外されているパスのみを表示します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

現在の合成済みデザインまたはインプリメント済みデザインにおけるクロック乗せ換え (CDC) パスの詳細をレポートします。非同期クロック (共通の周期がないクロック) 間のパス、およびフォルス パス制約または set_max_delay-datapath_only 例外により無視されている同期パスが解析されます。

デフォルトでは、`report_cdc` コマンドでデザインに含まれるすべてのクロック乗せ換えパスがレポートされます。`-from` および `-to` オプションを使用すると、特定のクロックドメインを指定できます。

`report_cdc` コマンドでは、ソース クロックとデスティネーション クロックの両方が定義されているパスのみがレポートされます。`report_cdc` コマンドを実行する前に `check_timing` コマンドを実行し、デザインに制約されていないクロックがないことを確認してください。I/O ポートに入力または出力遅延制約のみが指定されている場合、`report_cdc` コマンドでは I/O パスのみが解析されます。

パス レポートの重要度は、CDC トポロジによって Critical (クリティカル)、Warning (警告)、または Info (情報) になります。不明な同期トポロジは Critical としてレポートされ、確認する必要があります。2 段シンクロナイザーが使用されていて ASYNC_REG プロパティが設定されていない場合は、Warning となります。クロック イネーブル、MUX、および MUX ホールド CDC 構造は、安全であることを確認する必要があるため、重要度は Warning になります。その他の CDC パスの重要度は Info です。

`report_cdc` コマンドを実行すると、次の情報が返されます。

- Severity (重要度)
- Source Clock (ソース クロック)
- Destination Clock (デスティネーション クロック)
- CDC Type (CDC タイプ)
- Exceptions (例外)
- Endpoints (終点数)
- Safe (安全)
- Unknown (不明)
- No ASYNC_REG (ASYNC_REG プロパティなし)



重要: `set_msg_config` コマンドを使用して、`report_cdc` コマンドで返されるメッセージの重要度を設定することはできません。このコマンドは、メッセージ マネージャーを介してメッセージを生成しません。

引数

`-from <args>` (オプション): クロック乗せ換えをレポートするパスのソース クロック ドメインを指定します。クロックは、名前指定するか、または `get_clocks` コマンドを使用してオブジェクトとして指定します。

`-to <args>` (オプション): クロック乗せ換えをレポートするパスのデスティネーション クロック ドメインを指定します。クロックは、名前指定するか、または `get_clocks` コマンドを使用してオブジェクトとして指定します。

`-cells <arg>` (オプション): 指定した階層セルのレポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-details` (オプション): タイミング パスの詳細をレポートします。CDC パスのサマリ表と、各クロックのソース/デスティネーション クロックの詳細および CDC パスが表示されます。

`-summary` (オプション): デザインに対して返されるデフォルトのレポートです。サマリ レポートには、メッセージの重要度、ソース/デスティネーション クロック ペア、安全な CDC、不明または認識されない CDC、パスの終点数を示す表が含まれます。

`-all_checks_per_endpoint` (オプション): CDC パスの終点ごとに最もクリティカルな違反のみをレポートするのではなく、すべての違反をレポートします。このオプションを `-severity` オプションと使用すると、各終点に対して指定した重要度の違反をすべてレポートできます。

`-severity [Critical | Warning | Info]` (オプション): 指定した重要度レベルの CDC パスのみをレポートします。

`-no_header` (オプション): レポートにヘッダーを含めません。このオプションは、`-return_string` オプションを使用して結果を文字列として返す場合に特に便利です。

`-show_waiver` (オプション): `-details` オプションと共に使用し、除外されている CDC パスをレポートに含め、レポートにパスが除外されているかどうかを示す [Waived] 列を追加します。

`-no_waiver` (オプション): `create_waivers` コマンドで定義された除外を無視し、すべての CDC パスをレポートします。

`-waived` (オプション): `create_waiver` コマンドで除外されている CDC パスのみをレポートします。除外の定義ではなく実際の違反が返されます。除外の定義は、`report_waivers` コマンドでレポートできます。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するクロック乗せ換えレポートビューの名前を指定します。指定した名前が開いているレポートビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインのクロック乗せ換えを、除外されているパスも含めて詳細モードでレポートし、結果をファイルに保存しています。

```
report_cdc -details -show_waiver -file C:/Data/cdc_report.txt
```

次の例では、名前指定したクロックからクロックオブジェクトとして指定した別のクロックまでのクロック乗せ換えをレポートしています。

```
report_cdc -from clk_pin_p -to [get_clocks clk_rx_clk_core]
```

関連項目

- [all_clocks](#)
- [get_clocks](#)
- [report_clock_interaction](#)
- [report_clock_networks](#)
- [report_clocks](#)
- [report_timing](#)
- [report_timing_summary](#)
- [set_clock_groups](#)
- [set_false_path](#)

report_clock_interaction

クロック間のタイミング パスとクロックが供給されないレジスタをレポートします。

構文

```
report_clock_interaction [-delay_type <arg>] [-setup] [-hold]
                        [-significant_digits <arg>] [-no_header] [-file <arg>] [-append]
                        [-name <arg>] [-return_string] [-cells <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-delay_type]	パス遅延のタイプを指定します。有効な値は max、min、min_max で、デフォルトは max です。
[-setup]	最大遅延タイミング パスを考慮します (-delay_type max と同じ)。
[-hold]	最小遅延タイミング パスを考慮します (-delay_type min と同じ)。
[-significant_digits]	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 2 です。
[-no_header]	ヘッダーなしのレポートを生成します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-return_string]	レポートを文字列として返します。
[-cells]	report_clock_interaction コマンドを指定のセルに対して実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

クロックの関連性と複数のクロック ドメインを通過する信号をレポートし、メタステーブル状態、データ損失、非干渉性などの問題を特定します。このコマンドを実行するには、合成済みデザインまたはインプリメント済みデザインを開いている必要があります。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-delay_type <arg>` (オプション): クロック関連性レポートを実行する際に解析に使用する遅延のタイプを指定します。有効な値は `min`、`max`、および `min_max` です。`-delay_type` オプションのデフォルト値は `max` です。

`-setup` (オプション): セットアップ違反がないかどうかをチェックします。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): ホールド違反がないかどうかをチェックします。これは `-delay_type min` を指定するのと同じです。

注記: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` を指定するのと同じになります。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 2 です。

`-no_header` (オプション): レポートにヘッダーを含めません。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): ツールの GUI モードで表示する [Clock Interaction] ウィンドウの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-cells <arg>` (オプション): 指定した階層セルのレポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、インターコネクト遅延モデルを設定し、デバイスのスピード グレードを選択して、`report_clock_interaction` を実行しています。

```
set_delay_model -interconnect none
set_speed_grade -3
report_clock_interaction -delay_type min_max \
    -significant_digits 3 -name "results_1"
```

次の例では、クロック関連性レポートを GUI および指定のファイルに出力し、返された文字列を指定の変数に割り当てています。

```
set clk_int [report_clock_interaction -file clk_int.txt -name clk_int1 \
    -return_string]
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [report_clocks](#)
- [set_delay_model](#)
- [set_speed_grade](#)

report_clock_networks

クロック ネットワークをレポートします。

構文

```
report_clock_networks [-file <arg>] [-append] [-name <arg>]
  [-return_string] [-endpoints_only] [-levels <arg>] [-expand_buckets]
  [-suppress_endpoints <arg>] [-clocks <args>]
  [-unconstrained_roots <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-return_string]	レポートを文字列として返します。
[-endpoints_only]	クロック ネットワークの終点のみを出力します。-levels オプションと共に使用することはできません。
[-levels]	クロック ネットワークをインスタンスの n レベル拡張します。有効な値は 0 以上の値です。-endpoints_only オプションと共に使用することはできません。デフォルトは 0 です。
[-expand_buckets]	バケット処理された終点を拡張し、ピンを表示します。デフォルトでは、終点ピンはセル タイプでバケット処理されています。このオプションは、-levels オプションまたは -endpoints_only オプションを使用している場合にのみ有効です。
[-suppress_endpoints]	クロック ピンまたはクロック以外の終点ピンまでのパスをレポートしません。有効な値は clock、nonclock です。
[-clocks]	クロック ネットワークをレポートするクロックを指定します。指定しない場合は、すべてのクロック ネットワークがレポートされます。
[-unconstrained_roots]	クロック ネットワークをレポートする制約されていないルートピン/ポートを指定します。指定しない場合は、すべての制約されていないクロック ルートがレポートされます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

開いている合成済みまたはインプリメント済みデザインに含まれる各クロック ネットのネットワーク ファンアウトをレポートします。-name オプションを指定すると、クロック ネットワークが階層ツリーでグラフィカルに表示されます。

デフォルトのレポートは、クロックの始点であるクロック ネットの名前とインスタンス ピンを指定します。

レポートは、`-file`、`-return_string`、または `-name` オプションを指定した場合を除き、標準出力に返されます。

引数

`-file <arg>` (オプション): クロック ネットワーク レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): GUI に出力する結果の名前を指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-endpoints_only` (オプション): クロック ネットワークの終点をレポートします。セル タイプでパケット処理され、クロック ピンおよびクロック以外のピン順にリストされます。このオプションを `-levels` オプションと共に使用することはできません。

`-levels <arg>` (オプション): クロック ネットワーク レポートにインスタンス階層の指定のレベルまでを含めます。デフォルト値は 0 です。0 より大きい値を指定できます。十分なレベル数が指定されていれば、クロック パスからパスの終点までがレポートされます。パスの終点は、セル タイプでパケット処理され、クロック ピンおよびクロック以外のピン順にリストされます。このオプションを `-endpoints_only` オプションと共に使用することはできません。

`-expand_buckets` (オプション): `-endpoints_only` または `-levels` オプションで返された終点バケットをすべて拡張し、すべてのクロックの終点をレポートします。

`-suppress_endpoints [clock | nonclock]` (オプション): クロック ルートからクロック ピンまたはクロック以外の終点ピンまでのパスをレポートしません。クロック ネットワーク レポートからクロック ピンまたはクロック以外のピンを除外します。

`-clocks <args>` (オプション): クロック ネットワーク レポートに含めるクロック オブジェクトを指定します。指定しない場合、レポートにすべてのクロックが含まれます。

`-unconstrained_roots <args>` (オプション): UltraScale デバイスおよびクロック ルートを含むデバイス アーキテクチャで、クロック ネットワーク レポートに含める制約されていないクロック ルート ピンまたはポートを指定します。このオプションを指定しない場合、すべての制約されていないクロック ルートがレポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、クロック ネットワーク名と始点を指定したファイルにレポートしています。

```
report_clock_networks -file C:/Data/ClkNets.txt
```

次の例では、指定のクロックの終点をレポートしています。

```
report_clock_networks -endpoints_only -clocks wbClk
```

関連項目

- [create_clock](#)
- [get_clocks](#)

report_clock_utilization

デザインのクロック ネットに関する情報をレポートします。

構文

```
report_clock_utilization [-file <arg>] [-append] [-write_xdc <arg>]
    [-cells <args>] [-clock_roots_only] [-return_string] [-name <arg>]
    [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	既存のファイルの最後に追加します。
[-write_xdc]	クロック制約を出力するファイルの名前を指定します。ファイル名は指定する必要があります。
[-cells]	クロック使用率をレポートするセル/BEL インスタンスを指定します。
[-clock_roots_only]	クロック ルート割り当てのみをレポートします。
[-return_string]	レポートを文字列として返します。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

デザインのクロック ネットに関する情報と、ターゲット デバイスでのクロック リソースの使用率を返します。

生成クロック使用率レポートから、現在配置済みのクロック リソースの配置制約を生成できます。-write_xdc コマンドを使用すると、これらの制約を使用してデザインの今後の実行でクロック リソースの配置を保持できます。



重要: UltraScale デバイスでは、現在のクロック配置を再現することが目的の場合は、記述された XDC ファイルからの BUFGCE LOC プロパティを使用します。制約をクロック アーキテクチャの始点とし、Vivado Design Suite でクロック リソースがある程度柔軟に配置されるようにするには、BUFGCE LOC プロパティではなく同等の CLOCK_REGION プロパティを使用します。

デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-write_xdc <filename>` (オプション): さまざまなクロック リソースの XDC ロケーション制約を出力するファイルの名前を指定します。パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-cells <args>` (オプション): デザイン全体ではなく、指定したセルに基づく詳細をレポートします。

`-clock_roots_only` (オプション): UltraScale デバイスおよびクロック ルートを含むデバイス アーキテクチャで、各クロック ネットのクロック ルート割り当てのみをレポートします。

注記: このオプションは、7 シリーズ デバイスでは使用できません。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するクロック使用率レポート ビューの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインのクロック ネットに関する情報とターゲット デバイスでのクロック リソースの使用率が返され、指定のファイルに保存されます。

```
report_clock_utilization -file C:/Data/FPGA_Design/clock_util.txt
```

次の例では、クロック ネットとクロック リソースの使用率を標準出力に表示し、XDC ロケーション制約は指定のファイルに記述しています。

```
report_clock_utilization -write_xdc clock_util_xdc.txt
```

注記: XDC ファイル名の一部としてパスが指定されていないので、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

関連項目

- [create_clock](#)
- [create_generated_clock](#)

report_clocks

クロックをレポートします。

構文

```
report_clocks [-file <arg>] [-append] [-return_string] [-quiet] [-verbose]
[<clocks>]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<clocks>]	クロックを指定します。デフォルトは * です。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

デザインに含まれるすべてのクロックを含む表を返します。返されるクロックには、現在の合成済みデザインまたはインプリメント済みデザインに含まれる伝搬されたクロック、生成クロック、自動生成されたクロック、仮想クロック、および反転クロックが含まれます。各クロック ネットに関するより詳細な情報は、`report_clock_utilization` コマンドを使用して取得できます。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<clocks>` (オプション): 指定した検索パターンと一致するクロックを検索します。デフォルトの検索パターンはワイルドカード (*) で、デザインのすべてのクロックが返されます。複数の検索パターンを指定して、異なる検索条件に基づいてクロックを検索できます。

例

次の例では、現在のデザインに含まれるクロックの名前、周期、波形、およびソースが返されます。

```
report_clocks -file C:/Data/FPGA_Design/clock_out.txt
```

次の例では、デザインに含まれるクロックで、名前に「Clock」が含まれるものが返されます。

```
report_clocks *Clock*
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [report_clock_utilization](#)

report_compile_order

ファイルを解析し、階層を構築して、コンパイル順をレポートします。

構文

```
report_compile_order [-fileset <arg>] [-missing_instances] [-constraints]
                    [-sources] [-used_in <arg>] [-file <arg>] [-append]
                    [-of_objects <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-fileset]	コンパイル順をレポートするファイルセットを指定します。
[-missing_instances]	デザイン階層で不足しているインスタンスをレポートします。
[-constraints]	制約のコンパイル順をレポートします。
[-sources]	ソースのコンパイル順をレポートします。
[-used_in]	指定の段階で使用されるファイルのコンパイル順をレポートします。
[-file]	結果を保存するファイルの名前を指定します。
[-append]	結果を既存のファイルに追加します。
[-of_objects]	指定したファイル、ファイルセット、IP、リコンフィギャラブル モジュール (reconfig_module) のファイル オブジェクトを取得します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

さまざまなアクティブ ファイルセット (制約、デザイン ソース、およびシミュレーション ソース) のファイルのコンパイル順をレポートします。

合成、インプリメンテーション、シミュレーションのファイル処理順が返されます。-fileset、-constraints、-sources オプションを使用して特定のファイルセットを指定することにより、レポートを制限できます。

-used_in オプションを使用すると、USED_IN プロパティに基づいて、合成、シミュレーション、またはインプリメンテーションのいずれかの段階で使用されるファイルの処理順をレポートできます。

デフォルトではレポートは Tcl コンソールまたは標準出力に表示されますが、ファイルに記述することもできます。

引数

`-of_objects <args>` (オプション): 指定したファイルセット、サブデザイン、またはリコンフィギュレーション モジュールに関連付けられているファイルをレポートします。デフォルトでは、すべてのファイルセットが検索されます。`-compile_order` および `-used_in` オプションを指定した場合、`-of_objects` オプションで指定できるのは1つのファイルセット、または IP コア、ブロック デザイン、DSP デザインなどの1つのサブデザインのみです。サブデザインは、複合ファイルとも呼ばれます。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-fileset <arg>` (オプション): レポートするファイルセットを指定します。

`-missing_instances` (オプション): 現在のファイルセットまたは指定のファイルセットで、ソース ファイルのないセルのリストを返します。

`-constraints` (オプション): 現在のデザインの制約ファイルのコンパイル順を、デザインの IP の制約も含めてレポートします。

`-sources` (オプション): デザイン ソースの `sources_1` ファイルセットに含まれるファイルのコンパイル順をレポートします。

`-used_in <arg>` (オプション): ファイルの `USED_IN` プロパティの値の1つを指定し、その値に一致するファイルを返します。指定可能な値は、`synthesis`、`simulation`、`implementation` などです。`USED_IN` プロパティおよびサポートされる値は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) を参照してください。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインのアクティブ ファイルセットのコンパイル順をレポートしています。

```
report_compile_order
```

次の例では、現在のデザインでソース ファイルのないセルのリストを返し、指定のファイルに保存しています。

```
report_compile_order -missing_instances -file C:/Data/report1.txt -append
```

次の例では、アクティブ制約セットのファイルのコンパイル順をリストしています。

```
report_compile_order -constraints
```

関連項目

- [current_fileset](#)
- [get_files](#)
- [update_compile_order](#)

report_conditions

指定した条件オブジェクトの詳細を表示します。

構文

```
report_conditions [-quiet] [-verbose] [<ConditionObjs>...]
```

戻り値

各条件オブジェクトの名前、ID、条件式、およびコマンド

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<ConditionObjs>]	条件オブジェクトを ID または名前で指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

指定のシミュレーション条件をレポートするか、現在のシミュレーションの条件をすべてレポートします。このコマンドを使用するには、シミュレーションを開いている必要があります。

条件はシミュレーションの開始前に定義できます。条件を追加すると、シミュレータで信号の変化が検出されるたびに条件式が評価されます。指定の条件が真になると、条件コマンドが実行されます。

このコマンドを実行すると、条件の条件識別子、式、コマンド、および名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<ConditionObjs> (オプション): レポートする現在のシミュレーションの 1 つまたは複数の条件を指定します。条件識別子は、add_condition コマンドで条件を定義したときに返されます。デフォルトでは、すべての条件がレポートされます。

例

次の例では、現在のシミュレーションの条件をすべてレポートしています。条件識別子、式、コマンド、および名前がレポートされます。

```
report_conditions
#2: condition2
    Expression: {/testbench/reset == 0 }
    Command:    {
puts "Condition Reset was encountered at [current_time]. \
    Stopping simulation."
stop }
    Name:       resetLow
#3: condition3
    Expression: {/testbench/leds_n == X000 }
    Command:    {
puts "Condition LED Unknown was encountered at [current_time]. \
    Stopping simulation."
stop }
    Name:       ledUnknown
```

関連項目

- [add_condition](#)
- [remove_conditions](#)

report_config_implementation

ユーザー設定可能なイプリメンテーション パラメーターをレポートします。

構文

```
report_config_implementation [-file <arg>] [-force] [-append]
                             [-return_string] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-force]	既存のファイルを上書きします。
[-append]	既存のファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

インプリメンテーション プロセス用にユーザーが設定可能なイプリメンテーション フローの設定パラメーターをレポートします。これらのパラメーターは、`config_implementation` コマンドを使用して変更できます。

このコマンドを実行すると、選択したレポートまたは出力ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-file <arg>` (オプション): パラメーター レポートを指定したファイルに保存します。指定したファイルが既に存在する場合、`-force` オプションが指定されていなければ上書きされません。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-force` (オプション): 指定のファイルが存在する場合に上書きします。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

この例では、ユーザー設定可能なイプリメンテーション パラメーターを Tcl コンソールに表示しています。

```
report_config_implementation
```

関連項目

- [config_implementation](#)

report_config_timing

タイミング解析に影響する設定をレポートします。

構文

```
report_config_timing [-file <arg>] [-append] [-name <arg>] [-return_string]
                    [-all] [-no_header] [-rpx <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	結果をファイルに出力します。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-return_string]	レポートを文字列として返します。
[-all]	すべての設定をレポートします。デフォルトでは、一般的に重要な設定のみがレポートされます。
[-no_header]	ヘッダーなしのレポートを生成します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

現在のデザインのタイミング制約の設定をレポートします。

デフォルトでは、主要なタイミング制約のみを含む簡略されたレポートが示されます。タイミングに関するすべての設定を返すには、-all オプションを使用します。

引数

-file <arg> (オプション): タイミング制約設定レポートを保存するファイルを指定します。-append オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: -append オプションは、-file オプションを使用している場合にのみ使用可能です。

-name <arg> (オプション): GUI に出力する結果の名前を指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-all` (オプション): デザインの属性および制約に関連するすべてのタイミングのステートをレポートします。デフォルトでは、主要なタイミング属性に関連するもののみがレポートされます。

`-no_header` (オプション): レポートにヘッダーを含めません。デフォルトでは、レポートに次の情報を含むヘッダーが含まれます。

- レポート タイプ: `timer_configuration`。
- デザイン: デザインの最上位モジュール。
- パーツ: ターゲット パーツのデバイス、パッケージ、およびスピード グレード。
- バージョン: レポートの作成に使用されたツールのバージョン。
- 日付: レポートの日付。
- コマンド: レポートの作成に使用されたコマンド オプション。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のタイミング設定をレポートし、情報を文字列として返し、その文字列を指定の Tcl 変数に設定しています。

```
set timeConfig [report_config_timing -all -no_header -return_string]
puts $timeConfig
```

関連項目

- [delete_timing_results](#)

report_control_sets

デザイン特有の制御セットについてレポートします。

構文

```
report_control_sets [-file <arg>] [-append] [-hierarchical]
                   [-hierarchical_depth <arg>] [-sort_by <args>] [-cells <args>]
                   [-return_string] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	既存のファイルの最後に追加します。
[-hierarchical]	テキスト形式の階層レポートを生成します。
[-hierarchical_depth]	テキスト形式の階層レポートのレベルを指定します。デフォルトは 0 です。
[-sort_by]	並べ替え条件を指定します。-verbose オプションを使用している場合にのみ使用できます。有効な値は clk、clkEn、set です。例: report_control_sets -verbose -sort_by {clk clkEn}。
[-cells]	制御セットをレポートするセル/BEL インスタンスを指定します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在のデザインの制御セットをレポートします。

制御セットとは、スライス レジスタおよび LUT の制御信号 (クロック、CE、SR) をまとめたものです。レジスタを同じデバイス リソースにパックするには、同じ制御セットに含める必要があります。制御信号を持たないレジスタを制御信号を持つレジスタと共にデバイスのリソースにパックすることはできません。制御セット数が多いと、デバイスにフィットさせるのが困難になり、配線の密集やタイミング問題を引き起こす可能性があります。

デフォルトでは、report_control_sets コマンドで制御セット数のみを示すレポートが返されます。-verbose オプションを使用すると、デザイン全体または指定のセルの制御セットすべての詳細が返されます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-hierarchical` (オプション): デザインの制御セットの階層レポートを生成します。

`-hierarchical_depth <args>` (オプション): `-hierarchical` オプションを指定した場合にレポートする階層のレベル数を指定します。デフォルトは0で、デザイン階層全体がレポートされます。

`-sort_by <args>` (オプション): `-verbose` オプションで生成された詳細レポートを指定の条件で並べ替えます。有効な並べ替え条件は、`clk`、`clkEn`、および `set` です。

注記: `-sort_by` オプションは、`-verbose` オプションと共に使用します。

`-cells <args>` (オプション): 指定したセル オブジェクトの制御セットをレポートします。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの制御セットを、`clk` および `clkEn` 信号で並べ替えてレポートします。

```
report_control_sets -verbose -sort_by {clk clkEn}
```

次の例では、指定したセルの制御セットを `clk` および `set` で並べ替えてレポートします。

```
report_control_sets -verbose -sort_by {clk set} -cells [get_cells usb*]
```

report_datasheet

データシートをレポートします。

構文

```
report_datasheet [-significant_digits <arg>] [-file <arg>] [-append]
                 [-return_string] [-sort_by <arg>] [-name <arg>] [-show_all_corners]
                 [-show_oe_timing] [-group <args>] [-rpx <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-significant_digits]	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 3 です。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-sort_by]	並べ替え順を指定します。有効な値は clock、port で、デフォルトは clock です。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-show_all_corners]	すべてのコーナーを表示します。
[-show_oe_timing]	出力イネーブル (トライステート) のタイミングを表示します。
[-group]	スキューを算出する出力ポートのグループを定義します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

現在のデザインのデータシート レポートを生成します。クロックに対する入力 I/O のセットアップおよびホールド タイム、クロックから出力パッドへの最大/最小遅延、入力/出力バスのスキューをレポートします。

データシート レポートには、パッケージ トレース フライト タイムを含む、デザインのパッケージ ボール/パッドにおけるタイミング特性が含まれます。フライト タイムをディスエーブルにするには、次のコマンドを使用します。

```
config_timing_analysis -disable_flight_delays true
```

ソース同期出力スキューは、`report_datasheet` コマンドの `-group` オプションを使用してソース クロック出力ポートを含むデータ バスのすべてのポートをグループ化すると、自動的に算出できます。ソース クロック出力ポートをグループ リストの最初にリストする必要があります。次に例を示します。

```
report_datasheet -file output_filename -group [get_ports \
{clock_port data_bit[0] data_bit[1] data_bit[2]}]
```

引数

`-significant_digits <arg>` (オプション): 表示する桁数を指定します。有効な値は 0 ～ 3 で、デフォルトは 3 です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-sort_by [port | clock]` (オプション): 並べ替え順を指定します。有効な値は `clock` または `port` です。デフォルトは `clock` で、レポートがクロック順に並べられます。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するデータシート レポートの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-show_all_corners` (オプション): すべてのプロセス コーナーをレポートします。

`-show_oe_timing` (オプション): 出力イネーブルを使用するパスの最大値および最小値をレポートし、ON および OFF の両方の状態を示します。

`-group [get_ports {xxx1 xxx2 ... xxxN}]` (オプション): 解析するポートのグループを定義します。ポートは、`get_ports` コマンドを使用してポート オブジェクトとして指定する必要があります。最初のポートが、スキュー算出の基準として使用されます。ほとんどの場合、これはソース同期出力インターフェイスのクロック ポートです。それぞれ基準クロック ポートを持つグループを複数指定できます。 `-group` オプションを指定しない場合は、ソース クロックに基づいて出力ポートのグループが自動的に検出され、そのクロックに基づくスキューがレポートされます。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `. rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ポート順に並べたすべてのプロセス コーナーのデータシートが返されます。

```
report_datasheet -sort_by port -show_all_corners
```

次の例では、2 つのポート グループに対してスキュー算出のデータシートをレポートしています。各グループの最初のポートが、そのグループのスキュー算出の基準として使用されます。この例では、`CLK0OUT` が `DATA0` ~ `4` のクロック、`CLK1OUT` が `DATA4` ~ `7` のクロックです。

```
report_datasheet -file ds.txt -group [get_ports \
    {CLK0OUT DATA0 DATA1 DATA2 DATA3}] \
    -group [get_ports {CLK1OUT DATA4 DATA5 DATA6 DATA7}]
```

関連項目

- [get_ports](#)

report_debug_core

デバッグ コアの詳細をレポートします。

構文

```
report_debug_core [-file <arg>] [-append] [-return_string] [-full_path]
                  [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-append]</code>	結果をファイルの最後に追加します。上書きはしません。
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-full_path]</code>	レポートに完全な階層ネットパスを表示します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Debug \(デバッグ\)](#)

説明

現在のプロジェクトのさまざまな Vivado デバイス ツールのデバッグ コアおよびこれらのコアのパラメーターをレポートします。デバッグ コアをプロジェクトに追加するには、`create_debug_core` コマンドを使用します。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): レポートを文字列として返します。このオプションを `-file` オプションと共に使用することはできません。

`-full_path` (オプション): ネット名をネットの完全な階層パスを使用して表示します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デバッグ コアのレポートが指定したディレクトリの指定したファイルに書き込まれます。

```
report_debug_core -file C:/Data/FPGA_Design/project_1_cores.txt
```

関連項目

- [create_debug_core](#)

report_design_analysis

デザイン解析レポート

構文

```
report_design_analysis [-file <arg>] [-append] [-return_string]
  [-complexity] [-cells <args>] [-bounding_boxes <args>]
  [-hierarchical_depth <arg>] [-congestion] [-min_congestion_level <arg>]
  [-timing] [-setup] [-hold] [-show_all] [-full_logical_pin]
  [-routed_vs_estimated] [-logic_level_distribution]
  [-logic_level_dist_paths <arg>] [-min_level <arg>] [-max_level <arg>]
  [-return_timing_paths] [-of_timing_paths <args>] [-max_paths <arg>]
  [-extend] [-routes] [-end_point_clock <arg>] [-logic_levels <arg>]
  [-qor_summary] [-name <arg>] [-no_pr_attribute] [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-complexity]	デザインのインターコネクトの複雑性 (Rent) をレポートします。
[-cells]	指定したセルの解析情報をレポートします。
[-bounding_boxes]	指定した境界ボックスのインターコネクトの複雑性 (Rent) をレポートします。
[-hierarchical_depth]	-complexity オプションを指定した場合の階層レベル数を指定します。デフォルトは 1 です。
[-congestion]	デザインの密集度をレポートします。
[-min_congestion_level]	配線の密集度をレポートする最小密集レベルを 3 ~ 8 の整数値で指定します。デフォルトは 5 です。
[-timing]	クリティカル パスの特性をレポートします。
[-setup]	クリティカル セットアップ パスの特性をレポートします。
[-hold]	クリティカル ホールド パスの特性をレポートします。
[-show_all]	タイミング特性レポートにさらに特性を追加します。
[-full_logical_pin]	レポートに階層ピン名を表示します。
[-routed_vs_estimated]	クリティカル パスの関連の特性を見積もりモードでレポートします。
[-logic_level_distribution]	ロジック レベルの分配をレポートします。
[-logic_level_dist_paths]	-logic_level_distribution オプションを使用してロジック レベルの分布を解析する際のクリティカル パスの数を指定します。デフォルトは 1000 です。
[-min_level]	ロジック レベル数が <min_level> -1 以下のすべてのパスを 1 つのピンにグループ化します。1 以上の値を指定します。デフォルトでは使用されません。

名前	説明
<code>[-max_level]</code>	ロジック レベル数が <code><max_level> + 1</code> 以上のすべてのパスを 1 つのピンにグループ化します。 <code><max_level></code> は、0 または <code>-min_level</code> を使用している場合は <code><min_level></code> より大きい値にする必要があります。
<code>[-return_timing_paths]</code>	タイミング パス オブジェクトを返します。
<code>[-of_timing_paths]</code>	指定したパスの特性をレポートします。
<code>[-max_paths]</code>	-timing オプションを使用する際に考慮するパスの数を指定します。デフォルトは 1 です。
<code>[-extend]</code>	クリティカル パスの始点の前のワースト パスおよびクリティカル パスの終点の後のワースト パスの特定をレポートします。
<code>[-routes]</code>	分布をロジック レベル数ではなく配線数でレポートします。
<code>[-end_point_clock]</code>	このオプションで指定した終点クロック名のタイミング パス オブジェクトを返します。
<code>[-logic_levels]</code>	このオプションで指定したピン名のバケットに含まれるタイミング パス オブジェクトを返します。
<code>[-qor_summary]</code>	デザイン フロー サマリをレポートします。
<code>[-name]</code>	結果を出力する GUI パネルの名前を指定します。
<code>[-no_pr_attribute]</code>	PR 属性なしのレポートを生成します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

クリティカル パスのタイミング特性およびデザインの複雑性に関するデータをレポートします。この情報は、タイミング クロージャ問題および配線の密集の問題がある部分を特定して解析するのに役立ちます。このコマンドの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン解析およびクロージャ テクニック』(UG906) を参照してください。

`report_design_analysis` コマンドには、次の 3 つの操作モードがあります。

- タイミング: タイミング パスのタイミングおよび物理的な特性をレポートします。
- 複雑性: デザインの配線の複雑性および LUT の分布を解析します。
- 密集度: デザインの配線の密集度を解析します。

タイミング モードでは、スタティック タイミング エンジン呼び出し、クリティカル パス データを解析して各パスの特性をレポートします。パスの特性には、クロック スキューなどの重要な要素、クロック乗せ換え領域などの配置に関する障害、Pblock や LOC などの物理制約が含まれます。クリティカル パスの数を指定してパスのリストを拡張したり、タイミング パス オブジェクトを指定して特定のパスを解析できます。クリティカル パスの前後のパスを含めるオプションもあります。

タイミング モードでレポートされるパスの特性の定義は、次のとおりです。

- `PATH_TYPE`: セットアップ (SETUP) またはホールド (HOLD)。
- `REQUIREMENT`: スタティック タイミング解析からの遅延要件。

- PATH DELAY: スタティック タイミング解析からのデータパス遅延。
- LOGIC DELAY: パスの PATH DELAY のロジックによる遅延部分。
- NET DELAY: パスの PATH DELAY のワイヤによる遅延部分。ネット遅延は、`set_delay_model` コマンドで指定された見積もり配線遅延または実際の配線遅延に基づきます。
- CLOCK SKEW: ソース クロックとデスティネーション クロックの遅延差。
- SLACK: スタティック タイミング解析からのパス タイミング スラック。
- CLOCK RELATIONSHIP: SAME_CLOCK または RELATED_CLOCK。不足している可能性のあるクロック間制約を特定するのに役立ちます。
- TIMING EXCEPTION: パスに割り当てられている `set_false_path` や `set_multicycle_path` などのタイミング例外。
- LOGIC LEVELS: ソースとデスティネーションの間のロジック レベル数。-`logic_level_distribution` オプションを指定した場合にレポートされます。
- LOGICAL PATH: 始点および終点を含むパスのセルを順番に示したリスト。

注記: パーシャル リコンフィギュレーション (PR) デザインでは、セルがリコンフィギュラブル パーティション (:RP#) に含まれるか、デザインのスタティック領域 (:S) に含まれるかを示す文字列が論理パスに追加されます。レポートの下部にある変換テーブルに、:RP# の特定のリコンフィギュラブル パーティションへのマップが示されます。

- START POINT CLOCK: パスの始点のクロック ドメイン。
- END POINT CLOCK: パスの終点のクロック ドメイン。
- START POINT PIN PRIMITIVE: パスの始点のライブラリ セルおよびピン。
- END POINT PIN PRIMITIVE: パスの終点のライブラリ セルおよびピン。
- START POINT PIN: パスの始点のインスタンスおよびピン名。
- END POINT PIN: パスの終点のインスタンスおよびピン名。
- COMB DSP: パスにある組み合わせ DSP ブロックの数。
- DOA REG: パス上の DOA レジスタの数。
- DOB REG: パス上の DOB レジスタの数。
- MREG: パス上の MREG レジスタの数。
- PREG: パス上の PREG レジスタの数。
- BRAM CROSSINGS: パスが通過するブロック RAM 列の数。
- DSP CROSSINGS: パスが通過する DSP ブロック列の数。
- IO CROSSINGS: パスが通過する I/O 列の数。
- CONFIG CROSSINGS: パスが通過する CONFIG の数。
- SLR CROSSINGS: パスが通過する SLR の数。
- BOUNDING BOX SIZE: デバイスの RPM_X (水平方向) および RPM_Y (垂直方向) サイト座標に基づく RPM GRID ユニットで表したクリティカル パスを含む矩形。サイト (スライス、DSP、ブロック RAM など) によってサイズが異なるので、各サイトにデバイス内の位置を示す固有の RPM_X および RPM_Y プロパティがあります。

- CLOCK REGION DISTANCE: パスの始点から終点までで、水平方向および垂直方向に通過したクロック領域の数 (値のペア)。通過するクロック領域の数を最小限に抑えると、クリティカル パス遅延およびクロック スキューを向上できます。
 - 例 1: クロック領域 X1Y1 で開始し、クロック領域 X3Y3 で終了するクリティカル パスの場合、CLOCK_REGION_DISTANCE は (2, 2) になります。
 - 例 2: クロック領域 X2Y1 で開始し、クロック領域 X0Y0 で終了するクリティカル パスの場合、CLOCK_REGION_DISTANCE は (-2, -1) になります。
- PBLOCKS: パスが通過する Pblock の数。
- HIGH FANOUT: パスに含まれるネットの最大ファンアウト。
- CUMULATIVE FANOUT: パスの合計ファンアウト。
- DONT TOUCH: パスに含まれる DONT_TOUCH の値が TRUE であるセルの数。セルの DONT_TOUCH が TRUE の場合、最適化で削除されることはなくなり、phys_opt_design の複製などの有益な最適化がディスエーブルになります。
- MARK DEBUG: パスに含まれる MARK_DEBUG の値が TRUE であるセルの数。デフォルトでは、MARK_DEBUG が設定されているネットでは、DONT_TOUCH が TRUE に設定され、そのネットでの最適化がディスエーブルになります。DONT_TOUCH を FALSE に設定して最適化をイネーブルにすると、タイミングが向上する可能性があります。
- FIXED LOC: パスに含まれる IS_LOC_FIXED の値が TRUE である配置済みセルの数。FIXED セルは、place_design または phys_opt_design で移動されることはありません。
- FIXED ROUTE: パスに含まれる IS_ROUTE_FIXED の値が TRUE である配線済みセルの数。FIXED 配線は、route_design で解除されたり配線し直されたりすることはありません。
- HOLD FIX DETOUR: ホールド タイミングを修正するために配線後のクリティカル パスに適用された配線迂回の数。
- COMBINED LUT PAIRS: パスに含まれる LUT セルで、O6 および O5 出力の両方を使用するためにほかの LUT セルと共に同じ LUT BEL に組み合わせられた LUT の数。LUTNM、HLUTNM、または SOFT_HLUTNM と組み合わせられた LUT セルは、HLUTNM プロパティを空の文字列に設定することにより、組み合わせを解除して配置し直すことができます。これにより、LUT の組み合わせおよび組み合わせ解除のタイミングおよび密集への影響を調べることができます。
- パーシャル リコンフィギュレーション (PR) デザインでは、次のフィールドがレポートされます。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。
 - PR PATH TYPE: パスが完全にスタティック領域に含まれるか、完全にリコンフィギュラブル パーティション (RP) に含まれるか、または領域間の境界をまたいでいるかを示します。タイミング パスの遅延エレメントも領域間で分解されます。
 - STATIC CROSSINGS: リコンフィギュラブル パーティション (RP) のパスがスタティック領域に入る回数をレポートします。
 - RP CROSSINGS: スタティック領域のパスがリコンフィギュラブル パーティション (RP) 領域に入る回数をレポートします。
 - BOUNDARY FANOUT: PPLOC での境界パスのダウンストリーム ロードへのファンアウトをレポートします。

複雑性モードでは、現在のデザインに対して複雑性解析を実行し、複雑性を表す Rent 指数、平均ファンアウト、およびプリミティブ ヒストグラムをレポートします。解析は、最上位デザインに対して実行するか、デザインの各階層レベル (レベルは指定可能) で実行できます。

次に、複雑性モードでレポートされる特性の定義を示します。

- **Rent:** Rent の法則で定義された Rent 指数は、ネットリストのインターコネクトの複雑性を示す評価基準です。Rent 値が高い方が複雑であり、配線の密集を回避するのが困難になります。ほとんどのデザインは、Rent 値は 0.5 ~ 0.6 の範囲です。Rent 値 0.65 は高いと考慮され、0.85 は非常に高いと考慮されます。
- **平均ファンアウト:** グローバル バッファを除くデザインに含まれるロジック セルの平均ファンアウト。平均ファンアウトが大きいと、配置および配線が困難になる可能性があります。絶対値から困難さを予測することはできませんが、デザイン間または階層レベル間で値を比較すると有益な場合があります。
- **プリミティブ ヒストグラム:** デザインで使用される特定のプリミティブ タイプの合計を示します。Rent 値が高い原因が LUT6 セルが多数使用されていることである場合があります。ほかのサイズの LUT よりも多数の LUT6 がある場合、エリアを優先した合成ストラテジを使用することにより Rent 値を下げるができることがあります。



ヒント: 複雑性の特性で常に配線の密集を予測できるわけではありませんが、密集の問題が発生した場合に問題のエリアを特定するのに使用できます。

密集モードでは、デザインが解析され、配線の密集を緩和するのに役立つメトリクスが示されます。

`report_design_analysis` コマンドの結果を使用して、特定の配線ホット スポットを回避するよう配置を変更できます。

このコマンドを実行すると、作成されたファイルが返されるか、Tcl コンソールに解析結果が表示されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-file <arg>` (オプション): 解析結果を保存するファイルを指定します。`-append` オプションが指定されていないければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-complexity` (オプション): デザインの複雑性解析を実行し、Rent 指数、平均ファンアウト、およびプリミティブ ヒストグラムをレポートします。



ヒント: `-complexity` オプションは、`-cells` および `-hierarchical_depth` と共に指定して、デザインまたはセルの解析を制御できます。

`-cells <arg>` (オプション): デザイン解析をレポートする際に使用するセルを指定します。セルはデザインの階層モジュールである必要があります。このオプションは、最上位セル以外のセルを解析する場合に使用します。セルは、名前で指定するか、または `get_cells` コマンドを使用してオブジェクトとして指定します。`-hierarchical_depth` オプションと共に使用することはできません。

`-bounding_boxes <args>` (オプション): 複雑性を解析するデバイスの領域を指定します。これによりエリアが制限され、解析にかかる時間が短縮されます。この引数は、境界ボックスの左下角と右上角にあるデバイス タイルのペアで指定します。次のように複数の領域を指定できます。

```
-bounding_boxes { "CLE_M_X21Y239:CLEL_R_X28Y254" \
                  "CLEL_R_X18Y171:CLE_M_X26Y186" }
```



重要: この例に示すように、開始波かっこ ({}) と境界ボックス指定の間にスペースが必要です。

`-hierarchical_depth <arg>` (オプション): 複雑性解析を、デザインの最上位または指定のセルではなく、デザインの階層に対して実行します。この場合、デザインまたはセルの階層から指定したレベル数解析されます。デフォルト値は 1 です。`-complexity` オプションと共に使用する必要があります。

`-congestion` (オプション): 配置済みデザインの垂直方向および水平方向の配線の密集度をタイルごとにレポートします。このオプションは、インプリメンテーション後または `place_design` コマンドの実行後にのみ使用可能です。複雑性解析に `-cells` オプションが指定されていても、密集解析はデザイン全体に対して実行されます。

`-min_congestion_level <arg>` (オプション): デザインの密集を表示する最小密集レベルを指定します。サポートされる最小値は 3 で、3 未満の密集ウィンドウに関する情報はありません。このオプションを指定しない場合のデフォルトは 5 です。

`-timing` (オプション): デザイン解析をタイミング モードで実行し、クリティカル パスのセットアップおよびホールド特性をレポートします。ほかのオプションを指定しない場合、これが `report_design_analysis` コマンドのデフォルトです。



ヒント: `-timing` オプションは、`-setup`、`-hold`、`-of_timing_paths`、`-max_paths`、および `-extend` オプションと共に使用できますが、`-cells` または `-hierarchical_depth` オプションと共に使用することはできません。

`-setup` (オプション): セットアップ パスの特性のみをレポートします。`-timing` オプションと共に使用する必要があります。

`-hold` (オプション): ホールド パスの特性のみをレポートします。`-timing` オプションと共に使用する必要があります。



重要: `-hold` オプションを `-extend` オプションと共に使用すると、左側にワースト セットアップ パスを示すセットアップ レポート、右側にそれらのパスの対応するホールド レポートが生成されます。同じ始点と終点のホールド特性とセットアップ特性が並べて表示されるので、ホールドおよびセットアップの修正がお互いに影響しているかどうかを判断するのに役立ちます。

`-show_all` (オプション): タイミング特性レポートにさらに特性を追加します。

`-full_logical_pin` (オプション): デザイン解析レポートに完全な階層ピン名を出力します。

`-routed_vs_estimated` (オプション): 同じパスの見積もり配線遅延と実際の配線遅延を並べて表示します。



重要: 見積もり遅延と実際の遅延の両方を表示するには、`set_delay_model -interconnect` を `actual` に設定する必要があります。

`-logic_level_distribution` (オプション): `-timing` オプションを使用したタイミング レポートに、デザインの長いパスを特定するのに役立つロジック レベル分布を含めます。ロジック レベル分布をレポートすると実行時間が長くなるので、デフォルトではレポートされません。



ヒント: `-logic_level_distribution` オプションは、`-timing` オプションで特定されたクリティカル タイミングパスに対して実行されます。解析でデザインのロジック レベルの数が最も多いすべてのパスを検出するには、`-of_timing_paths` コマンドで `get_timing_paths` オプションを使用して該当するタイミング パスを定義できます。また、タイミング パスに `LOGIC_LEVELS` プロパティを使用してもこの情報を取得できます。

`-logic_level_dist_paths <arg>` (オプション): `-logic_level_distribution` オプションと共に使用でき、ロジック レベル分布を解析する際のクリティカル パスの数を指定します。有効な値は 1 以上で、デフォルトは 1000 です。

`-min_level <arg>` (オプション): ロジック レベル数または配線数が指定した値未満のタイミング パスすべてを 1 つのビンにグループ化します。`<arg>` は 1 以上の値にする必要があります。

`-max_level <arg>` (オプション): ロジック レベル数または配線数が指定した値を超えるタイミング パスすべてを 1 つのビンにグループ化します。`<arg>` は、`-min_level` オプションで指定した値より大きい値にするか、または 0 を指定できます。

`-return_timing_paths` (オプション): このオプションは、`-end_point_clock` および `-logic_levels` オプションと共に使用する必要があります。 `report_design_analysis` コマンドで解析レポートと共にタイミング パス (`timing_path`) オブジェクトが返されるように指定します。 `timing_path` オブジェクトは、タイミング パスを指定可能な `report_timing` コマンドや、2 つ目の `report_design_analysis` コマンドの `-of_timing_paths` オプションに渡すことができます。



ヒント: `-return_timing_paths` オプションを使用した場合、`-timing` オプションは無効になり、タイミング結果は返されません。指定のオプションに一致するタイミング パスがない場合は、「No timing paths returned」というメッセージが返されます。

`-of_timing_paths <args>` (オプション): 解析するタイミング パス オブジェクトを指定します。`-timing` オプションと共に使用する必要があります。このオプションを使用すると、クリティカル パスではなく、指定のパスの特性がレポートされます。 `get_timing_paths` コマンドを使用して `-of_timing_paths` オプションのタイミング パス オブジェクトを指定できます。

`-max_paths <arg>`: クリティカル パス解析でレポートするパスの数を指定します。デフォルトは 1 です。返されるパスの数には、ほかのオプションも影響します。たとえば、`-end_point_clock` で指定したクロックにより返されるパスの数が少なくなることがあります。

`-extend` (オプション): クリティカル パスの拡張セットアップ解析を実行します。ホールド解析は実行されません。`-timing` オプションと共に使用する必要があります。拡張解析では、3 つのパスがレポートされます。

- クリティカル パスの始点で終了するパス。
- クリティカル パス。
- クリティカル パスの終点で開始するパス。

`-routes` (オプション): パスの分布を、ロジック レベル数ではなく、終点クロックごとの配線数に基づいてレポートします。

`-end_point_clock <arg>` (オプション): このオプションは、`-logic_levels` オプションと共に使用する必要があります。指定の終点クロック ドメインで、指定のロジック レベル数タイミング パスを返します。クロックは、名前で指定するか、または `get_clocks` コマンドでオブジェクトとして返すことにより指定します。

`-logic_levels <arg>` (オプション): このオプションは、`-end_point_clock` オプションと共に使用する必要があります。指定の終点クロック ドメインで、指定のロジック レベル数タイミング パスを返します。`-logic_level_distribution` オプションでレポートされるロジック レベル ビンに関連するロジック レベルを指定できます (0、1、2、3、...11、16)。

-qor_summary (オプション): デザインの QoR のサマリを含めます。

-name <arg> (オプション): ツールの GUI モードで表示する [Design Analysis] ウィンドウの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

-no_pr_attribute (オプション): パーシャル リコンフィギュレーション (PR) 属性なしでデザイン解析をレポートします。パーシャル リコンフィギュレーション (PR) デザインでは、論理パスはリコンフィギュラブル パーティション (:RP#) に属するかスタティック領域 (:S) に属するかと、完全に PR 領域または S 領域に含まれるか、領域間にまたがるかが定義されています。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、指定した 2 つのセルの複雑性解析を実行しています。

```
report_design_analysis -complexity -cells {cpuEngine fftEngine}
```

次の例では、指定した境界ボックスの複雑性解析を実行しています。

```
report_design_analysis -complexity \
-bounding_boxes { "CLEL_M_X21Y239:CLEL_R_X28Y254"
"CLEL_R_X18Y171:CLEL_M_X26Y186" }
```

次の例では、デザインのブロック RAM からのワースト クリティカル パスの拡張解析を実行しています。

```
report_design_analysis -timing -of_timing_paths \
[get_timing_paths -from [all_rams]]
```

次の例では、指定のセルから 2 階層レベルまでの複雑性解析を実行し、デザインでタイミングおよび密集解析を実行しています。

```
report_design_analysis -complexity -hierarchical_depth 2 -timing -setup \
-hold -max_paths 10 -logic_level_distribution -logic_level_dist_paths 20 -
congestion
```

次の例では、report_design_analysis コマンドを使用して、指定した終点クロックとロジック レベルのタイミング パスを返し、それらのパスを report_timing コマンドで解析するために渡しています。

```
report_timing -of_objects [report_design_analysis -end_point_clock cpuClk \
-logic_levels 10 -timing -return_timing_paths]
```

関連項目

- [config_design_analysis](#)

- [get_cells](#)
- [get_timing_paths](#)
- [place_design](#)
- [report_timing](#)
- [report_timing_summary](#)
- [set_delay_model](#)
- [set_false_path](#)
- [set_multicycle_path](#)

report_disable_timing

ディスエーブルにしたタイミング アークをレポートします。

構文

```
report_disable_timing [-user_disabled] [-column_style <arg>] [-file <arg>]
[-append] [-cells <args>] [-return_string] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-user_disabled]</code>	ユーザーがディスエーブルにしたタイミング アークのみをレポートします。
<code>[-column_style]</code>	パス レポートの列のスタイルを指定します。有効な値は <code>variable_width</code> 、 <code>anchor_left</code> で、デフォルトは <code>variable_width</code> です。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-append]</code>	結果をファイルの最後に追加します。上書きはしません。
<code>[-cells]</code>	<code>report_disable_timing</code> コマンドを指定のセルに対して実行します。
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

現在の合成済みデザインまたはインプリメント済みデザインのタイミング解析から除外されるタイミング パスのレポートを表示します。

レポートは、タイミング パスに関連付けられているオブジェクトを示す [Cell or Port]、タイミング パスを示す [From] と [To]、条件、およびタイミングから除外されている理由を示す列で構成されます。除外されている理由には次のものがあります。

- `constraint: set_disable_timing` 制約が指定されている
- `constant`: ロジック定数
- `loop`: ロジック ループを断絶する
- `bidirect instance path`: フィードバック パスが双方向インスタンスを通過する
- `bidirect net path`: フィードバック パスが双方向ピン持つネット上にある

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-user_disabled` (オプション): `set_disable_timing` コマンドを使用してユーザーがディスエーブルにしたタイミング アークのみをレポートします。

`-column_style [variable_width | anchor_left]` (オプション): レポートの列幅の調整方法を指定します。このオプションを指定しない場合のデフォルトは `variable_width` です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、 `-file` オプションを使用している場合にのみ使用可能です。

`-cells <args>` (オプション): デザイン全体ではなく、指定したセルに基づく詳細をレポートします。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、タイミング解析から除外されるすべてのタイミング パスがレポートされます。

```
report_disable_timing
```

次の例では、タイミング解析から除外されるタイミング パスのレポートを文字列として出力し、変数に保存してから表示しています。

```
set bad_time [report_disable_timing -return_string]
puts $bad_time
```

関連項目

- [report_timing](#)
- [set_disable_timing](#)

report_drc

DRC を実行します。

構文

```
report_drc [-name <arg>] [-upgrade_cw] [-checks <args>] [-ruledecks <args>]
           [-file <arg>] [-rpx <arg>] [-append] [-waived] [-no_waivers]
           [-return_string] [-quiet] [-verbose]
```

使用法

名前	説明
[-name]	結果を出力する GUI パネルの名前を指定します。
[-upgrade_cw]	すべてのクリティカル 警告違反をエラーにアップグレードします。
[-checks]	DRC チェックを指定します。指定可能なチェックは、get_drc_checks コマンドを使用すると取得できます。
[-ruledecks]	DRC ルール チェックのコンテナであるルール デックを指定します。デフォルトは default です。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-rpx]	レポートのファイル名を指定します。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-waived]	除外されているチェックのみを実行してレポートを生成します。
[-no_waivers]	除外をデisableにします。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

DRC、Report (レポート)、Timing (タイミング)

説明

デザインに対して指定のデザイン ルール チェックまたはルール デックを実行し、検出されたエラーまたは違反をレポートします。

report_drc コマンドを実行するには、デザイン ルール チェックを実行するデザインを開いている必要があります。このコマンドを実行すると、デザイン ルール チェックで検出された違反の結果がレポートされます。違反は、Vivado オブジェクトとして返されます。Vivado オブジェクトは get_drc_violations コマンドを使用してリストでき、現在のデザインのセル、ピン、ポート、ネット、およびサイトに関連付けられています。DRC 違反オブジェクトに関連付けられているセル、ネットなどのデザイン オブジェクトは、get_cells コマンドの -of_objects オプションを使用するなどして取得できます。



ヒント: `report_drc` コマンドをマルチスレッドで実行し、プロセスを高速化できます。 `general.maxThreads` パラメーターの設定に関する詳細は、 `set_param` コマンドを参照してください。

Vivado ツールには、 `report_drc` コマンドで使用可能な定義済みのデザイン ルール チェックが多数含まれています。 `get_drc_checks` コマンドを使用すると、定義済みのデザイン ルール チェックを取得できます。 `create_drc_check` コマンドを使用すると、カスタム デザイン ルール チェックを作成できます。

ルール デッキはデザイン ルール チェックのグループで、I/O プランニングや配置などの FPGA デザイン フローの異なる段階で `report_drc` コマンドにより実行されます。ツールには定義済みのルール デッキが含まれていますが、 `create_drc_ruledeck` コマンドを使用して新しいユーザー定義のルール デッキを作成できます。 `get_drc_ruledecks` コマンドを使用すると、 `report_drc` コマンドで使用可能な定義済みのルール デッキを取得できます。

`-checks` または `-ruledeck` オプションを指定しない場合、 `report_drc` コマンドでデフォルトのルール デッキが実行されます。ユーザー定義 DRC を作成すると、デフォルトのルール デッキに自動的に追加されます。

DRC ルールは、ルール チェック オブジェクトの `IS_ENABLED` プロパティを使用してイネーブルまたはディスエーブルにできます。ルールの `IS_ENABLED` が `false` の場合、 `-checks` または `-ruledeck` で間接的に指定されていても、 `report_drc` コマンドでは実行されません。



ヒント: `reset_drc_check` コマンドを使用すると、DRC ルールのプロパティをデフォルト設定に戻すことができます。

`report_drc` コマンドの現在の結果をリセットして検出された違反をクリアするには、 `reset_drc` コマンドを使用します。

引数

`-name <arg>` (オプション): GUI モードで実行した場合の結果の名前を指定します。

`-upgrade_cw <arg>` (オプション): 検出されたすべてのクリティカル 警告をエラーとしてレポートします。

`-checks <args>` (オプション): DRC レポートを実行するルール チェックをリストします。指定したルール チェックが現在のデザインに対してチェックされます。ルールは、グループ名またはフル キーでリストされます。 `-checks` オプションを使用すると、指定したデザイン ルール チェックで一時的なユーザー定義ルール デッキが作成され、この一時的なルール デッキが実行されます。

注記: `-ruledeck` と `-checks` オプションを同時に使用することはできません。

`-ruledecks <arg>` (オプション): 1 つまたは複数の DRC ルール デッキの名前を指定します。ルール デッキは、DRC ルール チェックのリストの名前です。ツールであらかじめ定義されているルール デッキまたはユーザー定義ルール デッキを指定できます。 `report_drc` コマンドで、指定のルール デッキに追加されているデザイン ルール チェックがデザインに対して実行されます。カスタム ルール デッキを定義するには、 `create_drc_ruledeck` コマンドを使用します。定義済みのルール デッキを取得するには、 `get_drc_ruledecks` コマンドを使用します。

`-file <arg>` (オプション): DRC レポートを保存するファイルを指定します。 `-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、 `-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-waived` (オプション): `create_waiver` コマンドで除外されている DRC チェックのみを実行してレポートします。除外の定義ではなく実際の違反が返されます。除外の定義は、`report_waivers` コマンドでレポートできます。

`-no_waivers` (オプション): `create_waivers` コマンドで定義された除外を無視し、すべての DRC 違反をレポートします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、使用可能なルール デッキをリストしています。結果には、ツールであらかじめ定義されているルール デックと、ユーザー定義ルール デックがすべて含まれます。

```
get_drc_ruledecks
```

次の例では、指定のルール デックに含まれる DRC ルールのリストが返されます。

```
get_drc_checks -of_objects [get_drc_ruledecks placer_checks]
```

次の例では、現在のデザインに対して指定した DRC ルール デックおよびチェックを実行し、その出力を指定したファイルに書き込んでいます。

```
report_drc -ruledecks placer_checks -file C:/Data/DRC_Rpt1.txt
report_drc -checks {IOCNT-1 IOPCPR-1 IOPCMGT-1 IOCTMGT-1 IODIR-1} \
  -file C:/Data/DRC_Rpt1.txt -append
```

注記: 2 番目の `report_drc` コマンドには `-append` が指定されているので、コマンドの結果は指定のファイルの最後に追加されます。

関連項目

- [create_drc_check](#)

- [create_drc_ruledeck](#)
- [create_drc_violation](#)
- [create_waiver](#)
- [get_cells](#)
- [get_drc_checks](#)
- [get_drc_ruledecks](#)
- [get_drc_violations](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)
- [get_sites](#)
- [open_report](#)
- [reset_drc](#)
- [reset_drc_check](#)
- [set_param](#)

report_drivers

HDL ワイヤまたは信号オブジェクトのドライバーと現在の駆動値を表示します。

構文

```
report_drivers [-quiet] [-verbose] <hdl_object>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hdl_object>	レポートする HDL オブジェクトを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

`report_drivers` コマンドは、駆動信号の名前、値、および信号タイプ HDL オブジェクトの現在の値を表示します。

特定の HDL 信号またはネット オブジェクトに値を駆動している信号またはプロセスを判断するために使用します。信号のドライバーは、HDL ソース ファイルで信号への代入を実行する文です。

`report_drivers` コマンドの出力フォーマットは次のとおりです。

```
Drivers for <hdl_object>
  <Value of HDL Object>: Net <Hierarchical name of the probed signal>
    [ Declared Net : <The declared signal to which the probed signal is
connected>]
  <Value of Driver> : Driver <Hierarchical name of the HDL process
containing
the driver> at <file_name>:<line number>
```

注記: Declared Net (宣言されたネット) は、シミュレーションの現在のスコープのために、プローブされた信号名が実際に宣言された信号が階層名と異なる場合にレポートされます。プローブされた信号に対して宣言されたネットの各ビットが表示されます。

`report_drivers` コマンドで返される信号の値は、シミュレーションのステートによって異なります。次に、シミュレーションの前後でレポートを生成した場合の例を示します。

```
current_scope /testbench/dut
report_drivers leds_n[3:0]
Drivers for /testbench/dut/LEDS_n[3:0]
0 : Net /testbench/dut/LEDS_n[0]
  Declared Net : /testbench/leds_n[3]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
0 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
```

```

0 : Net /testbench/dut/LEDS_n[1]
Declared Net : /testbench/leds_n[2]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
0 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
0 : Net /testbench/dut/LEDS_n[2]
Declared Net : /testbench/leds_n[1]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
1 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
1 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
1 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
X : Net /testbench/dut/LEDS_n[3]
Declared Net : /testbench/leds_n[0]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
0 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
run all
report_drivers leds_n[3:0]
Drivers for /testbench/dut/LEDS_n[3:0]
0 : Net /testbench/dut/LEDS_n[0]
Declared Net : /testbench/leds_n[3]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
0 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
1 : Net /testbench/dut/LEDS_n[1]
Declared Net : /testbench/leds_n[2]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
0 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
0 : Net /testbench/dut/LEDS_n[2]
Declared Net : /testbench/leds_n[1]
1 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
1 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
1 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
1 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184
0 : Net /testbench/dut/LEDS_n[3]
Declared Net : /testbench/leds_n[0]
0 : Driver /testbench/dut/line__187 at C:/Data/sources/sinegen_demo.vhd:187
0 : Driver /testbench/dut/line__186 at C:/Data/sources/sinegen_demo.vhd:186
1 : Driver /testbench/dut/line__185 at C:/Data/sources/sinegen_demo.vhd:185
0 : Driver /testbench/dut/line__184 at C:/Data/sources/sinegen_demo.vhd:184

```

注記: シミュレーションの現在のスコープがテストベンチの最上位とは異なるので、宣言されたネットがレポートされています。

このコマンドを実行すると、指定したオブジェクトのドライバーのレポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hdl_objects>: ドライバーをレポートする VHDL 信号または Verilog ワイヤを指定します。HDL オブジェクトは、名前または `get_objects` コマンドで指定します。

例

次の例では、`get_objects` コマンドで返された HDL オブジェクトのドライバーをレポートしています。

```
report_drivers [get_objects leds_n]
```

関連項目

- [current_scope](#)
- [get_objects](#)

report_environment

システム情報をレポートします。

構文

```
report_environment [-file <arg>] [-format <arg>] [-append] [-return_string]
                  [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	システム情報を指定のファイルに書き込みます。
[-format]	レポートのフォーマットを指定します。有効な値は text および xml です。-file を使用した場合にのみ適用されます。xml を指定した場合、-append は使用できません。デフォルトは text です。
[-append]	レポートを既存のファイルの最後に追加します。
[-return_string]	レポートの内容を文字列値として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

ツールを実行しているシステム環境の詳細をレポートします。オペレーティング システムのバージョン、CPU、メモリ、使用可能なディスク容量、さまざまな環境変数の設定が含まれます。

デフォルトでは出力は標準出力に表示されますが、ファイルに書き込むこともできます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。-append オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-format [text | xml] (オプション): 出力レポートのデフォルト フォーマットは text です。XML レポートを出力することも可能です。XML 出力は -file オプションを指定した場合にのみ有効で、-append オプションと同時に使用することはできません。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: -append オプションは、-file オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在の環境を指定したファイルに保存しています。

```
report_environment -file C:/Data/toolEnv.txt
```

report_exceptions

タイミング例外をレポートします。

構文

```
report_exceptions [-from <args>] [-rise_from <args>] [-fall_from <args>]
[-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
[-rise_through <args>] [-fall_through <args>] [-ignored] [-summary]
[-coverage] [-ignored_objects] [-count_objects]
[-write_merged_exceptions] [-write_valid_exceptions] [-no_header]
[-file <arg>] [-append] [-return_string] [-name <arg>] [-quiet]
[-verbose]
```

使用法

名前	説明
[-from]	タイミングパスの始点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_from]	ピン、ポート、セル、またはクロックの立ち上がりエッジを始点として指定します。
[-fall_from]	ピン、ポート、セル、またはクロックの立ち下がりエッジを始点として指定します。
[-to]	タイミングパスの終点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_to]	ピン、ポート、セル、またはクロックの立ち上がりエッジを終点として指定します。
[-fall_to]	ピン、ポート、セル、またはクロックの立ち下がりエッジを終点として指定します。
[-through]	タイミングパスの通過点 (ピン、ポート、セル、またはネット) を指定します。
[-rise_through]	ピン、ポート、セル、またはネットの立ち上がりエッジを通過点として指定します。
[-fall_through]	ピン、ポート、セル、またはネットの立ち下がりエッジを通過点として指定します。
[-ignored]	無視される例外のみをレポートします。
[-summary]	すべての例外のサマリをレポートします。
[-coverage]	すべてのタイミング例外の適用範囲をレポートします。
[-ignored_objects]	無視される始点および終点をレポートします。
[-count_objects]	タイミング例外に含まれるオブジェクトの数をレポートします。
[-write_merged_exceptions]	有効なタイミング例外と無効なタイミング例外の両方をレポートします。
[-write_valid_exceptions]	有効なオブジェクトのみのタイミング例外をレポートします。
[-no_header]	ヘッダーなしのレポートを生成します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。

名前	説明
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-name]</code>	結果を出力する GUI パネルの名前を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

現在のデザインのタイミング制約で定義されているセットアップおよびホールド チェックに適用されるタイミング例外をレポートするか、指定したタイミング パスの例外をレポートします。

タイミング例外は、`set_false_path` や `set_multicycle_path` などのデザインのタイミング パスのデフォルトを変更するタイミング制約で定義できます。

デフォルトでは例外レポートは標準出力に出力されますが、その後の処理用にファイルまたは Tcl 文字列変数に出力することもできます。

引数

`-from <args>` (オプション): 例外をレポートするタイミング パスの始点を指定します。

`-rise_from <args>` (オプション): 立ち上がりエッジの例外をレポートするタイミング パスの始点を指定します。

`-fall_from <args>` (オプション): 立ち下がりエッジの例外をレポートするタイミング パスの始点を指定します。



重要: `report_exceptions` コマンドで `-from/-through/-to` オプションを使用すると、同じ `-from/-through/-to` コマンド ライン オプションを使用して定義されたタイミング例外のみがレポートされます。指定したパターンは異なるものにできますが、例外としてレポートされるようにするにはセル、ピン、およびポート オブジェクトが同じである必要があります。

`-to <args>` (オプション): 例外をレポートするタイミング パスの終点を指定します。

`-rise_to <args>` (オプション): 立ち上がりエッジの例外をレポートするタイミング パスの終点を指定します。

`-fall_to <args>` (オプション): 立ち下がりエッジの例外をレポートするタイミング パスの終点を指定します。

`-through <args>` (オプション): タイミング パスが通過するピン、セル、またはネットを指定します。

`-rise_through <args>` (オプション): 立ち上がりタイミング パスが通過するピン、セル、またはネットを指定します。

`-fall_through <args>` (オプション): 立ち下がりタイミング パスが通過するピン、セル、またはネットを指定します。

`-ignored` (オプション): 現在のデザインで Vivado タイミング エンジンにより無視されたタイミング パス例外をレポートします。制約の定義が正しくなかったり、デザイン オブジェクトがなかったりすると、制約が無視されることがあります。

`-summary` (オプション): すべてのタイミング例外のサマリをレポートします。

`-coverage` (オプション): すべてのタイミング例外の適用範囲をレポートします。適用範囲は、`-from`、`-through`、`-to` オプションで指定されるピン数に対してタイミング例外で達成されるピン数の割合 (%) で示されます。

`-ignored_objects` (オプション): タイミング パス例外の一部で、タイミング エンジンにより無視された始点および終点をレポートします。

`-count_objects` (オプション): 例外レポートに含まれるオブジェクトの数を返します。

`-write_valid_exceptions` (オプション): 有効なタイミング例外をレポートします。有効な例外は、始点と終点が無効です。

`-write_merged_exceptions` (オプション): 有効なタイミング例外と無効なタイミング例外の両方をレポートします。

`-no_header` (オプション): レポートにヘッダーを含めません。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。デフォルトでは、タイミング例外は標準出力または Tcl コンソールにレポートされます。`-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するレポートの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインのタイミング例外をすべてレポートしています。

```
report_exceptions
```

次の例では、現在のデザインで無視または無効になっているタイミング例外をすべてレポートしています。

```
report_exceptions -ignored
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [report_timing](#)
- [report_timing_summary](#)
- [set_false_path](#)
- [set_input_delay](#)
- [set_max_delay](#)
- [set_min_delay](#)
- [set_multicycle_path](#)
- [set_output_delay](#)

report_frames

`current_scope` がサブプログラム内で待機しているプロセスの場合に、スタック フレームをテキスト形式で表示します。

構文

```
report_frames [-quiet] [-verbose]
```

戻り値

文字列

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の HDL プロセス スコープ (`current_scope`) のサブプログラム呼び出し階層で、サブプログラム名文字列のリストと呼び出し元 HDL プロセスを返します。HDL プロセスから、階層内の最も最近のサブプログラムまでリストされます。各フレームには、フレーム インデックスが関連付けられます。最も最近のサブプログラムが一番上に表示され、これがインデックス 0 になります。シンボル (->) は `current_frame` を示します。

デフォルトでは、最も最近呼び出されたサブプログラム フレームが `current_frame` となります。
`current_frame` コマンドを使用すると、ほかのフレームを選択できます。詳細モードでは、各呼び出しのソース行番号とファイル名が表示されます。



重要: `report_frames` は `current_scope` に厳密に従います。`current_scope` がサブプログラム内で待機中の HDL プロセス スコープでない場合は、このコマンドで空のリストが返されます。

このコマンドを実行すると、現在のインスタンスのデザイン オブジェクトの名前が返されるか、正常に実行されなかった場合は何も返されません。

引数

`-verbose` (オプション): 詳細モード。各呼び出しのソース行番号とファイル名が表示されます。ソース ファイル名は、完全な絶対パスで示されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

例

サンプル デザイン:

```
module top;

    int i;

    function void f(input int in1);
        automatic int a;
        a = in1 + 7;
        $display($time, " in f :: a %d in1 %d ", a, in1);
    endfunction

    task automatic t(input int in2);
        int b;
        b = in2 + 10;
        $display($time, " in t :: in2 %d b %d ", in2, b);
        #5;
        f(b);        // Case C
        $display($time, " Back in t : after wait and f(%d) ", b);
    endtask

    initial begin                                // "/top/Initial18_0"
        $display($time, " in initial 1 ");
        i = 200;
        t(i);        // Case B
        $display($time, " Back in initial 1 after t(%d) ", i);
    end

    initial begin                                // "/top/Initial25_1"
        $display($time, " in initial 2 ");
        #2;
        f(50);        // Case A
        $display($time, " Back in initial 2 after f(50) ");
    end
endmodule
```

シミュレーションが関数 f の Case C の呼び出しで停止した場合、関数 f はタスク t の Case C で呼び出され、タスク t はプロセス /top/Initial18_0 の Case B で呼び出されます。

```
> current_scope
/top/Initial18_0

1. > report_frames
-> 0 : f
    1 : t
    2 : /top/Initial18_0
2. > report_frames -verbose
-> 0 : f @top.v:6
    1 : t @top.v:15
    2 : /top/Initial18_0 @top.v:21
```

関連項目

- [current_scope](#)
- [current_frame](#)

report_high_fanout_nets

ファンアウトの大きいネットをレポートします。

構文

```
report_high_fanout_nets [-file <arg>] [-format <arg>] [-append]
    [-ascending] [-timing] [-histogram] [-load_types] [-clock_regions]
    [-slr] [-max_nets <arg>] [-fanout_greater_than <arg>]
    [-fanout_lesser_than <arg>] [-name <arg>] [-cells <args>]
    [-clocks <args>] [-return_string] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-format]	レポートのフォーマットを指定します。有効な値は text、xml で、デフォルトは text です。-file を使用した場合にのみ適用されます。xml を指定した場合、-append は使用できません。デフォルトは text です。
[-append]	既存のファイルの最後に追加します。
[-ascending]	ネットを昇順にレポートします。
[-timing]	ネットのワースト スラック値およびワースト遅延値をレポートします。
[-histogram]	ファンアウトの大きいネットのヒストグラムをレポートします。
[-load_types]	ロードの詳細をレポートします。
[-clock_regions]	クロック領域でのロードの分配をレポートします。
[-slr]	SLR でのロードの分配をレポートします。
[-max_nets]	レポートするネットの最大数を指定します。デフォルトは 10 です。
[-fanout_greater_than]	ファンアウトの数が指定した整数値より大きいネットをレポートします。デフォルトは 0 です。
[-fanout_lesser_than]	ファンアウトの数が指定した整数値未満のネットをレポートします。デフォルトは INT_MAX です。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-cells]	指定したセルのネットをレポートします。
[-clocks]	指定したクロックのネットをレポートします。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

デザインのネットのファンアウトを、ファンアウトの大きいネットから順にレポートします。オプションを使用してレポートの内容を制御できます。

このコマンドは、インプリメント済みデザインまたは合成済みネットリストに対して実行できますが、インプリメント済みデザインで実行したほうがより完全な結果が得られます。

このコマンドを実行すると、デザインのネットのファンアウト レポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていないければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-format [text | xml]` (オプション): 出力レポートのデフォルト フォーマットは `text` です。XML レポートを出力することも可能です。XML 出力は `-file` オプションを指定した場合にのみ有効で、`-append` オプションと同時に使用することはできません。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-ascending` (オプション): ネットを昇順にレポートします。

`-timing` (オプション): ファンアウトの大きいネット上のワースト スラック (WNS) およびワースト遅延をレポートします。

注記: `-timing` オプションを使用すると `report_high_fanout_nets` コマンドの実行時間が長くなり、`-histogram` オプションはサポートされません。

`-histogram` (オプション): レポートをヒストグラム形式で表示し、ファンアウト数、ファンアウトの大きいネットのデザイン全体での数と割合を示します。

注記: このオプションは、`-ascending`、`-timing`、`-load_types`、`-clock_regions`、`-slr`、`-max_nets` オプションと一緒に使用することはできません。

`-load_types` (オプション): ネット上のさまざまなロード タイプをレポートします。ロード タイプ (データ、クロック、セット、リセットなど) またはロードが配置されているデバイス リソース (スライス、I/O など) を基準に並べ替えることができます。 `report_high_fanout_nets` を配置されていない合成済みデザインに対して実行した場合は、ロード タイプのみがレポートされます。

`-clock_regions` (オプション): クロック領域でのロードの分配をレポートします。このオプションは配置後にのみ使用可能で、ネット上のさまざまなロードが配置されているクロック領域をレポートします。

注記: `-load_types` と `-clock_regions` オプションを同時に使用することはできません。

`-slr` (オプション): SLR でのロードの分配をレポートします。このオプションは配置後にのみ使用可能で、ネット上のさまざまなロードが配置されている SLR をレポートします。

`-max_nets <arg>` (オプション): レポートするネット数を指定します。デフォルトは 10 です。

`-fanout_greater_than <arg>` (オプション): ファンアウト数が指定した値より大きいネットのみをレポートします。整数値で指定します。デフォルト値は 0 です。

`-fanout_lesser_than <arg>` (オプション): ファンアウト数が指定した値未満のネットのみをレポートします。整数値で指定します。デフォルト値は INT_MAX です。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するファンアウトの大きいネットのレポートの名前を指定します。指定した名前が開いているレポート ビューで既に使用されている場合は、そのビューが閉じられ、新しいレポートが開きます。

`-cells args` (オプション): デザインの指定したセル インスタンスに接続されているネットをレポートします。セルは、名前で指定するか、または `get_cells` コマンドを使用してセル オブジェクトとして指定します。

`-clocks <args>` (オプション): 指定したクロックのネットをレポートします。クロックは、名前で指定するか、または `get_clocks` コマンドを使用してクロック オブジェクトとして指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ファンアウトが 50 以上のネットを上位から 100 個レポートし、結果のヒストグラムを返しています。

```
report_high_fanout_nets -fanout_greater_than 50 -max_nets 100 -histogram
```

次の例では、指定したクロックのネットのファンアウトをレポートしています。

```
report_high_fanout_nets -clocks [get_clocks gt*]
```

次の例では、ファンアウトが 10 未満のネットをレポートし、結果を Tcl 変数として保存しています。

```
set myRep [report_high_fanout_nets -fanout_lesser_than 10 -return_string]
```

関連項目

- [get_cells](#)

report_hw_axi_txn

ハードウェア AXI トランザクション データをレポートします。

構文

```
report_hw_axi_txn [-w <arg>] [-t <arg>] [-quiet] [-verbose]
<hw_axi_txns>...
```

使用法

名前	説明
[-w]	出力ラインごとの出力データ バイトを指定します。デフォルトは 8 です。
[-t]	d[SIZE]: 符号付き 10 進数 b[SIZE]: バイナリ o[SIZE]: 8 進数 u[SIZE]: 符号なし 10 進数 x[SIZE]: 16 進数 SIZE は整数ごとのバイト数を示します。デフォルトは x4 (4 バイトの 16 進数) です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_axi_txns>	レポートするハードウェア AXI トランザクション オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

JTAG to AXI Master コア (hw_axi) の指定した AXI トランザクションの結果をレポートします。

このコマンドは、既存の hw_axi オブジェクトにハードウェア AXI トランザクション (hw_axi_txn) オブジェクトを作成し、hw_axi を実行して定義したトランザクションを実行した後に使用できます。

JTAG to AXI Master コアは、Tcl コマンドを使用してのみ制御可能です。AXI 読み出しおよび書き込みトランザクションの発行と実行には、create_hw_axi_txns コマンドを使用します。これらのコマンドを発行する前に、reset_hw_axi コマンドを使用して JTAG to AXI Master コアをリセットしておくことが重要です。

このコマンドを実行すると、トランザクション データが指定のフォーマットでレポートされるか、正常に実行されなかった場合はエラーが返されます。

引数

-w <arg> (オプション): 出力行ごとのデータ バイト数を指定します。デフォルトは行ごとに 8 バイトです。

-t <arg> (オプション): 出力データのフォーマットとサイズを指定します。トランザクション データのフォーマットとして有効な値は、次のとおりです。

- d[SIZE]: 符号付き 10 進数 (SIZE は整数ごとのバイト数)
- b[SIZE]: バイナリ (SIZE は整数ごとのバイト数)

- o[SIZE]: 8 進数 (SIZE は整数ごとのバイト数)
- u[SIZE]: 符号なし 10 進数 (SIZE は整数ごとのバイト数)
- x[SIZE]: 16 進数 (SIZE は整数ごとのバイト数)
- デフォルトの出力フォーマットは x4 (4 バイトの 16 進数) です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_axi_txns> (必須): レポートする hw_axi_txn オブジェクトを指定します。hw_axi_txn は、get_hw_axi_txns コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、hw_axi コアを AXI トランザクションを開始する既知のステートにリセットし、AXI コアの最初の 4 つのロケーションの内容を読み出す AXI トランザクションを作成して hw_axi に関連付け、hw_axi を実行してトランザクションを処理し、最後にトランザクションで読み出したデータをレポートしています。

```
reset_hw_axi [get_hw_axis hw_axi_1]
create_hw_axi_txn read_txn [get_hw_axis hw_axi_1] -type READ \
    -address 00000000 -size 32 -len 4 -cache 3 -id 0
run_hw_axi [get_hw_axi_txns [get_hw_axis]]
report_hw_axi_txn [get_hw_axi_txns read_txn]
```

次の例では、AXI 読み出しおよび書き込みトランザクションを作成し、hw_axi を実行して、その結果をレポートしています。

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
    -data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
    -len 4 -type read

run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report

run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report

close_hw_target;
disconnect_hw_server;
```

関連項目

- [delete_hw_axi_txn](#)
- [get_hw_axis](#)
- [get_hw_axi_txns](#)

- [refresh_hw_axi](#)
- [reset_hw_axi](#)

report_hw_ddrmc

Versal の統合およびソフト メモリ コントローラー (DDRMC) のメモリ コンフィギュレーション、キャリブレーション ステータス、段階、およびウィンドウ マージン データをフォーマットされたレポートとして出力します。

構文

```
report_hw_ddrmc [-file <arg>] [-append] [-return_string] [-quiet]
                 [-verbose] <hw_objects>
```

使用法

名前	説明
[-file]	レポート結果を出力するファイルの名前 (完全パス) を指定します。
[-append]	レポートの結果を指定したファイルに上書きするのではなく追加します。
[-return_string]	レポート結果を文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_objects>	ハードウェア DDRMC オブジェクトを指定します。

カテゴリ

Report (レポート)、Hardware (ハードウェア)

説明

メモリ IP のハードウェア コンフィギュレーション、キャリブレーション、マージンに関する情報をフォーマットしてレポートします。Vivado ロジック解析または Vivado Lab Edition で使用可能なグラフィカル マージン スキャンプロットは含まれません。

Vivado ツールでは、デザインにインプリメントされている Versal メモリ コントローラーはハードウェア DDRMC (hw_ddrmc) オブジェクトに関連付けられます。これらの hw_ddrmc オブジェクトを使用して、メモリ インターフェイス デザインのキャリブレーション、読み出し、および書き込みウィンドウ マージンを検証できます。ハードウェア マネージャー GUI を使用してキャリブレーション ステータスを確認し、クロックの立ち上がりエッジおよび立ち下がりエッジの読み出しマージン、単純なパターンおよび複雑なパターン両方の書き込みマージンを検証できます。また、ILA および VIO コアを使用して、読み出しおよび書き込みのデータ インテグリティを検証できます。

このコマンドを実行すると、レポートされたデータが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (オプション): `get_hw_ddrmcs` コマンドを使用して `hw_ddrmc` オブジェクトとして指定する必要があります。

例

次の例は、`hw_ddrmc` オブジェクトに関するレポートを生成し、指定したテキスト ファイルに出力します。

```
report_hw_ddrmc -file C:/Data/ddrmc_report.txt [get_hw_ddrmcs]
```

関連項目

- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_ddrmcs](#)
- [refresh_hw_ddrmc](#)

report_hw_mig

ハードウェア MIG キャリブレーションのステータスおよびマージン データをフォーマットしてレポートします。

構文

```
report_hw_mig [-file <arg>] [-append] [-return_string] [-quiet] [-verbose]
               <hw_objects>
```

使用法

名前	説明
[-file]	レポート結果を出力するファイルの名前 (完全パス) を指定します。
[-append]	レポートの結果を指定したファイルに上書きするのではなく追加します。
[-return_string]	レポート結果を文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hw_objects>	ハードウェア MIG オブジェクトを指定します。

カテゴリ

[Report \(レポート\)](#)、[Hardware \(ハードウェア\)](#)

説明

メモリ IP のハードウェア コンフィギュレーション、キャリブレーション、マージンに関する情報をフォーマットしてレポートします。Vivado ロジック解析または Vivado Lab Edition で使用可能なグラフィカル マージン スキャンプロットは含まれません。

Vivado ツールでは、デザインにインプリメントされているメモリ コントローラーはハードウェア MIG (hw_mig) オブジェクトに関連付けられます。hw_mig オブジェクトを使用して、メモリ インターフェイス デザインのキャリブレーション、読み出し、および書き込みウィンドウ マージンを検証できます。ハードウェア マネージャー GUI を使用してキャリブレーション ステータスを確認し、クロックまたは DQS の立ち上がりエッジおよび立ち下がりエッジの読み出しマージンを検証し、単純なパターンおよび複雑なパターンの両方のマージンを書き込むことができます。また、ILA コアを使用して、読み出しおよび書き込みのデータ インテグリティを検証できます。

このコマンドを実行すると、レポートされたデータが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (オプション): `hw_mig`、ハードウェア デバイス (`hw_device`)、ハードウェア ターゲット (`hw_target`)、またはハードウェア サーバー (`hw_server`) オブジェクトを指定します。

注記: オブジェクトは、名前で指定するのではなく、`get_hw_migs` などの `get_hw_XXX` コマンドを使用して指定する必要があります。

例

次の例では、`hw_mig` オブジェクトに関するレポートを生成し、指定のテキスト ファイルに出力しています。

```
report_hw_mig -file C:/Data/hw_mig_report.txt [get_hw_migs]
```

関連項目

- [commit_hw_mig](#)
- [current_hw_device](#)
- [get_hw_migs](#)
- [implement_mig_cores](#)
- [refresh_hw_mig](#)
- [set_property](#)

report_hw_targets

ハードウェア オブジェクトのプロパティをレポートします。

構文

```
report_hw_targets [-quiet] [-verbose]
```

戻り値

ハードウェア オブジェクトを指定します。

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

現在のハードウェア サーバー (`current_hw_server`) 上のハードウェア ターゲット (`hw_target`) すべてのコンフィギュレーションに関連するプロパティを返します。このコマンドでレポートされる情報は、次のとおりです。

- サーバーのプロパティ 情報: `current_hw_server` の HOST および PORT プロパティ。
- ターゲットのプロパティ 情報: `hw_server` の各ターゲットの NAME、FREQUENCY、DEVICE_COUNT、および SVF。
- デバイスのプロパティ 情報: 指定の `hw_target` 上の各デバイスに対して PART、ID CODE、IR LENGTH、MASK、PROGRAMMING、および PROBES FILE。

このコマンドを実行すると、正常に実行された場合は指定の情報が返され、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、接続された hw_server 上にあるすべてのターゲットのプロパティ 情報レポートしています。

```
report_hw_targets
```

関連項目

- [connect_hw_server](#)
- [create_hw_target](#)
- [get_hw_targets](#)

report_incremental_reuse

指定のデザイン チェックポイントから現在のデザインにインクリメンタル再利用可能な量をレポートします。

構文

```
report_incremental_reuse [-file <arg>] [-append] [-cells <args>]
                        [-hierarchical] [-hierarchical_depth <arg>] [-return_string] [-quiet]
                        [-verbose]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	既存のファイルの最後に追加します。
[-cells]	指定したセルのインクリメンタル再利用をレポートします。
[-hierarchical]	テキスト形式の階層インクリメンタル再利用レポートを生成します。
[-hierarchical_depth]	テキスト形式の階層インクリメンタル再利用レポートのレベルを指定します。デフォルトは 0 です。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

インクリメンタル インプリメンテーション フローで使用し、現在のデザインと `read_checkpoint - incremental` コマンドを使用して読み込んだインクリメンタル チェックポイントで一致している量をレポートします。

読み込まれたインクリメンタル チェックポイントを現在のデザインに対して解析し、インクリメンタル配置配線を実行するのに十分な類似性があるかどうかを確認できます。現在のデザインとチェックポイントの類似性が低い場合は、チェックポイントをインクリメンタル配置配線に使用することは推奨されません。インクリメンタル配置および配線の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

現々のデザインと読み込んでいるインクリメンタル チェックポイントの類似性が低い場合は、`open_run` または `read_checkpoint` を使用して元のデザインを復元する必要があります。または、`read_checkpoint - incremental` コマンドを再度実行して現在のデザインに新しいインクリメンタル チェックポイントを読み込むこともできます。

パーシャル リコンフィギュレーション (PR) デザインでは、リコンフィギュラブル モジュール (RM) の一致するセルの割合、RM で再利用されるセルの割合、および RM で固定されるセルの割合が荒れポートされます。また、リコンフィギュラブル モジュール サマリの表もレポートに追加されています。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。デフォルトでは、レポートは Tcl コンソールに表示されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、 `-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-name <arg>` (オプション): GUI モードで実行した場合の結果の名前を指定します。

`-cells <args>` (オプション): DCP ファイルから使用するセルを指定します。

`-hierarchical` (オプション): テキスト形式の階層インクリメンタル再利用レポートを生成します。

`-hierarchical_depth <arg>` (オプション): テキスト形式の階層インクリメンタル再利用レポートのレベルを指定します。デフォルトは 0 です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、 `set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインにインクリメンタル チェックポイントを読み込み、読み込まれたインクリメンタル チェックポイントと現在のデザインの類似性をレポートしています。

```
read_checkpoint -incremental C:/Data/reuse_checkpoint1.dcp
report_incremental_reuse
```

関連項目

- [open_run](#)
- [place_design](#)
- [read_checkpoint](#)
- [route_design](#)

report_io

デバイスのすべての I/O サイトに関する情報を表示します。

構文

```
report_io [-file <arg>] [-name <arg>] [-append] [-format <arg>]  
          [-return_string] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-append]	既存のファイルの最後に追加します。
[-format]	レポートのフォーマットを指定します。有効な値は text、xml で、デフォルトは text です。-file を使用した場合にのみ適用されます。xml を指定した場合、-append は使用できません。デフォルトは text です。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在のデザインの I/O バンクの詳細をレポートします。ターゲット パーツ、パッケージ、スピード グレード、デバイス上の各ピンに関する情報などが含まれます。

このコマンドを実行すると、レポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-name <arg> (オプション): GUI モードで実行した場合のレポートされる結果の名前を指定します。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。`-format xml` オプションと共に使用することはできません。

`-format [xml | text]` (オプション): 出力のフォーマットを XML または ASCII 形式のテキスト ファイルに指定します。デフォルトは `text` です。

注記: フォーマットは `-file` を指定した場合にのみ適用され、それ以外は無視されます。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの I/O ブロックをレポートしています。

```
report_io
```

関連項目

- [report_route_status](#)

report_ip_status

プロジェクトの IP インスタンスのステータスをレポートします。

構文

```
report_ip_status [-name <arg>] [-file <arg>] [-append] [-return_string]
                 [-license_status] [-resource_data] [-quiet] [-verbose]
```

戻り値

コマンドが正常に実行された場合は True

使用法

名前	説明
[-name]	GUI パネルに表示する結果の名前を指定します。
[-file]	結果を出力するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	既存のファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-license_status]	各 IP の生成された出力のライセンス ステータスをレポートします。
[-resource_data]	各 IP インスタンスのリソース データ使用率をレポートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[IPFlow \(IP フロー\)](#)

説明

現在のプロジェクトに含まれる IP コアを解析し、最新の IP カタログに対する IP のステートをレポートします。IP ステータス レポートには、次の情報が含まれます。

- Instance Name: 現在のプロジェクトに含まれる IP コア名。
- IP Status: 現在のプロジェクトに含まれる IP コアのステートの説明。
- Recommendation: ステータスに基づき推奨される操作。
- Lock Status: 現在のプロジェクトに含まれる IP コアのロック ステータスの説明。
- Change Log: カタログ内の IP アップデートの変更ログの場所。変更ログには、最新 IP の変更に関する説明が含まれます。
- IP Name: IP コアのカタログ内の名前。
- IP Version: 現在のプロジェクトで使用されている IP のバージョン。

- New Version: IP カタログの最新バージョン。
- New license: 新しい IP バージョンのライセンス ステータス。
- Original Part: カタログ内の IP に関連付けられている元のパーツ。

最新でない IP またはロックされた IP は、アップグレードして出力ファイルを再生成する必要がある場合があります。詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896) を参照してください。

report_ip_status コマンドは、ローカル マシンまたはライセンス サーバー上で使用可能なライセンスをチェックし、現在のプロジェクトに含まれるすべての IP コアのライセンスを検索します。ライセンスが見つかった場合、ライセンス情報が表示されます。ライセンスが見つからない場合は、そのことが表示されます。

このコマンドを実行すると、IP ステータス レポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-name <arg> (オプション): GUI モードで実行した場合の結果の名前を指定します。

-file <arg> (オプション): レポートを保存するファイルを指定します。-append オプションが指定されていないければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: -append オプションは、-file オプションを使用している場合にのみ使用可能です。

-return_string (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを -file オプションと共に使用することはできません。

-license_status (オプション): デザインで使用されている IP のさまざまな要件とライセンス ステータスのみを表示します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、IP ステータス レポートを指定のファイルに出力しています。ファイルが存在する場合は上書きされます。

```
report_ip_status -file C:/Data/reports/ip_status.txt -append
```

関連項目

- [get_ips](#)
- [import_ip](#)
- [read_ip](#)
- [upgrade_ip](#)

report_methodology

設計手法チェック

構文

```
report_methodology [-name <arg>] [-cells <args>] [-checks <args>]
                  [-file <arg>] [-rpx <arg>] [-append] [-waived] [-no_waivers]
                  [-slack_lesser_than <arg>] [-return_string] [-quiet] [-verbose]
```

使用法

名前	説明
[-name]	結果を出力する GUI パネルの名前を指定します。
[-cells]	report_methodology コマンドを指定したセルに対して実行します。
[-checks]	設計手法チェックをレポートします。指定可能なチェックは、get_methodology_checks コマンドを使用すると取得できます。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-rpx]	レポートのファイル名を指定します。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-waived]	除外されているチェックのみを実行してレポートを生成します。
[-no_waivers]	除外をディスエーブルにします。
[-slack_lesser_than]	SYNTH ルールのスラックしきい値を ns (浮動小数点) で指定します。デフォルトは 2.0 です。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Methodology (設計手法)、Report (レポート)、Timing (タイミング)

説明

現在のデザインに対して指定の設計手法チェックを実行し、検出されたエラーまたは違反をレポートします。

設計手法チェックは特別なデザイン ルール チェック (DRC) であり、別の Tcl コマンドで実行します。設計手法チェックはデザイン フローの重要な一部であり、インプリメンテーション後、ビットストリームの生成前に実行する必須の手順と考えてください。



ヒント: 別の report_methodology コマンドで実行することを除けば、チェックはほかのデザイン ルール チェックと同じです。

`report_methodology` コマンドを実行するには、デザイン ルール チェックを実行するデザインを開いている必要があります。このコマンドを実行すると、デザイン ルール チェックで検出された違反の結果がレポートされます。違反は、Vivado オブジェクトとして返されます。Vivado オブジェクトは `get_methodology_violations` コマンドを使用してリストでき、現在のデザインのセル、ピン、ポート、ネット、およびサイトに関連付けられています。設計手法違反オブジェクトに関連付けられているセル、ネットなどのデザイン オブジェクトは、`get_cells` コマンドの `-of_objects` オプションを使用するなどして取得できます。

`report_methodology` コマンドは、`methodology` ルール デックを実行します。`-checks` オプションを使用して、実行するチェックを実行することも可能です。デフォルト ルール デックの設計手法チェックは、ルール チェック オブジェクトの `IS_ENABLED` プロパティを使用してイネーブルまたはディスエーブルにできます。

```
set_property IS_ENABLED FALSE [get_methodology_checks PDRC-190]
```

ルールの `IS_ENABLED` が `false` の場合、`report_methodology` コマンドでは実行されません。



ヒント: `reset_methodology_check` コマンドを使用すると、設計手法ルールのプロパティをデフォルト設定に戻すことができます。

`report_methodology` コマンドの現在の結果をリセットして検出された違反をクリアするには、`reset_methodology` コマンドを使用します。

引数

`-name <arg>` (オプション): GUI モードで実行した場合の結果の名前を指定します。

`-cells <arg>` (オプション): `report_methodology` を実行する際に使用するセルを指定します。セルはデザインの階層モジュールである必要があります。このオプションは、最上位セル以外のセルを解析する場合に使用します。セルは、名前で指定するか、または `get_cells` コマンドを使用してオブジェクトとして指定します。

`-checks <args>` (オプション): 設計手法レポートを実行するルール チェックをリストします。指定したルール チェックが現在のデザインに対してチェックされます。ルールは、グループ名またはフル キーでリストされます。`-checks` オプションを使用すると、指定したデザイン ルール チェックで一時的なユーザー定義ルール デックが作成され、この一時的なルール デックが実行されます。

`-file <arg>` (オプション): 設計手法レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-waived` (オプション): `create_waiver` コマンドで除外されている設計手法チェックのみを実行してレポートします。除外の定義ではなく実際の違反が返されます。除外の定義は、`report_waivers` コマンドでレポートできます。

`-no_waivers` (オプション): `create_waivers` コマンドで定義された除外を無視し、すべての設計手法違反をレポートします。

`-slack_lesser_than <arg>` (オプション): 算出されたスラック値が指定した値より小さい場合にのみ SYNTH 設計手法違反をレポートします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインに対してデフォルトの設計手法チェックを実行し、その結果を指定したファイルに記述しています。

```
report_methodology -file C:/Data/methodology_Rpt1.txt -append
```

注記: `-append` オプションは、結果を指定したファイルの最後に追加します。

関連項目

- [get_methodology_checks](#)
- [get_methodology_violations](#)
- [open_report](#)
- [report_drc](#)
- [reset_methodology](#)

report_objects

指定した HDL オブジェクト (変数、信号、ワイヤ、またはレジスタ) の詳細を表示します。

構文

```
report_objects [-quiet] [-verbose] [<hdl_objects>...]
```

戻り値

HDL オブジェクトの名前、タイプ、データ型

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hdl_objects>]	レポートする HDL オブジェクトを指定します。デフォルトは report_objects [get_objects *] です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

report_objects コマンドは、指定の HDL オブジェクトのタイプ、名前、言語を Tcl コンソールまたは Tcl シェルにレポートします。このコマンドを使用するには、シミュレーションを開いている必要があります。

このコマンドを実行すると、指定のオブジェクトの簡単な説明が返されます。より詳細な情報を取得するには、describe コマンドを使用します。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event があります。HDL 定数には、Verilog のパラメーターおよび localparam、VHDL ジェネリックおよび定数が含まれます。

このコマンドを実行すると、HDL オブジェクトのタイプ、名前、コード タイプ (Verilog/VHDL) が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hdl_objects> (オプション): レポートするオブジェクトを指定します。デフォルトでは、シミュレーションの現在のスコープ (`current_scope`) にあるすべてのオブジェクトがレポートされます。

注記: オブジェクトは、名前または `get_objects` コマンドで指定します。

例

次の例では、指定のオブジェクトが、現在のシミュレーションのスコープによってどのようにレポートされるかを示しています。

```
current_scope testbench
/testbench
report_objects [get_objects leds_n]
  Declared: {leds_n[3:0]}          Verilog
current_scope dut
/testbench/dut
report_objects leds_n
  Out: {LEDS_n[3:0]}              VHDL
```

次の例では、現在のシミュレーション スコープの指定した HDL オブジェクトをレポートしています。

```
report_objects [get_objects GPIO*]
```

関連項目

- [current_scope](#)
- [describe](#)
- [get_objects](#)

report_operating_conditions

消費電力見積もりの動作条件値を取得します。

構文

```
report_operating_conditions [-voltage <args>] [-grade] [-process]
    [-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
    [-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
    [-design_power_budget] [-all] [-file <arg>] [-return_string] [-append]
    [-quiet] [-verbose]
```

使用法

名前	説明
[-voltage]	電圧値を取得します。サポートされる電圧は、ファミリーによって異なります。
[-grade]	電圧グレードを指定します。サポートされる値は、ファミリーによって異なります。
[-process]	プロセスを取得します。
[-junction_temp]	ジャンクション温度 (C) を取得します。
[-ambient_temp]	周囲温度 (C) を取得します。
[-thetaja]	ThetaJA (C/W) を取得します。
[-thetasa]	ThetaSA を取得します。
[-airflow]	エアフロー (LFM) を取得します。
[-heatsink]	ヒートシンクのサイズを取得します。
[-thetajb]	ThetaJB を取得します。
[-board]	ボード タイプを取得します。
[-board_temp]	ボード温度 (C) を取得します。
[-board_layers]	ボードの層数を取得します。
[-design_power_budget]	デザインの消費電力バジェット (W) をリセットします。
[-all]	このヘルプ メッセージにリストされるすべての動作条件を取得します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-return_string]	動作条件を文字列として返します。
[-append]	動作条件をファイルの最後に追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

デザインのパフォーマンスを解析する際に使用される実際の動作条件を表示します。動作条件は、`set_operating_conditions` コマンドを使用して定義します。

デバイスの環境動作条件は、`report_power` コマンドによる消費電力解析では使用されますが、タイミング解析では使用されません。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-voltage` (オプション): 電圧ペアのリストをレポートします。サポートされる電圧は、ファミリによって異なります。

`-grade` (オプション): ターゲット デバイスの温度グレードをレポートします。

`-process` (オプション): 製造プロセス特性をレポートします。

`-junction_temp` (オプション): モデリングに使用されるデバイスのジャンクション温度をレポートします。

`-ambient_temp` (オプション): 環境周囲温度をレポートします。

`-thetaja` (オプション): モデリングに使用される Theta-JA 熱抵抗をレポートします。

`-thetasa` (オプション): モデリングに使用される Theta-SA 熱抵抗をレポートします。

`-airflow` (オプション): モデリングに使用するエアフローを LFM (リニア フィート/分) でレポートします。

`-heatsink` (オプション): モデリングに使用されるヒートシンクのブタイプをレポートします。

`-thetajb` (オプション): モデリングに使用される Theta-JB 熱抵抗をレポートします。

`-board` (オプション): モデリングに使用されるボードのサイズをレポートします。

`-board_temp` (オプション): モデリングに使用されるボードの温度を摂氏でレポートします。

`-board_layers` (オプション): モデリングに使用されるボードの層数をレポートします。

`-design_power_budget` (オプション): デザインの消費電力バジェットをワットでレポートします。この値は、`report_power` コマンドで計算されたオンチップ消費電力とデザインの消費電力バジェットとの差をレポートするのに使用されます。

`-all` (オプション): 動作条件の現在の値をすべてレポートします。各条件が個別にレポートされるのを避ける場合に使用します。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` が使用されていない場合は、同じ名前の既存のファイルが上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、インダストリアル温度グレード デバイスの周囲温度を 75°C に設定し、それらの設定をディスク上のファイルに書き込んでいます。

```
set_operating_conditions -grade industrial -junction_temp 75
report_operating_conditions -grade -junction_temp -return_string -file \
~/conditions.txt
```

関連項目

- [set_operating_conditions](#)

report_param

すべてのパラメーターに関する情報を取得します。

構文

```
report_param [-file <arg>] [-append] [-non_default] [-return_string]
             [-quiet] [-verbose] [<pattern>]
```

戻り値

パラメーター レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-non_default]	デフォルト以外の値に設定されているパラメーターのみをレポートします。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<pattern>]	検索パターンに一致するパラメーターを取得します。デフォルトは * です。

カテゴリ

[PropertyAndParameter](#) (プロパティおよびパラメーター)、[Report](#) (レポート)

説明

ユーザーが定義可能なパラメーターのリスト、現在の値、パラメーターの設定または制御内容を取得します。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、 `-file` オプションを使用している場合にのみ使用可能です。

`-non_default` (オプション): デフォルト値から変更されているパラメーターのみをレポートします。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pattern>` (オプション): パラメーターを検索するパターンを指定します。デフォルトの検索パターンはワイルドカード (*) で、すべてのユーザー定義可能なパラメーターが取得されます。

例

次の例では、すべてのユーザー定義可能なパラメーターの名前、値、説明が返されます。

```
report_param
```

次の例では、指定した検索パターンに一致するユーザー定義可能なパラメーターの名前、値、および説明が返されます。

```
report_param *coll*
```

関連項目

- [get_param](#)
- [list_param](#)
- [reset_param](#)
- [set_param](#)

report_phys_opt

物理合成変換の詳細をレポートします。

構文

```
report_phys_opt [-file <arg>] [-append] [-return_string] [-quiet]  
                [-verbose]
```

使用法

名前	説明
[-file]	出力ファイルを指定します。
[-append]	結果をファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

`phys_opt_design` コマンドで実行されたファンアウト ドライバー複製およびロード分配最適化の結果をレポートします。

引数

`-file <arg>` (オプション): 物理最適化レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): レポートの出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`phys_opt_design` コマンドで現在のデザインに対して実行された物理最適化をレポートしています。

```
report_phys_opt -file C:/Data/physOpt_Report.txt
```

関連項目

- [phys_opt_design](#)

report_pipeline_analysis

パイプライン レジスタ挿入解析を実行し、レポートを表示します。

構文

```
report_pipeline_analysis [-cells <args>] [-verbose] [-clocks <args>]  
    [-file <arg>] [-include_paths_to_pipeline] [-append]  
    [-max_added_latency <arg>] [-report_loops] [-return_string] [-quiet]
```

使用法

名前	説明
[-cells]	指定した各階層セルを個別に解析し、セルの外部のフィードバックループを無視します。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[-clocks]	指定したクロックのみのレポートを表示します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-include_paths_to_pipeline]	パイプラインが推奨されるパスをレポートします。-file を使用した場合にのみ使用可能です。
[-append]	既存のファイルの最後に追加します。
[-max_added_latency]	システムに追加可能な最大追加レイテンシを指定します。0 に設定した場合は無制限になります。デフォルトは 100 です。
[-report_loops]	ループ情報もレポートします。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。

カテゴリ

[Tools \(ツール\)](#)

説明

合成済みデザインを解析し、デザインにパイプライン段を挿入した場合に各クロック ドメインでの周波数 (Fmax) がどれだけ向上するかをレポートします。パイプライン処理では向上しないデザインのループを検索してそれらのループがデザインのクリティカル パスであるかどうかを判断する解析も実行されます。

このコマンドを実行すると、パイプライン段と Fmax の向上を示す表が返されます。レポートにはまず元のデザインが示され、Fmax が向上しなくなるまでレイテンシ段 (1、2、...) が追加されます。このコマンドは、デザインの周波数パフォーマンスの理論的な上限をレポートします。

この解析は通常、論理ネットリスト構造によりパフォーマンスが決定する配置前の合成済みネットリストに対して実行されます。このコマンドは、最上位デザインまたは独立階層 (OOC) サブモジュールに対して実行できます。このレポートにより、デザイン周波数を向上させることはできるか、Fmax を向上するにはデザインに何段のパイプラインレジスタを追加する必要があるかがわかります。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、ファイルに書き込むこともできます。

引数

`-cells <args>` (オプション): 指定した階層セルのパイプライン解析を実行します。複数のセルを指定すると、各セルに対してパイプライン解析レポートが生成されます。セルは、名前で指定するか、または `get_cells` コマンドを使用してオブジェクトとして指定できます。

`-verbose <arg>` (オプション): レポートに含める詳細レベルを指定します。0 より大きい整数値を指定できます。

`-clocks <args>` (オプション): 解析するクロック ドメインを指定します。指定しない場合、すべてのクロックのタイミング パスが解析されます。このオプションを使用すると、指定のクロック ドメインに関連するパス グループのみを解析できます。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-include_paths_to_pipeline` (オプション): 現在のデザインでパイプラインが推奨されるパスをレポートします。



ヒント: このオプションを使用する場合は、`-file` オプションも使用する必要があります。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-max_added_latency <arg>` (オプション): パイプラインの挿入により追加する遅延の追加レベルを指定します。パイプライン解析で考慮するパイプラインの最大段数を表す 1 ~ 100 の整数で指定します。デフォルトは 0 で、デザイン パフォーマンスが向上しなくなるまでパイプライン遅延が追加されます。このオプションで指定した段数まで、または Fmax が向上しなくなるまで解析が実行されます。

`-report_loops` (オプション): シーケンシャル フィードバック ループ内で最低速のパスをレポートします。これらは始点と終点と同じシーケンシャル セルであるパスで、フィードバック パスには 0、1、または複数のシーケンシャル セルが含まれます。シーケンシャル ループはパイプライン処理できません。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

例

次の例では、現在のデザインに含まれるクロックの名前、周期、波形、およびソースが返されます。

```
report_pipeline_analysis -file C:/Data/FPGA_Design/pipeline_report.txt
```

関連項目

- [get_cells](#)
- [get_clocks](#)
- [report_cdc](#)
- [report_design_analysis](#)

report_power

消費電力見積もりを実行し、レポートを表示します。

構文

```
report_power [-no_propagation] [-hier <arg>] [-hierarchical_depth <arg>]
             [-vid] [-advisory] [-file <arg>] [-name <arg>] [-format <arg>]
             [-xpe <arg>] [-l <arg>] [-return_string] [-append] [-rpx <arg>]
             [-quiet] [-verbose]
```

使用法

名前	説明
[-no_propagation]	ネットのスイッチング アクティビティを見積もる伝搬エンジンをディスエーブルにします。
[-hier]	階層レポートの形式を指定します。有効な値は logic、power、all で、デフォルトは power です。
[-hierarchical_depth]	テキスト形式の階層レポートのレベルを指定します。デフォルトは 4 です。
[-vid]	デバイスの電圧 ID (VID) を指定します。
[-advisory]	消費電力アドバイザリ テキスト レポートを出力します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-format]	消費電力見積もりレポートのフォーマットを指定します。有効な値は text および xml で、デフォルトは text です。
[-xpe]	XPE にインポートできるように結果を XML ファイルに出力します。
[-l]	詳細レポートに含める最大行数を指定します。デフォルトは 10 です。
[-return_string]	レポートを文字列として返します。
[-append]	消費電力レポートをファイルの最後に追加します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Power (電力)

説明

現在のデザインで消費電力解析を実行し、デバイスの現在の動作条件とデザインのスイッチング レートに基づく消費電力の詳細をレポートします。動作条件は、set_operating_conditions コマンドを使用して設定できます。スイッチング アクティビティを定義するには、set_switching_activity コマンドを使用します。

スイッチング アクティビティは、`read_saif` コマンドを使用して SAIF ファイルから読み込むこともできます。SAIF ファイルからのアクティビティがデザイン ノードにアノテートされ、消費電力が正しく見積もられます。

このコマンドを実行するには、合成済みデザインまたはインプリメント済みデザインを開いている必要があります。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-no_propagation` (オプション): 未定義のノードに対してアクティビティを見積もるのにベクターなし伝搬エンジンを使用します。このオプションを使用すると、デザインを高速に解析する伝搬エンジンがディスエーブルになります。

`-hier [power | logic | all]` (オプション): デザインの各階層のサマリ消費電力 (`power`)、階層の異なるロジック エlement ごとの消費電力 (`logic`)、またはデザイン階層のサマリ消費電力とロジック エlement ごとの消費電力 (`all`) を表示します。デフォルトは `power` です。

`-hierarchical_depth <arg>` (オプション): 消費電力レポートの階層ごとのセクションにレポートする階層レベル数を指定します。デフォルトのレベル数は 2 で、最上位とそのすぐ下にある最上位の子インスタンスのみがレポートされます。0 に設定すると、すべての階層レベルがレポートされます。このオプションはテキスト形式の消費電力レポートにのみ適用され、GUI または RPX レポートには適用されません。

`-vid` (オプション): ターゲット デバイスの電圧 ID ビットを使用します。電圧 ID は適応型電圧制御 (AVS) の一種で、Virtex®-7 ファミリの一部のデバイスを低電圧 0.9V で動作させながら 1.0V の公称電源で動作させた場合と同じパフォーマンスを達成します。電圧 ID 対応のデバイスでは、ワースト ケース (最大) のスタティック消費電力が約 30% 低減されており、放熱量も大幅に少なくなっています。

`-advisory` (オプション): 消費電力レポートに、デザインに制御信号の異常なスイッチング アクティビティがあるかをチェックするアドバイザリ表を追加します。これは、Vivado IDE の消費電力制約アドバイザ機能で生成される表と同じです。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): 結果の名前を指定します。

`-format [text | xml]` (オプション): 出力レポートのデフォルト フォーマットは `text` です。XML レポートを出力することも可能です。XML 出力は `-file` オプションを指定した場合にのみ有効で、`-append` オプションと同時に使用することはできません。

`-xpe <arg>` (オプション): Xilinx® Power Estimator (XPE) スプレッドシートにインポートできるように結果を XML ファイルに出力します。詳細は、『Xilinx Power Estimator ユーザー ガイド』(UG440) を参照してください。

`-l <arg>` (オプション): 詳細レポート セクションの最大行数を指定します。0 以上の値を指定します。

注記: `-l` オプションを指定すると、指定しないよりも詳細な情報が含まれます。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ネット伝搬なしで消費電力解析を実行し、XPE で使用できるように結果を XML ファイルに出力します。

```
report_power -no_propagation -xpe C:/Data/design1.xpe
```

関連項目

- [read_saif](#)
- [set_switching_activity](#)
- [set_operating_conditions](#)

report_power_opt

消費電力の最適化をレポートします。

構文

```
report_power_opt [-cell <args>] [-file <arg>] [-format <arg>] [-name <arg>]  
[-append] [-return_string] [-quiet] [-verbose]
```

使用法

名前	説明
[-cell]	インスタンス名を指定します。デフォルトは空です。
[-file]	出力ファイルを指定します。
[-format]	レポートのフォーマットを指定します。有効な値は text および xml です。-file を使用した場合にのみ適用されます。xml を指定した場合、-append は使用できません。デフォルトは text です。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-append]	既存のファイルの最後に追加します。このオプションを指定しない場合、既存のファイルは上書きされます。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Power (電力)

説明

power_opt_design コマンドを使用してデザインに実行した消費電力最適化をレポートします。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

-cell <args> (オプション): 指定したセルまたはセル インスタンスの消費電力最適化をレポートします。デフォルトでは、デザイン全体の消費電力最適化がレポートされます。

-file <arg> (オプション): レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-format [text | xml]` (オプション): `-file` オプションを使用している場合に、消費電力最適化レポートのフォーマットを指定します。デフォルトはテキスト フォーマットです。サードパーティ アプリでできるように XML フォーマット ファイルを記述することもできます。

注記: `-format xml` を指定した場合、`-append` オプションはサポートされません。

`-name <arg>` (オプション): GUI モードで実行した場合の消費電力最適化レポートの名前を指定します。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは `-file` オプションを使用している場合にのみ使用可能です。XML フォーマットでは使用できません。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインに対して実行された消費電力最適化をレポートし、結果を指定のファイルに XML フォーマットで記述しています。

```
report_power_opt -format xml -file C:/Data/power_opt.xml
```

関連項目

- [power_opt_design](#)
- [report_power](#)

report_pr_configuration_analysis

複数のコンフィギュレーション間のリコンフィギャラブル パーティション解析をレポートします。

構文

```
report_pr_configuration_analysis [-complexity] [-clocking] [-timing]
    [-cells <args>] [-dcps <args>] [-rent] [-nworst <arg>] [-file <arg>]
    [-quiet] [-verbose]
```

使用法

名前	説明
[-complexity]	複雑性解析を実行します。
[-clocking]	クロッキング解析を実行します。
[-timing]	境界ネット タイミング解析を実行します。
[-cells]	リコンフィギャラブル セルの名前をリストします。
[-dcps]	各リコンフィギャラブル セルのデザイン チェックポイントをリストします。DCP の順序は、-cells オプションのものと一致している必要があります。
[-rent]	複雑性解析の一部として Rent 指数を計算します。大型のデザインでは、実行時間が長くなります。
[-nworst]	レポートするワースト境界パスの数を指定します。デフォルトは 10 です。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明



重要: まず、プロジェクトの PR_FLOW プロパティを TRUE に設定するか、[Tools]→[Enable Partial Reconfiguration] コマンドを使用して、プロジェクトをパースシャル リコンフィギュレーション (PR) プロジェクトとして定義する必要があります。

create_pr_configuration コマンドで定義されている複数のコンフィギュレーション間のリコンフィギャラブル パーティション解析をレポートします。このレポートは、選択した各リコンフィギャラブル モジュールを比較し、PR デザインに関する情報を提供します。リソース使用量、フロアプラン、クロッキング、タイミング メトリクスが解析されるので、PR デザイン全体を管理するのに役立ちます。このコマンドの詳細は、『Vivado Design Suite ユーザーガイド: Dynamic Function eXchange』(UG909)を参照してください。

この解析を実行すると、各 RM が指定したチェックポイントの情報に基づいて解析されます。合成後のチェックポイントも指定できますが、リンクおよび拡張がすべて終了した opt_design の後にならないと完全な情報は入手できません。

引数

`-complexity` (オプション): デザインの複雑性解析を実行し、Rent 指数、平均ファンアウト、およびプリミティブ ヒストグラムをレポートします。



ヒント: `-complexity` オプションは、`-cells` と共に指定してデザインまたはセルの解析を制御できます。

`-clocking` (オプション): 各 RP のクロック使用率およびロードに関する情報をレポートします。デザインの全体的に必要なクロック分配をプランニングするのに役立ちます。

`-timing` (オプション): 境界インターフェースのタイミングの詳細をレポートします。RM の入出力でのボトルネックを解析するのに役立ちます。

`-cells <arg>` (オプション): レポートを生成するリコンフィギャラブル セルのリストを指定します。レポートの詳細は、デザイン全体 (`current_instance`) ではなく、指定したセルに基づきます。

`-dcps <arg>` (オプション): 各リコンフィギャラブル セルのデザイン チェックポイント (DCP) を指定します。



重要: DCP ファイルの順序は、`-cells` オプションのものと一致している必要があります。

`-rent` (オプション): 複雑性レポートに Rent メトリクスを追加します。このオプションを使用すると大型デザインでは時間がかかるので、デフォルトでは複雑性解析に含まれていません。

`-nworst <arg>` (オプション): レポートするワースト境界パスの数を指定します。デフォルトは 10 です。

`-file <arg>` (オプション): 解析結果を [Tcl Console] ウィンドウに表示するのではなく、指定したファイルに保存します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインの複雑性解析を実行しています。

```
report_pr_configuration_analysis -complexity
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)
- [delete_reconfig_modules](#)
- [get_partition_defs](#)

- [get_reconfig_modules](#)
- [set_property](#)

report_property

オブジェクトのプロパティをレポートします。

構文

```
report_property [-all] [-class <arg>] [-return_string] [-file <arg>]
               [-append] [-regexp] [-quiet] [-verbose] [<object>] [<pattern>]
```

戻り値

プロパティ レポート

使用法

名前	説明
[-all]	オブジェクトのプロパティを設定していないものも含めてすべてレポートします。
[-class]	プロパティを取得するオブジェクト タイプを指定します。 <object> を指定する場合は使用できません。
[-return_string]	Tcl インタープリターの result 変数に report_property の実行結果を設定します。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-regexp]	検索パターンを正規表現として処理します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<object>]	プロパティを取得するオブジェクトを指定します。
[<pattern>]	プロパティを検索するパターンを指定します。デフォルトは * です。

カテゴリ

[Object \(オブジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)、[Report \(レポート\)](#)

説明

指定したオブジェクトまたはオブジェクトのクラスのすべてのプロパティに対して、その名前、タイプ、および値を取得します。

注記: `list_property` でもオブジェクトのすべてのプロパティのリストが返されますが、プロパティ タイプと値は含まれません。

`report_property` コマンドのオブジェクトは、`get_*` コマンドを使用してオブジェクトとして指定できます。
`lindex` コマンドを使用して、オブジェクトのリストから特定のオブジェクトを指定できます。

```
report_property [lindex [get_cells] 0]
```

オブジェクトのクラスのプロパティを見つける場合は、オブジェクトを指定する代わりに `-class` オプションを使用します。

このコマンドを実行すると、オブジェクトのプロパティのレポートが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-all` (オプション): プロパティ値が定義されていなくても、オブジェクトのすべてのプロパティを返します。

`-class <arg>` (オプション): 指定したオブジェクトのクラスのプロパティを返します。大文字/小文字が区別されます。ほとんどのクラス名は小文字で記述されます。

注記: `-class` オプションを `<object>` と共に使用することはできません。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-regex` (オプション): 検索パターンが正規表現で記述されていることを指定します。サイリンクスの正規表現では、Tcl コマンドは常に検索文字列の先頭にアンカーされています。検索文字列の先頭または末尾に「`.`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文については、<http://perldoc.perl.org/perlre.html> を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<object>` (オプション): プロパティをレポートするオブジェクトを 1 つ指定します。

注記: 複数のオブジェクトを指定すると、エラーメッセージが表示されます。

`<pattern>` (オプション): 指定した `<object>` のプロパティまたは `-class` オプションで指定したオブジェクト クラスのプロパティを検索するパターンを指定します。プロパティ名が指定の検索パターンに一致するプロパティのみが返されます。デフォルトの検索パターンはワイルドカード (*) で、指定したオブジェクトのプロパティすべてのリストが取得されます。

注記: 大文字/小文字が区別されます。ほとんどのプロパティ名は大文字で記述されます。

例

次の例では、指定したオブジェクトのプロパティすべてが返されます。

```
report_property -all [get_cells cpuEngine]
```

次の例では、指定のクラスオブジェクトのプロパティが返されます。

```
report_property -class bel
```

次の例では、正規表現で指定した検索パターンに一致する現在のハードウェア デバイス (`current_hw_device`) のプロパティが返されます。

```
report_property [current_hw_device] -regexp .*PROG.*
```

ツールでサポートされる異なるデザイン オブジェクトに使用可能なプロパティを取得するには、複数の `report_property` コマンドを連続して使用します。次の例では、指定したオブジェクトのプロパティすべてが返されます。

```
report_property -all [current_project]
report_property -all [current_fileset]
report_property -all [current_design]
report_property -all [current_run]
```

関連項目

- [create_property](#)
- [current_design](#)
- [current_fileset](#)
- [current_hw_device](#)
- [current_project](#)
- [current_run](#)
- [get_cells](#)
- [get_property](#)
- [list_property](#)
- [list_property_value](#)
- [reset_property](#)
- [set_property](#)

report_pulse_width

パルス幅チェックをレポートします。

構文

```
report_pulse_width [-file <arg>] [-append] [-name <arg>] [-return_string]
  [-warn_on_violation] [-all_violators] [-significant_digits <arg>]
  [-limit <arg>] [-min_period] [-max_period] [-low_pulse] [-high_pulse]
  [-max_skew] [-clocks <args>] [-no_header] [-cells <args>] [-rpx <arg>]
  [-quiet] [-verbose] [<objects>]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-name]	出力を保存する結果の名前を指定します。
[-return_string]	レポートを文字列として返します。
[-warn_on_violation]	レポートにタイミング違反が含まれる場合にクリティカル警告を表示します。
[-all_violators]	チェック違反が発生しているピン/ポートのみをレポートします。
[-significant_digits]	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 3 です。
[-limit]	クロックごとにレポートする特定のタイプのチェック数を指定します。デフォルトは 1 です。
[-min_period]	最小周期チェックのみをレポートします。
[-max_period]	最大周期チェックのみをレポートします。
[-low_pulse]	最小 Low パルス幅チェックのみをレポートします。
[-high_pulse]	最小 High パルス幅チェックのみをレポートします。
[-max_skew]	最大スキュー チェックのみをレポートします。
[-clocks]	最小パルス幅/最小周期チェックをレポートするクロックをリストします。
[-no_header]	
[-cells]	report_pulse_width コマンドを指定のセルに対して実行します。
[-rpx]	インタラクティブ結果を出力するファイルの名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	最小パルス幅をチェックするオブジェクトを指定します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明

指定したクロック ネットワーク内でのパルス幅およびフリップフロップに到達したときのパルス幅をレポートします。クロックの立ち上がりエッジの最大遅延と立ち下がりエッジの最小値エンを使用して、High パルス幅のチェックも実行します。クロックの立ち上がりエッジの最小遅延と立ち下がりエッジの最大値エンを使用して、Low パルス幅のチェックを実行します。これにより、立ち上がりエッジと立ち下がりエッジのワースト ケースの遅延が考慮されるため、現在の合成済みデザインまたはインプリメント済みデザインのワースト ケース解析が得られます。最大スキュー (クロック信号間に許容される最大のタイミング差) もレポートされます。

デフォルトでは、レポートに最小パルス幅、最大パルス幅、Low パルス幅、High パルス幅、および最大スキュー チェックが含まれます。特定のチェックを指定すると、同時に指定されていない限りほかのチェックはディスエーブルになります。

デフォルトではレポートは標準出力に出力されますが、その後の処理用にファイルまたは Tcl 文字列変数に出力することもできます。レポートは、`-file`、`-return_string`、または `-name` を指定した場合を除き、デフォルトでは標準出力に返されます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。 `-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): GUI で表示する場合の結果の名前を指定します。GUI のパルス幅レポートは、`delete_timing_results` コマンドを使用して削除できます。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-warn_on_violation` (オプション): レポートに違反が含まれる場合に Vivado Design Suite でクリティカル警告が生成されるよう指定します。

`-all_violators` (オプション): 違反が検出されたオブジェクトのみをレポートします。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 2 です。

`-limit <arg>` (オプション): クロックごとにレポートする特定のタイプのチェック数を指定します。1 以上の値を指定します。デフォルトは 1 です。

`-min_period` (オプション): 最小周期チェックをレポートします。

`-max_period` (オプション): 最大周期チェックをレポートします。

`-low_pulse` (オプション): 最小 Low パルス幅チェックをレポートします。

`-high_pulse` (オプション): 最小 High パルス幅チェックをレポートします。

`-max_skew` (オプション): 2 つのクロック ピン間のスキュー制約をチェックします。

注記: `report_pulse_width` コマンドのデフォルトでは、`min_period`、`max_period`、`low_pulse`、`high_pulse`、および `max_skew` がレポートされます。これらのオプションの 1 つまたは複数指定すると、指定したチェックのみがレポートされます。

`-clocks <arg>` (オプション): パルス幅/周期チェックをレポートするリストします。`-clocks` オプションを指定しない場合、すべてのクロックがチェックされます。

`-no_header` (オプション): レポートにヘッダーを含めません。このオプションは、`-return_string` オプションを使用して結果を文字列として返す場合に特に便利です。

`-cells <arg>` (オプション): 指定した階層セルのレポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): パルス幅をレポートするピン オブジェクトを指定します。`<objects>` を指定しない場合、すべてのピンがチェックされます。

例

次の例では、最小周期および Low パルス幅チェックを実行し、結果を GUI に指定した名前を表示しています。

```
report_pulse_width -min_period -low_pulse -name timing_1
```

関連項目

- [delete_timing_results](#)

report_qor_assessment

実現可能性チェックを実行します。

構文

```
report_qor_assessment [-file <arg>] [-max_paths <arg>] [-append] [-quiet]
                        [-verbose]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-max_paths]	推奨解析で考慮するパスの数を指定します。デフォルトは 100 です。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Feasibility \(設計実現可能性\)](#)、[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

デザインの問題箇所を見つけ出し、デザインが要件を満たす可能性を評価します。このコマンドを実行するには、エラボレート済み、合成済み、またはインプリメント済みデザインを開いている必要があります。

QoR (結果の品質) 評価レポートには、複数のカテゴリが含まれます。

- 設計手法
- 合成
- インプリメンテーション
- デザイン階層
- パーシャル リコンフィギュレーション
- フロアプラン

report_qor_assessment コマンドには report_methodology チェックの一部が含まれ、問題の原因となる不適切な手法が特定されますが、report_qor_assessment では必ずしも不適切ではないが現在のデザインの構造、スタイル、サイズ、または複雑性のために成功率の低いものもチェックされるので、report_methodology よりも包括的です。

QoR 評価レポートの主要な機能は、密集およびパフォーマンス低下の原因となる状況を予測できることです。このコマンドは、合成後のインプリメンテーション フローのどの段階でも実行できます。report_qor_assessment コマンドでは 1 ~ 5 のスコアが返され、1 はエラーが発生することを示し、5 はインプリメンテーションが問題なく実行されタイミングが満たされる可能性が高いことを示します。

QoR 評価レポートで検出された違反に対しては、QoR 推奨項目レポートに問題を回避またはデザインを変更して結果を向上するための推奨項目が示されます。

引数

`-file <arg>` (オプション): QoR 評価レポートを指定のファイルに保存します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-max_paths <arg>` (オプション): 解析するクリティカル パスの数を指定します。デフォルトは、100 個のワースト タイミング パスです。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、10 個のワースト パスを解析した後、推奨項目をレポートします。

```
report_qor_assessment -max_paths 10
```

関連項目

- [report_design_analysis](#)
- [report_qor_suggestions](#)

report_qor_suggestions

QoR の推奨項目を示します。

構文

```
report_qor_suggestions [-file <arg>] [-name <arg>] [-append]
                        [-return_string] [-max_strategies <arg>] [-max_paths <arg>]
                        [-evaluate_pipelining] [-no_split] [-models_dir <arg>] [-cell <args>]
                        [-of_objects <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポートを文字列として返します。
[-max_strategies]	推奨するストラテジの数を指定します。デフォルトは 3 です。
[-max_paths]	推奨解析で考慮するパスの数を指定します。デフォルトは 100 です。
[-evaluate_pipelining]	DSP/BRAM パイプライン XDC ファイルを生成します。
[-no_split]	表の行を分離せずにレポートします。
[-models_dir]	モデルを含むディレクトリへのパスを指定します。デフォルトは /proj/rdi-xco/builds/HEAD/nightly/RUNNING_BUILD/packages/customer/vivado/data/deca/models_dir です。
[-cell]	指定したセルの QoR (結果の品質) をレポートします。
[-of_objects]	QoR 推奨項目オブジェクトのリストを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Feasibility \(設計実現可能性\)](#)、[Report \(レポート\)](#)、[Timing \(タイミング\)](#)、[Object \(オブジェクト\)](#)

説明

QoR (結果の品質) を向上するために推奨されるデザインおよびツール オプションをレポートします。このレポートは、タイミング制約、ネットリスト特性、タイミングが満たされていないパス、および密集情報を確認し、QoR を改善する可能性のある推奨項目を決定します。レポートは、合成後または各イプリメンテーション段階後に生成できますが、デザインを開いておく必要があります。

report_qor_suggestions コマンドは、特定された推奨項目に関する QoR 推奨項目オブジェクトを作成します。これらの推奨項目オブジェクトは、Vivado ツールでイネーブルにして合成およびイプリメンテーションの結果を向上するために使用できます。推奨項目には、自動的に適用できるもの、手動の操作が必要なもの、Tcl デザイン制約を記述する必要があるものがあります。

推奨項目オブジェクトは、`get_qor_suggestions` コマンドを使用して取得できます。QoR オブジェクトには、適用できる段階を定義する `APPLICABLE_FOR` プロパティ、使用をイネーブルにする `ENABLED` プロパティ、自動的に適用可能であることを示す `AUTO` プロパティ、および推奨項目が生成された段階を示す `GENERATED_AT` プロパティがあります。推奨項目をデザインに適用するには、`ENABLED` プロパティが設定されており、`APPLICABLE_FOR` プロパティが合成またはイプリメンテーション段階 `run` に設定されている必要があります。

`write_qor_suggestions` コマンドを使用すると、デザインからの QoR 推奨項目を RQS ファイルに記述できます。デザイン フローを適切な段階にリセットした後、`read_qor_suggestions` コマンドを使用して推奨項目を読み込み、合成またはイプリメンテーション段階を実行してイネーブルの推奨項目を適用します。

QoR 推奨項目を使用するのに推奨される方法は、次のとおりです。

1. `report_qor_suggestions` コマンドを実行し、推奨項目を生成します。
2. `get_qor_suggestions` コマンドを実行して推奨項目を確認し、`ENABLE` プロパティを設定します。
3. `write_qor_suggestions` コマンドを実行して推奨項目を RQS ファイルに記述してディスク上に保存します。
4. デザインを適切な段階にリセットします。
5. `read_qor_suggestions` コマンドを実行してデザインに RQS ファイルを読み込み、推奨項目を復元します。
6. 合成 (`synth_design`) またはイプリメンテーション段階 (`opt_design` など) を実行して推奨項目を適用します。

引数

`-file <arg>` (オプション): QOR レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): ツールを GUI モードで実行している場合に Vivado IDE に表示するビューの名前を指定します。Vivado ツールで一度に開くことのできる QoR 推奨項目レポートは 1 つのみです。これは、ほかのレポートとは異なります。レポート ビューで既に開いているレポートがある場合は、そのレポートが閉じられ、新しいレポートが開きます。結果を保存するには、レポートを再実行する前に `write_qor_suggestions` コマンドを使用してください。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-max_paths <arg>` (オプション): 解析するクリティカル パスの数を指定します。デフォルトは、100 個のワースト タイミング パスです。

`-no_split` (オプション): レポートを生成するときに表の行を分割しません。

`-cell <arg>` (オプション): 指定した階層セルの QoR 推奨項目をレポートします。

`-of_objects <args>` (オプション): `get_qor_suggestions` コマンドを使用して返された QoR オブジェクトのリストで定義される既存の推奨項目をレポートします。`-of_objects` オプションを指定すると、デザインを解析して新しい推奨項目が特定されるわけではなく、指定した既存の推奨項目オブジェクトがレポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、10 個のワースト パスを解析した後、推奨項目をレポートします。

```
report_qor_suggestions -max_paths 10
```

次の例は、`report_qor_suggestions` コマンドの前回の実行で生成された既存の推奨項目をレポートしています。

```
report_qor_suggestions -of_objects [get_qor_suggestions]
```

関連項目

- [get_qor_suggestions](#)
- [read_qor_suggestions](#)
- [report_qor_assessment](#)
- [write_qor_suggestions](#)

report_ram_utilization

デザインの RAM に関する設定をレポートします。

構文

```
report_ram_utilization [-append] [-file <arg>] [-return_string]
                        [-cells <args>] [-include_path_info] [-detail] [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
<code>[-append]</code>	既存のファイルの最後に追加します。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-cells]</code>	指定したセルに含まれるメモリ アレイに関する情報のみをレポートします。
<code>[-include_path_info]</code>	パスの情報を RAM 入力/出力に追加します。
<code>[-detail]</code>	指定した場合、推論されないプリミティブまたは以前のバージョンの Vivado 合成では推論されていたプリミティブに関する情報がレポートに含まれます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

現在の合成済みデザインまたはインプリメント済みデザインでのターゲット パーツ上の RAM リソース使用率をレポートします。レポートは、`-file` または `-return_string` オプションを指定した場合を除き、標準出力に返されます。

レポートには、次の表が含まれます。

- [Summary]: 使用率を RAM のタイプ別 (URAM、ブロック RAM、および分散 RAM) に示します。各 RAM タイプ内で、さらに個々のプリミティブに分けられます。推論されるプリミティブの詳細も示されます。ほかの表は Vivado 合成の推論中に供給される情報に依存しているので、推論されるプリミティブの割合は重要です。

直接推論されるプリミティブまたは XPM の使用により推論されるプリミティブは、RTL 記述によりグループ化されます。レポートには、配列の RTL 記述がどのようにパーツのプリミティブにマップされるか、消費電力の面から効率的にインプリメントされているかを判断するために使用する各ポートおよび接続情報のパフォーマンスに関する情報が示されます。

異なる合成ツールを使用していたり、デザインでプリミティブのインスタネーションを多く使用している場合は、`-detail` オプションを使用してより詳細な情報を得ることができますが、RAM はより大型の配列にグループ化されず、一部の情報は失われます。[Summary] および [Non-Inferred Memory Primitives] 表のみがレポートされます。

[Summary] 表の情報は、異なるタイプの RAM の使用率に不均衡があるかを判断するために使用できます。`place_design` の実行後は、情報の精度が上がります。[Summary] 表の情報に基づいて、多用されているプリミティブから使用が少ないプリミティブに変更すると、効率が多少落ちたとしても有益な場合があります。

分散 RAM のインプリメンテーションも追加で評価できます。分散 RAM プリミティブによって、ビット数/LUT の効率は異なります。たとえば、RAM32M16 は 8 つの LUT を使用し、14 データ ビットまで可能です。RAM32X1D プリミティブでは、LUT を組み合わせた場合、8 つの LUT で最大 8 ビットが可能です。

- [Memory Description]: 推論されたメモリでの RTL を示します。深さ、幅、メモリ タイプ、およびタイミング要件が示されます。メモリ配列の合計ビット数順にリストされます。配列名はほかの表でも使用されるので、レポートが大きい場合に検索に使用できます。

注記: 分散 RAM では、プリミティブ ポートの名前にかかわらず、読み出し/書き込みポートはポート A にマップされ、読み出しポートはポート B にマップされます。

- [Memory Utilization]: 各メモリ配列がどのようにプリミティブにマップされているかを示します。マップが効率的であるか、マップが深さまたは幅で制限されているかも示されます。
- [Memory Performance]: RAM のパフォーマンスの詳細を示します。出力レジスタおよびカスケードの使用に関する情報も含まれます。`-include_path_info` オプションを指定すると、各ピンからのワースト パス情報も示されます。
- [Memory Power]: 使用されている RAM の消費電力の詳細を示します。カスケードに関する情報およびイネーブル ピンが電源または信号のどちらに接続されているかも示します。
- [Non-Inferred Memory Primitives]: `-details` オプションを指定するとレポートされます。推論されていない各プリミティブの情報も示されます。レポートのこの部分のサイズのため、分散 RAM は含まれません。

このコマンドを実行すると、レポートが Tcl コンソールに返されるか、ファイルに記述されるか、文字列として返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。同じ名前のファイルがある場合、警告なしに上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-cells <arg>` (オプション): 現在のデザインの 1 つまたは複数の階層セルで使用される RAM リソースをレポートします。セルは、名前で指定するのではなく、`get_cells` オプションを使用してオブジェクトとして指定する必要があります。

`-include_path_info` (オプション): レポートのメモリ パフォーマンスの表のピンからワースト ケース タイミングパスの簡略された情報を含めます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、推論された RAM リソースと推論されていない RAM リソースのレポートを生成し、指定のファイルに記述します。

```
report_ram_utilization -file C:/Data/ram_util.txt
```

関連項目

- [get_cells](#)
- [report_utilization](#)

report_route_status

配線のステータスをレポートします。

構文

```
report_route_status [-return_string] [-file <arg>] [-append]
  [-of_objects <args>] [-route_type <arg>] [-list_all_nets] [-show_all]
  [-dump_routes] [-has_routing] [-boolean_check <arg>] [-ignore_cache]
  [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-return_string]</code>	Tcl インタープリターの <code>result</code> 変数にレポートの実行結果を設定します。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-append]</code>	結果をファイルの最後に追加します。上書きはしません。
<code>[-of_objects]</code>	指定した配線オブジェクトの詳細な配線情報をレポートします。
<code>[-route_type]</code>	指定したステータスの配線のみをレポートします。有効な値は UNPLACED、NOLOADS、NODRIVER、UNROUTED、ANTENNAS、CONFLICTS、PARTIAL、INTRASITE、HIERPORT、ROUTED です。-of_objects を使用した場合は無視されます。
<code>[-list_all_nets]</code>	デザインの各ネットに対して完全な配線情報をレポートします。-of_objects を使用した場合は無視されます。
<code>[-show_all]</code>	UNPLACED または PARTIAL 配線とマークされた配線に関連するすべてのピン、ANTENNAS または CONFLICTS 配線とマークされた配線に関連するすべてのノードをリストします。デフォルトでは、1 つの配線に対して最初の 15 個のピンまたはノードがレポートされます。
<code>[-dump_routes]</code>	デザインの各配線済みネットに対して完全な配線ツリーをレポートします。これは非常に詳細なレポートです。
<code>[-has_routing]</code>	このデザインに保存されている配線がない場合は 0、ある場合は 1 を返します。その他のオプションはすべて無視されます。
<code>[-boolean_check]</code>	指定したフラグが真の場合は 1、偽の場合は 0 を返します。有効なフラグは、PLACED_FULLY、PARTIALLY_ROUTED、ROUTED_FULLY、ERRORS_IN_ROUTES です。その他すべてのオプションは無視されます。-has_routing オプションと共に使用することはできません。
<code>[-ignore_cache]</code>	キャッシュされている情報を破棄し、デザイン全体の配線ステータスを算出し直します。これは非常に低速です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在のデザインの配線ステートをレポートします。

単にデザインに配線があるかどうかを示す 1 から、デザインの各ネットの完全な配線ツリーまで、さまざまな情報を含むステータス レポートを生成できます。

引数

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-of_objects <args>` (オプション): 指定した配線、ネット、または `xdef_net` オブジェクトの完全な配線ツリーをレポートします。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-route_type <arg>` (オプション): 指定した配線ステータスの配線のみをレポートします。有効な配線ステータスは、次のとおりです。

- ANTENNAS: ネットにアンテナまたはアイランドが含まれる。アンテナは有効なロードに到達しない接続された配線、アイランドは配線ツリーの切断された配線です。
- CONFLICTS: ネットに競合がある (複数の配線が同じノードを共有する、サイトの内部で論理ネットがサイト ピンを駆動しているがそのサイト ピンが異なる論理ネットから配線されている、または配線済みサイトが無効であるとレポートされた)。
- HIERPORT: ネットにロードまたはドライバーがなく、最上位終端ポートに接続されている。
- INTRASITE: サイト内で完全に配線されていて、配線は不要。
- NODRIVER: ネットにドライバーがない。
- NOLOADS: 配線にロードがなく、配線は不要。
- PARTIAL: ネットに配線はあるが、完全に配線されていない。
- ROUTED: ネットがエラーなしで完全に配置配線されている。
- UNPLACED: 配置されていないピンがある。
- UNROUTED: ネットに配線がない。

注記: `-of_objects` オプションを使用した場合は、`-route_type` オプションは無視されます。

`-list_all_nets` (オプション): デザインに含まれるすべてのネットの配線ステータス サマリをレポートします。

注記: `-of_objects` オプションを使用した場合は無視されます。

`-show_all` (オプション): UNPLACED または PARTIAL 配線とマークされた配線に関連するすべてのピン、ANTENNAS または CONFLICTS 配線とマークされた配線に関連するすべてのノードをリストします。デフォルトでは、1 つの配線に対して最初の 15 個のピンまたはノードがレポートされます。

`-dump_routes` (オプション): デザインの各配線済みネットに対して完全な配線ツリーをレポートします。

注記: このオプションを使用するとレポートが長くなり、生成するのに時間がかかる場合があります。

`-has_routing` (オプション): デザインが配線されていない場合は 0、デザインに配線が含まれている場合は 1 を返します。`-has_routing` を指定した場合、その他のオプションはすべて無視されます。

注記: 1 が返された場合でも、デザインが完全に配線されているとは限りません。

`-boolean_check <arg>` (オプション): 指定したデザイン属性が真の場合は 1、偽の場合は 0 を返します。指定可能なデザイン属性は、PLACED_FULLY、PARTIALLY_ROUTED、ROUTED_FULLY、ERRORS_IN_ROUTES などです。

注記: `-has_routing` オプションと共に使用することはできません。

`-ignore_cache` (オプション): デフォルトでは、`report_route_status` コマンドは反復的であり、デザインがインプリメントされるにつれ、新しいネットおよび配線の配線情報のみがアップデートされます。このオプションを指定すると、キャッシュに含まれる情報が無視され、デザイン全体のレポートが再生成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定のネットの配線ステータスをレポートしています。

```
report_route_status -of_objects [get_nets u4*]
```

関連項目

- [route_design](#)

report_scopes

指定したスコープまたは現在のスコープの子スコープ (宣言領域) の名前を表示します。

構文

```
report_scopes [-quiet] [-verbose] [<hdl_scopes>...]
```

戻り値

HDL スコープのプロパティのサブセット

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hdl_scopes>]</code>	レポートする HDL オブジェクトを指定します。デフォルトは <code>report_scopes [get_scopes *]</code> です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

シミュレーションの現在のスコープまたは指定のスコープにある HDL スコープのタイプをレポートします。

HDL スコープは、HDL ファイルのオブジェクトのが宣言されている部分です。Verilog および VHDL の HDL スコープの例は、次のとおりです。

- Verilog スコープ: モジュール、関数、タスク、プロセス、その他の begin-end ブロック
- VHDL スコープ: エンティティ / アーキテクチャ ペア、ブロック、関数、プロシージャ、プロセス

このコマンドを使用するには、シミュレーションを開いている必要があります。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hdl_scopes>` (オプション): レポートする HDL スコープを指定します。デフォルトは現在のスコープです。

例

次の例では、/tb/UUT の子スコープをレポートしています。

```
report_scopes [get_scopes /tb/UUT/* filter {type==module}]
```

次の例では、現在のスコープの子スコープをレポートしています。

```
report_scopes
    VHDL Instance: {U_DEBOUNCE_0}
    VHDL Instance: {U_DEBOUNCE_1}
    VHDL Instance: {U_SINEGEN}
    VHDL Instance: {U_FSM}
    VHDL Process: {line__138}
    VHDL Process: {line__184}
    VHDL Process: {line__185}
    VHDL Process: {line__186}
    VHDL Process: {line__187}
    VHDL Process: {line__191}
```

関連項目

- [current_scope](#)
- [get_scopes](#)

report_sim_device

ターゲット パーツのセル タイプに対して正しい SIM_DEVICE 属性の値をレポートします。

構文

```
report_sim_device [-part <arg>] [-file <arg>] [-append] [-return_string]  
                  [-quiet] [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-part]	パーツを指定します。
[-file]	出力ファイルを指定します。
[-append]	結果をファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

report_simlib_info

シミュレーション ライブラリの情報をレポートします。

構文

```
report_simlib_info [-file <arg>] [-append] [-quiet] [-verbose] <path>
```

使用法

名前	説明
[-file]	出力ファイル名を指定します。デフォルトは report_simlib_info.log です。
[-append]	情報をファイルの最後に追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<path>	コンパイル済みライブラリへのパスを指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

compile_simlib コマンドでコンパイルされたライブラリに関する情報をレポートします。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。-append オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: -append オプションは、-file オプションを使用している場合にのみ使用可能です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<path> (必須): コンパイルされたシミュレーション ライブラリへのパスを指定します。

例

次の例では、指定したパスにあるコンパイル済みシミュレーション ライブラリに関する情報をレポートしています。

```
report_simlib_info C:/Data/compiled_simlib
```

関連項目

- [compile_simlib](#)

report_ssn

現在のパッケージおよびピン配置で SSN 解析を実行します。

構文

```
report_ssn [-name <arg>] [-return_string] [-format <arg>] [-file <arg>]
           [-append] [-phase] [-quiet] [-verbose]
```

戻り値

SSN レポート

使用法

名前	説明
[-name]	結果を出力する GUI パネルの名前を指定します。
[-return_string]	レポートを文字列として返します。
[-format]	レポート フォーマットを指定します。有効な値は CSV、HTML、TXT で、デフォルトは csv です。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	レポートを指定したファイルの最後に追加します。
[-phase]	解析で複数クロック位相を考慮します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在のデザインの同時スイッチ ノイズ (SSN) 解析を実行します。SSN 解析は、出力の切り替わりがインターフェイスのノイズ マージンに与える影響を予測するのに適した方法です。算出および見積もりは、デザインで発生する可能性のあるノイズに関連した問題を特定するためのさまざまな変数に基づいており、最終的なデザインのサインオフ条件としては使用しないでください。

SSN 解析では、同時に切り替わる出力が I/O バンク内のほかの出力ポートに与える影響を見積もります。SSN 予測には、I/O バンク特定の電気特性も組み込まれており、SSN でのパッケージの影響も正確にモデリングされます。

report_ssn コマンドは、set_operating_condition の -grade オプションで指定された選択デバイスの温度グレードに影響される可能性があります。ノイズ解析を実行する前に温度グレードを設定しておく、と、コマーシャル、拡張、Q グレード、または防衛グレード デバイスのおける信号のノイズを確認できます。

デフォルトでは、`report_ssn` コマンドですべてのポートが非同期にトグルすると想定されます。そのため、ワーストケースのノイズ解析となり、不必要に悪い見積もりとなる可能性があります。`-phase` オプションを使用すると、デザインのクロック情報が考慮され、より正確な SSN ノイズがレポートされます。クロックは、`create_clock` および `create_generated_clock` コマンドを使用して定義されている必要があります。生成クロックの周期、位相シフト、デューティ サイクルは、SSN 解析に大きく影響します。

`report_ssn` コマンドは、ザイリンクス UltraScale アーキテクチャ、Spartan-6、Virtex-6、Virtex-7、Kintex-7、および Artix-7 デバイスの詳細な SSN 解析をレポートします。レポートは、`-file`、`-return_string`、または `-name` オプションを指定した場合を除き、標準出力に返されます。



ヒント: `report_ssn` コマンドがサポートされないパーツもあります。SSN 解析をサポートしないターゲット パーツに対して `report_ssn` コマンドを実行すると、エラーが返されます。パーツの `SSN_REPORT` プロパティをクエリすると、このコマンドがサポートされるかどうかわかります。詳細は、例を参照してください。

引数

`-name <arg>` (オプション): GUI に出力する結果の名前を指定します。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-format [CSV | HTML | TXT]` (オプション): 出力のフォーマットを CSV、HTML、または ASCII (TXT) に指定します。デフォルトは CSV です。

注記: フォーマットは `-file` を指定した場合にのみ適用され、それ以外は無視されます。

`-file <arg>` (オプション): SSN レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append`: コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-phase` (オプション): SSN 解析でより正確な結果を得るため、クロック スイッチサイクルを考慮します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインに対して SSN 解析を実行し、その出力を指定したファイルに HTML 形式で保存しています。

```
report_ssn -format html -file C:/Data/devSSN.html
```

次の例では、現在のデザインに対して SSN 解析を位相解析を含めて実行し、その出力を文字列として指定の変数に保存しています。

```
set devSSN [report_ssn -phase -format html -return_string]
```

注記: この例では `-file` が指定されていないので、`-format` は無視されます。

次の例では、現在のプロジェクトのパーツをクエリして `report_ssn` コマンドがサポートされるかどうかを確認し、その後このコマンドをサポートする同じパーツ ファミリのパーツのリストを取得しています。

```
get_property SSN_REPORT [get_property PART [current_project]]
get_parts -filter "FAMILY == [get_property FAMILY [get_property PART \
[current_project]]] && SSN_REPORT"
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [get_parts](#)
- [get_property](#)
- [reset_ssn](#)
- [report_property](#)

report_stacks

デザインのサブプログラム内で待機中のプロセスの名前をテキスト形式で表示します。

構文

```
report_stacks [-of_instance <arg>] [-quiet] [-verbose]
```

戻り値

文字列

使用法

名前	説明
<code>[-of_instance]</code>	デフォルトは NULL です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

サブプログラム内で待機中の HDL プロセスの名前をテキスト形式で表示します。

引数

`-of_instance` (オプション): スタックをレポートするインスタンスを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

サンプル デザイン:

```
module top;

    int i;

    function void f(input int in1);
        automatic int a;
        a = in1 + 7;
        $display($time, " in f :: a %d in1 %d ", a, in1);
    endfunction

    task automatic t(input int in2);
        int b;
        b = in2 + 10;
        $display($time, " in t :: in2 %d b %d ", in2, b);
        #5;
        f(b);        // Case C
        $display($time, " Back in t : after wait and f(%d) ", b);
    endtask

    initial begin                                // "/top/Initial18_0"
        $display($time, " in initial 1 ");
        i = 200;
        t(i);        // Case B
        $display($time, " Back in initial 1 after t(%d) ", i);
    end

    initial begin                                // "/top/Initial25_1"
        $display($time, " in initial 2 ");
        #2;
        f(50);        // Case A
        $display($time, " Back in initial 2 after f(50) ");
    end
end
endmodule
```

シミュレーションが関数 f 内で Case A の呼び出しのために停止した場合、2つのプロセス /top/Initial18_0 および /top/Initial25_1 はそれぞれタスク t (Case B の呼び出し) および関数 f (Case A の呼び出し) 内で待機します。

```
1. > report_stacks
Verilog Process: {/top/Initial18_0}
Verilog Process: {/top/Initial25_1}
```

関連項目

- [get_stacks](#)

report_switching_activity

指定したオブジェクトのスイッチング アクティビティを取得します。

構文

```
report_switching_activity [-static_probability] [-signal_rate]
    [-toggle_rate] [-default_static_probability] [-default_toggle_rate]
    [-file <arg>] [-return_string] [-append] [-hier] [-all] [-type <args>]
    [-quiet] [-verbose] [<objects>...]
```

使用法

名前	説明
[-static_probability]	スタティック確率をレポートします。
[-signal_rate]	信号レートをレポートします。
[-toggle_rate]	トグル レートをレポートします。
[-default_static_probability]	デフォルトのスタティック確率をレポートします。
[-default_toggle_rate]	デフォルトのトグル レートをレポートします。
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-return_string]	スイッチング アクティビティを文字列として返します。
[-append]	スイッチング アクティビティをファイルの最後に追加します。
[-hier]	<objects> で指定した階層インスタンスのすべてのレベルにあるネットのスイッチング アクティビティをレポートします。
[-all]	デザインに含まれるすべてのネットのスイッチング アクティビティをレポートします。
[-type]	指定のカテゴリのノードを指定します。有効なタイプは、io_output、io_bidir_enable、register、lut_ram、lut、dsp、bram_enable、bram_wr_enable、gt_txdata、gt_rxdata です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	objects

カテゴリ

Report (レポート)

説明

現在の合成済みデザインまたはインプリメント済みデザインに含まれるネット、ポート、ピン、およびセルの異なるスイッチング アクティビティをレポートします。レポートされるスイッチング アクティビティには、ネット、ポート、ピンの単純な信号レートと単純なスタティック確率、およびセルのステートに依存するスタティック確率が含まれます。

レポートされる値は、set_switching_activity コマンドで定義します。

注記: このコマンドを実行すると、指定のオブジェクトまたは現在のデザインのスイッチング アクティビティが返されます。

デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-static_probability` (オプション): レポートの一部としてスタティック確率を返します。

`-signal_rate` (オプション): レポートの一部として信号レートを返します。

`-toggle_rate` (オプション): 同期ロジック エLEMENTの出力の指定したクロック入力に対するスイッチング レートとしてトグル レート (%) をレポートします。

`-default_static_probability` (オプション): 現在のデザインの消費電力解析で使用されるデフォルトのスタティック確率をレポートします。デフォルトのスタティック確率を設定するには、`set_switching_activity` コマンドを使用します。

注記: このオプションを指定すると、デフォルトは現在のデザインに適用されるので、オブジェクトを指定する必要がありません。

`-default_toggle_rate` (オプション): 消費電力解析で現在のデザインのプライマリ入力に使用されるデフォルトのトグル レートをレポートします。デフォルトのトグル レートを指定するには、`set_switching_activity` コマンドを使用します。

注記: このオプションを指定すると、デフォルトは現在のデザインに適用されるので、オブジェクトを指定する必要がありません。

`-file <filename>` (オプション): レポートを指定のパスの指定のファイルに書き込みます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): データを Tcl 変数に代入できるようにテキスト文字列として返します。

`-hier` (オプション): 指定の階層オブジェクト (<objects>) のすべてのレベルにおける信号のスイッチング アクティビティをレポートします。`-hier` を使用しない場合、現在の階層レベルで指定の <objects> のスイッチング アクティビティがリセットされます。

`-all` (オプション): `-type` で指定したすべてのインスタンスに含まれるネットのスイッチング アクティビティをレポートします。`-type` オプションと共に使用する必要があります。

`-type <arg>` (オプション): 指定したタイプのロジック エンティティのスイッチング アクティビティをレポートします。デフォルトでは、現在のデザインの最上位または指定の <objects> に適用されます。`-type` オプションは、現在のデザインの最上位にあるロジック オブジェクトの指定のタイプにコマンドを適用します。`-all` または `-hier` オプションを使用して、コマンドを適用するオブジェクトの範囲を変更できます。有効なロジック タイプは次のとおりです。

- `io_output`: プライマリ出力。
- `io_bidir_enable`: 双方向ポートのイネーブル ピン。

- register: 指定のデザイン/階層のすべてのレジスタ 出力。
- lut: 指定のデザイン/階層のすべての LUT 出力。
- lut_ram: 指定のデザイン/階層のすべての分散 RAM 出力。
- dsp: 指定のデザイン/階層のすべての DSP 出力。
- bram_enable: BRAM のイネーブル ピン (ENARDEN/ENBWREN)。
- bram_wr_enable: BRAM のライト イネーブル (WEA/WEBWE)。
- gt_txdata: すべての GT の出力 TX データ ピン。
- gt_rxdata: すべての GT の出力 RX データ ピン。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<objects> (オプション): スイッチング アクティビティをレポートするポート、ピン、およびネット オブジェクトを指定します。-type オプションを使用してロジック オブジェクトを定義した場合は、セルを指定します。

例

次の例では、すべての出力ポートの信号レートおよびスタティック確率をレポートしています。

```
report_switching_activity -signal_rate -static_probability [all_outputs]
```

次の例では、デザインに含まれるすべての LUT オブジェクトの信号レートとスタティック確率がレポートされます。

```
report_switching_activity -signal_rate -static_probability -type lut -all
```

関連項目

- [power_opt_design](#)
- [report_power](#)
- [reset_switching_activity](#)
- [set_switching_activity](#)

report_synchronizer_mtbf

MTBF (平均故障間隔) を算出し、レポートを表示します。

構文

```
report_synchronizer_mtbf [-file <arg>] [-append] [-return_string]
                        [-warn_if_mtbf_below <arg>] [-quiet] [-no_header] [-report_endpoints]
                        [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-return_string]	レポート出力を文字列として返します。
[-warn_if_mtbf_below]	デフォルトは 1e+12 です。
[-quiet]	コマンド エラーを表示しません。
[-no_header]	ヘッダーなしのレポートを生成します。
[-report_endpoints]	CDC パスの終点をレポートします。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明



推奨: このコマンドは、ザイリンクス UltraScale デバイスのみでサポートされ、7 シリーズ デバイスではサポートされません。

report_synchronizer_mtbf コマンドは、デザインに含まれる各クロック乗せ換え (CDC) のシンクロナイザー チェーンの平均故障間隔 (MTBF) をレポートし、すべてのシンクロナイザーを含む全体的な MTBF を示します。シンクロナイザー レジスタの ASYNC_REG プロパティは TRUE に設定し、レポートで適切にシンクロナイザーとして識別されるようにする必要があります。

非同期クロック乗せ換え (CDC) では、データが非同期にキャプチャされ、回路の異なるロードによって異なる値になる可能性があるため、メタステーブル状態となることがあります。複数のクロック ドメインを含むデザインで、非同期データ転送が発生するものや、内部クロックで外部非同期信号がキャプチャされるものは、シンクロナイザー レジスタを使用して全体的な回路の信頼性を向上できます。シンクロナイザー チェーンは複数のレジスタが順に接続されたもので、最初の段が非同期クロック ドメインからデータ信号をキャプチャします。その後のレジスタ段は、メタステーブル イベントが発生した場合のセトリング時間を追加し、MTBF を増加します。シンクロナイザー レジスタの ASYNC_REG プロパティは TRUE に設定する必要があります。MTBF のレポートに加え、ASYNC_REG プロパティを設定すると、合成、シミュレーション、インプリメンテーションで MTBF を増加するよう最適化が実行されるようになります、シンクロナイザー回路の全体的な動作が向上します。



ヒント: シンクロナイザー チェーン内のレジスタに異なるセット/リセットまたはクロック イネーブルを使用しないようにしてください。

このコマンドを実行すると、MTBF レポートが返されるか、正常に実行されなかった場合はエラーが返されます。CDC の制約が不適切な場合など、MTBF が正しく算出できない場合は、警告メッセージが表示されます。次の状況では、シンクロナイザーの MTBF 値が不明 (UNDEFINED) になります。

- CDC の 1 つまたは両方のクロックが制約されていない。
- シンクロナイザー チェーンのレジスタに関連するタイミング違反がある。
- CDC データに対して 0 トグル レートが検出される。

0 トグル レートの場合、`set_switching_activity` コマンドを使用して CDC ネットのトグル レートを現実的な値に手で置き換えることが必要なことがあります。これには、トグル レートとスタティック確率の割り当てが含まれます。

- トグル レート: CDC データ信号の秒ごとの百万単位の遷移数。
- スタティック確率: CDC データ信号が High に駆動される期間の割合。

例: `resync[0]` という CDC ネットにトグル レート 12.5%、スタティック確率 0.5 を割り当てる場合:

```
set_switching_activity -toggle_rate 12.5 -static_probability 0.5 \
[get_nets resync[0]]
```

レポートには、デザインの各シンクロナイザー チェーンに関する次のデータが含まれます。

- MTBF: CDC シンクロナイザーの平均故障間隔を可変時間単位 (秒から年) で示します。無効な MTBF 値は UNDEFINED (不明) とレポートされます。
- Data Toggle Rate (データ トグル レート): CDC データが切り替わるレート。 `report_switching_activity` コマンドでレポートされるデザインのデフォルトのスイッチング アクティビティに基づいています。Mts (秒ごとの百万単位の遷移数) でレポートされます。このレートは、CDC ネット オブジェクトに対して `set_switching_activity` コマンドを使用することにより変更できます。
- Data Sample Rate (データ サンプリング レート): CDC データがサンプリングされるレート。シンクロナイザー チェーンの周波数と同等であり、MHz でレポートされます。
- Settling Time (セトリング時間): シンクロナイザー レジスタ出力からのタイミング パス上の正のスラックの合計 (ns)。セトリング時間が長くなると MTBF が増加します。
- Sending Domain (送信ドメイン): CDC データのソースのクロック ドメイン。ソース クロックが定義されていない場合は、UNCONSTRAINED とレポートされます。
- Receiving Domain (受信ドメイン): CDC データのデスティネーションのクロック ドメイン。デスティネーション クロックが定義されていない場合は、UNCONSTRAINED とレポートされます。

- **Number of Stages (段数):** シンクロナイザー チェーンの長さ。ASYNC_REG の値が TRUE であるレジスタの数と同じです。MTBF の算出値により、出力レジスタでメタステーブル イベントが発生するかどうか判断されます。たとえば ASYNC_REG プロパティが設定された 2 つのレジスタを含む典型的なシンクロナイザーでは、MTBF の算出値は、最後の ASYNC_REG レジスタの後にある出力レジスタでメタステーブル イベントにより無効な値がキャプチャされる確率を示します。シンクロナイザーが複数の出力レジスタに接続されている場合、すべてのレジスタで同じロジック レベルがキャプチャされることを確実にするため、すべてのパスからの最小スラックが MTBF 算出に使用されます。
- **CDC Net Name (CDC ネット名):** CDC データ (非同期にキャプチャされるデータ) の論理ネット名。

このコマンドを実行すると、MTBF レポートが返されるか、正常に実行されなかった場合はエラーが返されます。

このレポートには、デザインに含まれるすべてのシンクロナイザーの MTBF を使用して、各シンクロナイザーの MTBF 値の逆数の合計を反転して算出された全体的な MTBF も含まれます。シンクロナイザーの個数を N とすると、 $(1 / (1/MTBF_1 + 1/MTBF_2 + \dots + 1/MTBF_N))$ という式で算出されます。

引数

-file <arg> (オプション): レポートを保存するファイルを指定します。同じ名前のファイルがある場合、警告なしに上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: **-append** オプションは、**-file** オプションを使用している場合にのみ使用可能です。

-return_string (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを **-file** オプションと共に使用することはできません。

-warn_if_mtb_f_below <arg> (オプション): 浮動小数点値を指定し、それ未満の場合に警告を表示します。デフォルト値は 1e+12 です。

-no_header (オプション): 標準ヘッダーを含まないレポートを生成します。

-report_endpoints (オプション): CDC パスの終点の総数をレポートします。安全な終点、危険な終点、および不明な終点の合計です。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、**set_msg_config** コマンドで定義できます。

例

次の例では、MTBF レポートを指定のファイルに保存しています。

```
report_synchronizer_mtb_f -file C:/Data/mtbf_report.txt
```

関連項目

- [get_nets](#)
- [report_cdc](#)
- [report_clock_interaction](#)
- [report_clock_networks](#)
- [set_switching_activity](#)

report_timing

タイミング パスをレポートします。

構文

```
report_timing [-from <args>] [-rise_from <args>] [-fall_from <args>]
[-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
[-rise_through <args>] [-fall_through <args>] [-delay_type <arg>]
[-setup] [-hold] [-max_paths <arg>] [-nworst <arg>] [-unique_pins]
[-path_type <arg>] [-input_pins] [-no_header] [-no_reused_label]
[-slack_lesser_than <arg>] [-slack_greater_than <arg>] [-group <args>]
[-sort_by <arg>] [-no_report_unconstrained] [-user_ignored]
[-of_objects <args>] [-significant_digits <arg>] [-column_style <arg>]
[-file <arg>] [-append] [-name <arg>] [-no_pr_attribute]
[-routable_nets] [-return_string] [-warn_on_violation] [-cells <args>]
[-rpx <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-from]	タイミング パスの始点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_from]	ピン、ポート、セル、またはクロックの立ち上がりエッジを始点として指定します。
[-fall_from]	ピン、ポート、セル、またはクロックの立ち下がりエッジを始点として指定します。
[-to]	タイミング パスの終点 (ピン、ポート、セル、またはクロック) を指定します。
[-rise_to]	ピン、ポート、セル、またはクロックの立ち上がりエッジを終点として指定します。
[-fall_to]	ピン、ポート、セル、またはクロックの立ち下がりエッジを終点として指定します。
[-through]	タイミング パスの通過点 (ピン、ポート、セル、またはネット) を指定します。
[-rise_through]	ピン、ポート、セル、またはネットの立ち上がりエッジを通過点として指定します。
[-fall_through]	ピン、ポート、セル、またはネットの立ち下がりエッジを通過点として指定します。
[-delay_type]	パス遅延のタイプを指定します。有効な値は max、min、min_max、max_rise、max_fall、min_rise、min_fall で、デフォルトは max です。
[-setup]	最大遅延タイミング パスをレポートします (-delay_type max と同じ)。
[-hold]	最小遅延タイミング パスをレポートします (-delay_type min と同じ)。
[-max_paths]	スラック順に並べた場合に出力するパスの最大数、またはグループごとに並べた場合のパス グループごとに出力するパスの最大数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
[-nworst]	終点までのワーストパスを N 個までリストします。1 以上の値を指定します。デフォルト値は 1 です。

名前	説明
<code>[-unique_pins]</code>	ピンの各セットに対して、パス グループごとに 1 つのパスを表示します。
<code>[-path_type]</code>	パス レポートのフォーマットを指定します。有効な値は end、summary、short、full、full_clock、full_clock_expanded で、デフォルトは full_clock_expanded です。
<code>[-input_pins]</code>	パスの入力ピンを表示します。
<code>[-no_header]</code>	ヘッダーなしのレポートを生成します。
<code>[-no_reused_label]</code>	レポートのピンに再利用ステータスを示すラベルを表示しません。
<code>[-slack_lesser_than]</code>	この値よりも小さいスラックのパスを表示します。デフォルトは 1e+30 です。
<code>[-slack_greater_than]</code>	この値よりも大きいスラックのパスを表示します。デフォルトは -1e+30 です。
<code>[-group]</code>	指定のグループのパスのみをレポートします。
<code>[-sort_by]</code>	パスの並べ替え順を指定します。有効な値は group、slack で、デフォルトは slack です。
<code>[-no_report_unconstrained]</code>	スラックが無限のパスはレポートしません。
<code>[-user_ignored]</code>	set_false_path または set_clock_groups タイミング制約のためにスラックが無限のパスのみをレポートします。
<code>[-of_objects]</code>	タイミングをレポートするパスを指定します。
<code>[-significant_digits]</code>	有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 3 です。
<code>[-column_style]</code>	パス レポートの列のスタイルを指定します。有効な値は variable_width、anchor_left、fixed_width で、デフォルトは anchor_left です。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-append]</code>	結果をファイルの最後に追加します。上書きはしません。
<code>[-name]</code>	結果を出力する GUI パネルの名前を指定します。
<code>[-no_pr_attribute]</code>	パーシャル リコンフィギュレーション デザインで、ネットリスト リソースがスタティック領域にあるのかリコンフィギュレーション領域にあるのかをレポートしません。
<code>[-routable_nets]</code>	配線可能なネット数をタイミング バスのプロパティとして保存します。
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-warn_on_violation]</code>	レポートにタイミング違反が含まれる場合にクリティカル警告を表示します。
<code>[-cells]</code>	report_timing コマンドを指定のセルに対して実行します。
<code>[-rpx]</code>	インタラクティブ結果を出力するファイルの名前を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明



重要: デザインにタイミング制約が含まれない場合、`report_timing` コマンドはデザイン内の制約が適用されていないパスをレポートします。1 つでもパスにタイミング制約が適用されている場合は、`-from/-to` オプションで制約が適用されていないタイミング パスが指定されている場合を除き、`report_timing` コマンドはデザイン内の制約が適用されているパスのみをレポートします。

現在の合成済みデザインまたはインプリメント済みデザインの指定のタイミング パスに対してタイミング解析を実行します。デフォルトでは、各パス グループのスラックが最大のタイミング パスがレポートされます。`-nworst` または `-max_paths` オプションを使用すると、レポートするパスおよび遅延の数を変更できます。



ヒント: `report_timing` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメーターの設定に関する詳細は、`set_param` コマンドを参照してください。

タイミング エンジンにはクワッド タイミング モードで実行され、スロー コーナーとファースト コーナーの最小遅延と最大遅延が解析されます。`config_timing_corners` コマンドで実行する解析のタイプを設定できますが、変更するとタイミング解析の適用範囲が減少するので、デフォルトを変更することはお勧めしません。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じて、結果を GUI に表示したり、ファイルに書き込んだり、文字列として返すこともできます。

引数

`-from <args>` (オプション): 解析するタイミング パスの始点を指定します。ポート、ピン、またはセルをタイミング パスの始点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが始点として指定されます。

`-rise_from <args>` (オプション): `-from` オプションと同様ですが、指定した始点からの信号の立ち上がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジで駆動されるパスのみが始点として考慮されます。

`-fall_from <args>` (オプション): `-from` オプションと同様ですが、指定した始点からの信号の立ち下がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジで駆動されるパスのみが始点として考慮されます。

`-to <args>` (オプション): 解析するタイミング パスの終点 (デスティネーション オブジェクト) を指定します。ポート、ピン、またはセル オブジェクトを終点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが終点として指定されます。

`-rise_to <args>` (オプション): `-to` オプションと同様ですが、指定した終点に到達する信号の立ち上がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジでキャプチャされるパスのみが終点として考慮されます。

`-fall_to <args>` (オプション): `-to` オプションと同様ですが、指定した終点に到達する信号の立ち下がりエッジのみをタイミング解析で考慮します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジでキャプチャされるパスのみが終点として考慮されます。

`-through <args>` (オプション): 指定したピン、セル インスタンス、またはネットを通過するパスのみをタイミング解析で考慮します。個別の `-through` (または `-rise_through` および `-fall_through`) 点を順次指定し、デザインを通過する特定のパスを定義できます。特定のパスを定義するには、通過点の指定順序が重要です。通過点を複数のオブジェクトで指定することもできます。この場合、指定の通過オブジェクトのいずれかを通過するタイミング パスが考慮されます。

`-rise_through <args>` (オプション): `-through` オプションと同様ですが、指定したオブジェクトで立ち上がるパスのみでタイミング解析を実行します。

`-fall_through <args>` (オプション): `-through` オプションと同様ですが、指定したオブジェクトで立ち下がるパスのみでタイミング解析を実行します。

`-delay_type <arg>` (オプション): タイミング レポートを実行する際に解析に使用する遅延のタイプを指定します。有効な値は `min`、`max`、`min_max`、`max_rise`、`max_fall`、`min_rise`、`min_fall` です。`-delay_type` オプションのデフォルト値は `max` です。

`-setup` (オプション): セットアップ違反がないかどうかをチェックします。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): ホールド違反がないかどうかをチェックします。これは `-delay_type min` を指定するのと同じです。



ヒント: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` を指定するのと同じになります。

`-max_paths <arg>` (オプション): `-sort_by` での指定の応じて、スラック順に並べた場合に出力するパスの最大数、またはグループごとに並べた場合にパス グループごとに出力するパスの最大数を指定します。1 以上の値を指定します。デフォルトでは、`report_timing` コマンドによりワースト タイミング パス 1 つ、またはパス グループごとにワースト パス 1 つがレポートされます。

`-nworst <arg>` (オプション): タイミング レポートに出力する終点ごとのタイミング パスの数を指定します。タイミング レポートには、指定した数 (<N>) のワースト パスが表示されます。1 以上の値を指定します。デフォルト値は 1 です。

`-unique_pins` (オプション): ピンの各セットに対して 1 つのタイミング パスのみを表示します。

`-path_type <arg>` (オプション): タイミング レポートに出力するパス データを指定します。デフォルト フォーマットは `full_clock_expanded` です。有効なパス タイプは、次のとおりです。

- `end`: パスの終点のみをタイミング値と共に表示します。
- `summary`: 始点と終点をスラックと共に表示します。
- `short`: 始点と終点をタイミング値と共に表示します。
- `full`: 始点、通過点、終点を含む完全なタイミング パスを表示します。
- `full_clock`: 完全なタイミング パスに加えて完全なクロック パスを表示します。
- `full_clock_expanded`: `full_clock` で表示されるタイミング パスに加え、マスター クロックと生成クロック間の完全なクロック パスを表示します。これがデフォルト設定です。

`-input_pins` (オプション): タイミング パス レポートに入力ピンを表示します。`-path_type` を `full`、`full_clock`、および `full_clock_expanded` に設定した場合に使用します。

`-no_header` (オプション): レポートにヘッダーを含めません。

`-no_reused_label` (オプション): 既存のデザイン チェックポイント (DCP) ファイルに基づいてインクリメンタル 配置配線を使用するデザインの再利用情報のレポートをディスエーブルにします。配置および配線はどちらも、インクリメンタル インプリメンテーション フローを使用して、デザイン チェックポイント ファイル (DCP) に保存されている結果に基づいて、インクリメンタルに実行できます。インクリメンタル配置および配線の詳細は、`read_checkpoint` コマンドまたは『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。デフォルトでは、インクリメンタル配置配線を使用するデザインでは、指定のインクリメンタル チェックポイントから再利用された物理データに関する情報を示すラベルがピンに表示されます。このオプションを指定すると、次を含む再利用ラベルが削除されます。

- `Routing`: このピンへの配置配線は再利用されています。

- Placement: セル配置は再利用されているが、このピンへの配線は再利用されていません。
- Moved: セル配置およびこのピンへの配線はどちらも再利用されていません。
- New: セル、ネット、またはピンは新規です。インクリメンタル チェックポイントに存在しません。

`-slack_lesser_than <arg>` (オプション): 算出されたスラック値が指定した値より小さいパスのタイミングをレポートします。`-slack_greater_than` と共に使用すると、スラック値の範囲を指定できます。

`-slack_greater_than <arg>` (オプション): 算出されたスラック値が指定した値より大きいパスのタイミングをレポートします。`-slack_lesser_than` と共に使用すると、スラック値の範囲を指定できます。

`-group <args>` (オプション): 指定したパス グループのパスのタイミングをレポートします。現在定義されているパス グループを確認するには、`get_path_groups` コマンドを使用します。



ヒント: 各クロックにパス グループが作成されます。パス グループは、`group_path` コマンドを使用して定義することもできます。`-group` オプションは、同じくタイミング パス オブジェクトを指定する `-of_objects` オプションと共に使用することはできません。

`-sort_by [slack | group]` (オプション): レポートのタイミング パスをスラック値順またはパス グループごととリストします。有効な値は `slack` または `group` です。デフォルトでは、`report_timing` コマンドでワースト タイミング パス 1 つまたは `-nworst` で指定した数のタイミング パスがレポートされますが、`-sort_by group` を使用すると、`report_timing` コマンドで各パス グループに対してワースト タイミング パス 1 つまたは `-nworst` で指定した数のタイミング パスがレポートされます。

`-no_report_unconstrained` (オプション): 制約が適用されていないパスのタイミングはレポートしません。このオプションを指定しない場合、`report_timing` コマンドで制約が適用されていないパス (スラックは無限) が含まれます。

`-user_ignored` (オプション): `set_false_path` または `set_clock_groups` 制約のため通常タイミングで無視されるパスのみをレポートします。

注記: `-user_ignored` と `-no_report_unconstrained` オプションを同時に使用することはできません。また、`-user_ignored` オプションは `-slack_lesser_than` および `-slack_greater_than` オプションと共に使用することもできません。

`-of_objects <args>` (オプション): 指定のタイミング パス オブジェクトのタイミングをレポートします。`get_timing_paths` コマンドと共に使用されます。



ヒント: `-of_objects` オプションは、`-from`、`-to`、`-through` オプション、およびそのバリエーションと共に使用することはできません。`-of_objects` オプションは `DELAY_TYPE` プロパティを含むタイミング パス オブジェクトを定義するので、遅延タイプを定義する `-setup`、`-hold`、および `-delay_type` オプションと共に使用することはできません。`-of_objects` オプションは、同じくタイミング パス オブジェクトのグループを定義する `-group` オプションと共に使用することはできません。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 3 です。

`-column_style [variable_width | anchor_left | fixed_width]` (オプション): タイミング レポート出力のタイミング パス部分のフォーマットを指定します。デフォルト フォーマットは `anchor_left` です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): GUI で表示する場合の結果の名前を指定します。

`-no_pr_attribute <arg>` (オプション): パーシャル リコンフィギュレーション (PR) 属性データの標準レポートをディisableにします。PR デザインでは、セルがリコンフィギュラブル パーティション (:RP#) に含まれるか、デザインのスタティック領域 (:S) に含まれるかを示す文字列が論理パスに追加されます。

`-routable_nets` (オプション): 配線可能なネット数を返されたタイミング パスのプロパティとしてレポートします。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-warn_on_violation` (オプション): タイミング レポートにタイミング違反が含まれる場合に Vivado Design Suite でクリティカル警告が生成されるよう指定します。

`-cells <arg>` (オプション): 指定した階層セルのタイミング レポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインのワースト パス 5 つの入力ピンを含む完全なタイミング パスと、そのタイミング値をレポートしています。

```
report_timing -nworst 5 -path_type full -input_pins
```

次の例では、複数の通過点を使用して特定のパス (state_reg1 を通過) および代替パス (count_3 または count_4 を通過) を指定し、タイミング結果を指定のファイルに記述しています。

```
report_timing -from go -through {state_reg1} \  
  -through { count_3 count_4 } \  
  -to done -path_type summary -file C:/Data/timing1.txt
```

関連項目

- [get_path_groups](#)
- [get_timing_paths](#)
- [group_path](#)
- [place_design](#)
- [report_timing_summary](#)
- [route_design](#)
- [set_clock_groups](#)
- [set_false_path](#)

report_timing_summary

タイミング サマリをレポートします。

構文

```
report_timing_summary [-check_timing_verbose] [-delay_type <arg>]
  [-no_detailed_paths] [-setup] [-hold] [-max_paths <arg>]
  [-nworst <arg>] [-unique_pins] [-path_type <arg>] [-no_reused_label]
  [-input_pins] [-no_pr_attribute] [-routable_nets]
  [-slack_lesser_than <arg>] [-report_unconstrained]
  [-significant_digits <arg>] [-no_header] [-file <arg>] [-append]
  [-name <arg>] [-return_string] [-warn_on_violation] [-datasheet]
  [-cells <args>] [-rpx <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-check_timing_verbose]</code>	発生する可能性のあるタイミング問題をチェックする際に詳細レポートを生成します。
<code>[-delay_type]</code>	パス遅延のタイプを指定します。有効な値は max、min、min_max で、デフォルトは min_max です。
<code>[-no_detailed_paths]</code>	解析された各クロックおよびパス グループのタイミング パスをレポートしません。
<code>[-setup]</code>	最大遅延タイミング パスをレポートします (-delay_type max と同じ)。
<code>[-hold]</code>	最小遅延タイミング パスをレポートします (-delay_type min と同じ)。
<code>[-max_paths]</code>	クロックまたはパス グループごとにレポートするパスの最大数を指定します。1 以上の値を指定します。デフォルト値は 1 です。
<code>[-nworst]</code>	終点までのワーストパスを N 個までリストします。1 以上の値を指定します。デフォルト値は 1 です。
<code>[-unique_pins]</code>	ピンの各セットに対して、パス グループごとに 1 つのパスを表示します。
<code>[-path_type]</code>	パス レポートのフォーマットを指定します。有効な値は end、summary、short、full、full_clock、full_clock_expanded で、デフォルトは full_clock_expanded です。
<code>[-no_reused_label]</code>	レポートのピンに再利用ステータスを示すラベルを表示しません。
<code>[-input_pins]</code>	パスの入力ピンを表示します。
<code>[-no_pr_attribute]</code>	パースャル リコンフィギュレーション デザインで、ネットリストリソースがスタティック領域にあるのかりコンフィギュレーション領域にあるのかをレポートしません。
<code>[-routable_nets]</code>	配線可能なネット数をタイミング パスのプロパティとして保存します。
<code>[-slack_lesser_than]</code>	この値よりも小さいスラックのパスを表示します。デフォルトは 1e+30 です。
<code>[-report_unconstrained]</code>	制約が設定されていないパスおよびユーザーの指定により無視されているパスをレポートします。

名前	説明
<code>[-significant_digits]</code>	有効桁数を指定します。有効な値は 0 ～ 3 で、デフォルト値は 3 です。
<code>[-no_header]</code>	ヘッダーなしのレポートを生成します。
<code>[-file]</code>	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
<code>[-append]</code>	結果をファイルの最後に追加します。上書きはしません。
<code>[-name]</code>	結果を出力する GUI パネルの名前を指定します。
<code>[-return_string]</code>	レポートを文字列として返します。
<code>[-warn_on_violation]</code>	レポートにタイミング違反が含まれる場合にクリティカル警告を表示します。
<code>[-datasheet]</code>	データシート レポートを含めます。
<code>[-cells]</code>	<code>report_timing_summary</code> コマンドを指定のセルに対して実行します。
<code>[-rpx]</code>	インタラクティブ結果を出力するファイルの名前を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)、Timing (タイミング)

説明



ヒント: `report_timing_summary` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメーターの設定に関する詳細は、`set_param` コマンドを参照してください。

デザインでタイミング要件が満たされているかどうかを確認するためのタイミング サマリを生成します。開いている合成済みデザインまたはインプリメント済みデザインに対して実行できます。

タイミング サマリ レポートには、次の情報が含まれます。

- **Timer Settings:** レポートのタイミング情報を生成するのに使用されたタイミング エンジンの設定を示します。
- **Check Timing:** `check_timing` コマンドで生成されるのと同じ情報を含み、発生する可能性のあるタイミング問題を示します。
- **Design Timing Summary:** 最悪の負のスラック (WNS)、負のスラックの合計 (TNS)、最悪のホールド スラック (WHS)、ホールド スラックの合計 (THS)、コンポーネントのスイッチ制限 (CSL) など、デザインのタイミングのサマリを示します。
- **Clock Definitions:** `report_clocks` コマンドで生成されるのと同じ情報を含み、`create_clock`、`create_generated_clock` コマンド、またはツールで自動的に生成されたすべてのクロックを示します。
- **Intra-Clock Table:** ソース クロックとデスティネーション クロックが同じタイミング パスのサマリを示します。
- **Inter-Clock Table:** ソース クロックとデスティネーション クロックが異なるタイミング パスのサマリを示します。
- **Path Group Table:** `group_path` コマンドで作成されたデフォルトのパス グループとユーザー定義のパス グループを示します。
- **Timing Details:** `report_timing` コマンドで生成されるのと同様のタイミング パス (最大遅延と最小遅延の両方) および定義された各クロックのコンポーネント スwitch制限を示します。

- **Data sheet:** `report_datasheet` コマンドを実行した場合と同じ情報を含み、I/O ポートでのデザインのタイミング特性を示します。データシート情報は、`-datasheet` オプションを指定するとサマリ レポートに追加されます。

このコマンドは、`launch_runs` コマンドの一部としてインプリメンテーション中に自動的に実行されます。

注記: デフォルトでは、レポートは Tcl コンソールまたは STD 出力に表示されますが、必要に応じてファイルに書き込んだり、文字列として返すこともできます。

引数

`-check_timing_verbose` (オプション): 詳細なタイミング サマリ レポートを出力します。

`-delay_type <arg>` (オプション): タイミング レポートを実行する際に解析に使用する遅延のタイプを指定します。有効な値は `min`、`max`、`min_max` で、`-delay_type` オプションのデフォルト値は `min_max` です。

`-no_detailed_paths` (オプション): 解析された各クロックおよびパス グループの完全なタイミング パスをレポートしません。

`-setup` (オプション): セットアップ違反がないかどうかをチェックします。これは `-delay_type max` を指定するのと同じです。

`-hold` (オプション): ホールド違反がないかどうかをチェックします。これは `-delay_type min` を指定するのと同じです。

注記: `-setup` と `-hold` の両方を指定すると、`-delay_type min_max` を指定するのと同じになります。

`-max_paths <arg>` (オプション): クロックまたはパス グループごとにレポートするパスの最大数を指定します。1 以上の値を指定します。デフォルトでは、`report_timing_summary` コマンドによりワースト タイミング パス 1 つ、またはパス グループごとにワースト パス 1 つがレポートされます。

`-nworst <arg>` (オプション): タイミング レポートに出力するタイミング パスの数を指定します。タイミング レポートには、指定した数 (<N>) のワースト パスから終点までが表示されます。1 以上の値を指定します。デフォルト値は 1 です。

`-unique_pins` (オプション): ピンの各セットを通過するタイミング パスのみを、パス グループごとに 1 つずつレポートします。

`-path_type <arg>` (オプション): タイミング サマリ レポートに出力するパス データを指定します。デフォルト フォーマットは `full_clock_expanded` です。有効なパス タイプは、次のとおりです。

- `end`: パスの終点のみをタイミング値と共に表示します。
- `summary`: 始点と終点をスラックと共に表示します。
- `short`: 始点と終点をタイミング値と共に表示します。
- `full`: 始点、通過点、終点を含む完全なタイミング パスを表示します。
- `full_clock`: 完全なタイミング パスに加えて完全なクロック パスを表示します。
- `full_clock_expanded`: `full_clock` で表示されるタイミング パスに加え、マスター クロックと生成クロック間の完全なクロック パスを表示します。これがデフォルト設定です。

`-no_reused_label` (オプション): 既存のデザイン チェックポイント (DCP) ファイルに基づいてインクリメンタル 配置配線を使用するデザインの再利用情報のレポートをディスエーブルにします。配置および配線はどちらも、インクリメンタル インプリメンテーション フローを使用して、デザイン チェックポイント ファイル (DCP) に保存されている結果に基づいて、インクリメンタルに実行できます。インクリメンタル配置および配線の詳細は、`read_checkpoint` コマンドまたは『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。デフォルトでは、インクリメンタル配置配線を使用するデザインでは、指定のインクリメンタル チェックポイントから再利用された物理データに関する情報を示すラベルがピンに表示されます。このオプションを指定すると、次を含む再利用ラベルが削除されます。

- Routing: このピンへの配置配線は再利用されています。
- Placement: セル配置は再利用されているが、このピンへの配線は再利用されていません。
- Moved: セル配置およびこのピンへの配線はどちらも再利用されていません。
- New: セル、ネット、またはピンは新規です。インクリメンタル チェックポイントに存在しません。

`-input_pins` (オプション): タイミング パス レポートに入力ピンを表示します。`-path_type` を `full`、`full_clock`、および `full_clock_expanded` に設定した場合に使用します。

`-no_pr_attribute <arg>` (オプション): パーシャル リコンフィギュレーション (PR) デザインで、ネットリスト リソースがスタティック領域にあるのかリコンフィギュレーション領域にあるのかを示すデータをレポートに含めません。

`-routable_nets` (オプション): 配線可能なネット数を返されたタイミング パスのプロパティとしてレポートします。

`-slack_lesser_than <arg>` (オプション): 算出されたスラック値が指定した値より小さいパスのタイミングをレポートします。

`-report_unconstrained` (オプション): 現在のデザインに含まれる制約が適用されていないパスのタイミングをレポートします。デフォルトでは、`report_timing_summary` コマンドのレポートに制約が適用されていないパスのタイミングは含まれません。



ヒント: Vivado IDE では、タイミング レポートに [User Ignored Paths] セクションを含めるには、このオプションを指定する必要があります。

`-significant_digits <arg>` (オプション): 出力結果の有効桁数を指定します。有効な値は 0 ~ 3 で、デフォルト値は 3 です。

`-no_header` (オプション): レポートにヘッダー情報を含めません。このオプションは、その後の処理用にタイミング サマリ レポートを文字列として出力する場合に便利です。

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていなければ、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-name <arg>` (オプション): GUI で表示する場合の結果の名前を指定します。GUI のタイミング サマリ レポートは、`delete_timing_results` コマンドを使用して削除できます。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-warn_on_violation` (オプション): タイミング レポートにタイミング違反が含まれる場合に Vivado Design Suite でクリティカル警告が生成されるよう指定します。

`-datasheet` (オプション): タイミング サマリ レポートにデータシート情報を追加します。

`-cells <arg>` (オプション): 指定した階層セルのタイミング サマリ レポートを生成します。レポートの詳細は、デザイン全体ではなく、指定したセルに基づきます。

`-rpx <arg>` (オプション): ザイリンクス レポート ファイル (RPX) を出力するファイルの名前とパスを指定します。これは、`-file` オプションを使用してファイルにレポート結果を保存するのとは異なります。RPX ファイルはインタラクティブ レポートで、すべてのレポート情報が含まれ、`open_report` コマンドを使用して Vivado Design Suite のメモリに読み込み直すことができます。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名にファイル拡張子 `.rpx` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインのタイミング サマリをレポートしています。

```
report_timing_summary
```

次の例では、現在のデザインの制約が適用されていないパスを含むホールド タイミング サマリを、指定のオプションでレポートしています。

```
report_timing_summary -delay_type min -path_type full_clock_expanded \
    -report_unconstrained -max_paths 2 -nworst 1 -significant_digits 2 \
    -input_pins -name {timing_6}
```

関連項目

- [check_timing](#)
- [create_clock](#)
- [create_generated_clock](#)
- [delete_timing_results](#)
- [get_path_groups](#)
- [get_timing_paths](#)

- [group_path](#)
- [open_report](#)
- [report_clocks](#)
- [report_timing](#)
- [report_datasheet](#)

report_transformed_primitives

UNISIM プリミティブ変換の詳細をレポートします。

構文

```
report_transformed_primitives [-file <arg>] [-append] [-return_string]
                              [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	出力ファイルを指定します。
[-append]	結果をファイルの最後に追加します。
[-return_string]	レポートを文字列として返します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

現在のデザインの変換されたプリミティブをレポートします。

合成済みデザインを開き、メモリに読み込むプロセスの一部として、レガシ ネットリスト プリミティブがサポートされる Unisim プリミティブのサブセットに変換されます。

デフォルトではレポートは標準出力に表示されますが、その後の処理用にファイルまたは Tcl 文字列変数に出力することもできます。

引数

-file <arg> (オプション): 変換されたプリミティブのレポートを指定のファイルに保存します。-append オプションが指定されていない場合は、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

-append (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: -append オプションは、-file オプションを使用している場合にのみ使用可能です。

-return_string (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを -file オプションと共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの変換されたプリミティブをレポートし、その結果を指定した Tcl 変数に格納しています。

```
set transPrim [ report_transformed_primitives -return_string ]
```

report_utilization

デバイス使用率を算出し、レポートを表示します。

構文

```
report_utilization [-file <arg>] [-append] [-pblocks <args>]
  [-evaluate_pblock] [-exclude_child_pblocks] [-exclude_non_assigned]
  [-cells <args>] [-return_string] [-slr] [-packthru] [-name <arg>]
  [-no_primitives] [-omit_locs] [-hierarchical] [-spreadsheet_file <arg>]
  [-spreadsheet_table <arg>] [-spreadsheet_depth <arg>]
  [-hierarchical_depth <arg>] [-hierarchical_percentages] [-quiet]
  [-verbose]
```

戻り値

レポート

使用法

名前	説明
[-file]	結果を保存するファイルの名前を指定します。このオプションを使用しない場合、出力はコンソールに表示されます。
[-append]	結果をファイルの最後に追加します。上書きはしません。
[-pblocks]	指定した Pblock の使用率をレポートします。
[-evaluate_pblock]	指定のセルで要求される使用率をレポートし、指定された Pblock エリアとして供給します。
[-exclude_child_pblocks]	使用率レポートから子 Pblock を除外します。
[-exclude_non_assigned]	使用率レポートから割り当てられていないセルを除外します。
[-cells]	指定したセルの使用率をレポートします。
[-return_string]	レポートを文字列として返します。
[-slr]	SLR のリソース使用率をレポートします。
[-packthru]	パックスルーとしてのみ使用される LUT をレポートします。
[-name]	結果を出力する GUI パネルの名前を指定します。
[-no_primitives]	レポートからプリミティブのセクションを削除します。
[-omit_locs]	レポートから [Loced] 列を削除します。
[-hierarchical]	テキスト形式の階層レポートを生成します。
[-spreadsheet_file]	使用率の表をスプレッドシートとしてエクスポートするファイルを指定します。この機能は GUI モードでのみ使用可能です。
[-spreadsheet_table]	スプレッドシート ファイルとしてエクスポートする使用率の表を指定します。デフォルト値は Hierarchy です。
[-spreadsheet_depth]	スプレッドシートの深さレベルを指定します。デフォルト値は 8 です。
[-hierarchical_depth]	テキスト形式の階層レポートのレベルを指定します。デフォルトは 0 です。
[-hierarchical_percentages]	テキスト形式の階層レポートの使用率をパーセントでレポートします。

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Report (レポート)

説明

現在の合成済みデザインまたはインプリメント済みデザインでのターゲット パーツ上のリソース使用率をレポートします。レポートは、`-file`、`-return_string`、または `-name` オプションを指定した場合を除き、標準出力に返されます。



ヒント: リソース使用率は設計プロセスの初期段階でレポートすることもできますが、合成からインプリメンテーションに進むにつれより正確なレポートが得られます。

このコマンドを実行すると、指定の情報が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-file <arg>` (オプション): レポートを保存するファイルを指定します。`-append` オプションが指定されていない場合、指定したファイルが既に存在する場合は上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-pblocks <arg>` (オプション): デザインの 1 つまたは複数の Pblock で使用されるリソースをレポートします。

注記: `-pblocks` オプションは、`-name` オプションと共に使用することはできません。

`-evaluate_pblock <args>` (オプション): 指定した `-pblocks` および `-cells` に対して what-if 解析を実行します。Pblock に指定したセルが配置された場合の使用率をレポートします。使用率の数値は、使用可能なリソースに対する割り当てられたセルの比率を示します。このオプションを使用するには、`-pblocks` および `-cells` オプションを両方指定する必要があります。

`-exclude_child_pblocks` (オプション): `-pblocks` オプションで指定した Pblock の入れ子の Pblock を除いた使用率をレポートします。入れ子の Pblock に割り当てられているセルが含まれないので、Pblock 領域の使用率は低くなります。

`-exclude_non_assigned` (オプション): `-pblocks` オプションを指定している場合に、使用率レポートから Pblock に割り当てられていないセルを除外します。使用されているが指定した Pblock には割り当てられていないセルが含まれないので、Pblock 領域の使用率は低くなります。

`-cells <arg>` (オプション): 現在のデザインの 1 つまたは複数の階層セルで使用されるリソースをレポートします。

`-return_string` (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-slr` (オプション): SLR を含むデバイスの各 SLR の使用率をレポートします。

`-packthru` (オプション): ルートスルーに使用される LUT をレポートします。これは、使用率レポートに「LUTs used exclusively as route-thrus」とレポートされます。

`-name <arg>` (オプション): GUI に出力する結果の名前を指定します。

`-no_primitives` (オプション): レポートからプリミティブ セクションを削除します。プリミティブ セクションでは、デバイスで使用されているロジック プリミティブの数とタイプがレポートされます。

`-omit_locs` (オプション): レポートから [Fixed] 列を削除します。[Fixed] 列には、ユーザーが手動でデバイスに配置したロジック エLEMENT の数をレポートします。

`-hierarchical` (オプション): デザイン階層ごとにデバイスの使用率をレポートします。

`-hierarchical_depth <arg>` (オプション): 階層ごとに使用率をレポートする際に、レポートする階層レベル数を指定します。デフォルトは 0 で、`-hierarchical` オプションを使用するとデザイン階層全体がレポートされます。

`-hierarchical_percentages` (オプション): 階層レポートの使用率データをパーセントでレポートします。

`-spreadsheet_file <arg>` (オプション): 使用率レポートを指定の XLSX フォーマットのスプレッドシートにエクスポートします。スプレッドシート ファイルにエクスポートする機能は、`-name` オプションも指定してレポートを GUI に生成する場合にのみ使用可能です。



ヒント: Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名はファイル拡張子 `.xlsx` を付けて指定する必要があります。パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたは Vivado ツールを起動したディレクトリにファイルが作成されます。

`-spreadsheet_table <arg>` (オプション): スプレッドシート ファイルにエクスポートする使用率の表を指定します。デフォルトでは、デザイン全体の Hierarchy 表がエクスポートされます。このオプションを使用する場合は、`-spreadsheet_file` オプションも使用する必要があります。表の名前は、ツリー表示で表を選択すると [Report Utilization] ウィンドウに表示されます。次に、表の名前の例を示します。

- "Hierarchy"
- "Slice Logic Distribution"
- "Slice Logic Distribution - LUT as Memory"
- "Slice Logic Distribution - LUT as Memory - LUT as Distributed RAM"
- "Slice Logic Distribution - LUT as Memory - LUT as Shift Register"
- "Slice Logic Distribution - LUT as Logic"
- "Slice Logic - F8 Muxes"
- "Slice Logic - Slice Registers"
- "Slice Logic - Slice Registers - Registers as AND/OR"
- "Memory - Block RAM Tile"
- "Memory - Block RAM Tile - RAMB18"
- "DSP - DSPs"
- "Clocking - BUFGCTRL"
- "Specific Feature - BSCANE2"

- "Primitives"

`-spreadsheet_depth <arg>` (オプション): スプレッドシート ファイルにエクスポートするデザインの最上位からの階層の深さを指定します。デフォルト値は 8 です。このオプションを使用する場合は、`-spreadsheet_file` オプションも使用する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザインのすべての Pblock で使用されるリソース使用率を、指定したファイルに記述しています。

```
report_utilization -pblocks [get_pblocks] -file C:/Data/pblocks_util.txt
```

次の例では、デザイン全体の使用率を指定の名前で GUI に表示し、[Clocking - BUFGCTRL] 表を指定のスプレッドシート ファイルにエクスポートしています。

```
report_utilization -name utilization_1 -spreadsheet_file util_table.xlsx \
-spreadsheet_table "Clocking - BUFGCTRL"
```

次の例では、デザインの各 Pblock で使用されるリソース使用率を、各 Pblock のレポートを指定した 1 つのファイルに追加して記述しています。

```
foreach x [get_pblocks] {
    puts "Reporting Pblock: $x -----"
    report_utilization -append -file C:/Data/pblocks_util.txt -pblocks $x
}
```

関連項目

- [delete_utilization_results](#)

report_values

指定した HDL オブジェクト (変数、信号、ワイヤ、またはレジスタ) の現在のシミュレーション値を表示します。

構文

```
report_values [-radix <arg>] [-quiet] [-verbose] [<hdl_objects>...]
```

戻り値

HDL オブジェクトの名前と値

使用法

名前	説明
[-radix]	HDL オブジェクトの値の表示に使用する基数を指定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hdl_objects>]	レポートする HDL オブジェクトを指定します。デフォルトは report_objects [get_objects *] です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

指定した HDL オブジェクトの現在のシミュレーション時間での値をレポートします。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event などがあります。

HDL 定数には、Verilog のパラメーターおよび localparam、VHDL ジェネリックおよび定数が含まれます。HDL スコープは、Verilog のモジュール、関数、タスク、プロセス、begin-end ブロックなど、HDL コードの宣言部分で定義されます。VHDL スコープには、エンティティ/アーキテクチャ定義、関数、プロシージャ、およびプロセス ブロックが含まれます。

引数

-radix <arg> (オプション): 指定したオブジェクトの値を表示するのに使用する基数を指定します。有効な値は default、dec、bin、oct、hex、unsigned、ascii、および smag です。

注記: dec は、符号付き 10 進数を示します。符号なしデータの場合は、unsigned を指定してください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hdl_objects>` (必須): 値を取得する 1 つまたは複数の HDL オブジェクトを指定します。オブジェクトは、名前または `get_objects` コマンドで指定します。

例

次の例では、現在の時間におけるすべてのオブジェクトの値をレポートしています。

```
report_values [get_objects]
```

次の例では、指定したバスから返された値の基数 `bin`、`dec`、および `unsigned` による違いを示しています。

```
report_values -radix bin /test/bench_VStatus_pad_0_i[7:0]
Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog 10100101
report_values -radix unsigned /test/bench_VStatus_pad_0_i[7:0]
Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog 165
report_values -radix dec /test/bench_VStatus_pad_0_i[7:0]
Declared: {/test/bench_VStatus_pad_0_i[7:0]} Verilog -91
```

関連項目

- [current_time](#)
- [get_objects](#)
- [get_value](#)
- [set_value](#)

report_waivers

DRC/METHODOLOGY/CDC のメッセージ除外のステータスをレポートします。

構文

```
report_waivers [-file <arg>] [-type <arg>] [-write_valid_waivers]
               [-write_ignored_waivers] [-append] [-return_string]
               [-show_msgs_with_no_waivers] [-quiet] [-verbose]
```

使用法

名前	説明
[-file]	除外をレポートするファイルを指定します。
[-type]	除外のタイプを指定します。有効な値は、ALL、DRC、METHODOLOGY、CDC です。
[-write_valid_waivers]	最後に report_drc/methodology/cdc run を実行したときに使用された除外をレポートします。
[-write_ignored_waivers]	最後に report_drc/methodology/cdc run を実行したときに使用されなかった除外をレポートします。
[-append]	レポート結果を -file オプションで指定したファイルの最後に追加します。
[-return_string]	レポートの結果を文字列オブジェクトとして返します。
[-show_msgs_with_no_waivers]	除外が定義されていない report_drc、methodology、cdc メッセージもレポートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Waiver \(除外\)](#)、[Report \(レポート\)](#)、[Object \(オブジェクト\)](#)

説明

DRC、METHODOLOGY、および CDC 違反メッセージをレポートし、現在のデザインに設定されている除外を示します。

また、report_drc、report_methodology、および report_cdc コマンドには、除外された違反またはチェックに対してレポートを実行するオプションがあります。

引数

-file <arg> (オプション): 除外レポートを保存するファイルを指定します。指定したファイルが既に存在する場合は、上書きされます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`-type <arg>` (オプション): レポートする除外のタイプを指定します。有効な値は DRC、METHODOLOGY、および CDC です。`-type` オプションを指定しない場合、すべての除外がレポートされます。

`-write_valid_waivers <arg>` (オプション): `-file` オプションと共に使用し、最後に DRC、設計手法、または CDC レポートを実行したときに使用された有効な除外を記述します。

`-write_ignored_waivers <arg>` (オプション): `-file` オプションと共に使用し、最後に DRC、設計手法、または CDC レポートを実行したときに使用されなかった除外を記述します。

`-append` (オプション): レポートの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-return_string` (オプション): レポート出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

注記: このオプションを `-file` オプションと共に使用することはできません。

`-show_msgs_with_no_waivers` (オプション): デフォルトでは、`report_waivers` コマンドで除外が定義されていない DRC、METHODOLOGY、および CDC 違反のデータは表示されません。このオプションを使用すると、除外が定義されていない違反メッセージもレポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のデザインの除外をすべてレポートしています。

```
report_waivers
```

関連項目

- [create_waiver](#)
- [delete_waivers](#)
- [get_waivers](#)
- [report_cdc](#)
- [report_drc](#)
- [report_methodology](#)
- [write_waivers](#)

reset_drc

DRC レポートを削除します。

構文

```
reset_drc [-name <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-name]	削除する DRC 結果の名前を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

DRC、Report (レポート)

説明

指定の結果セットから DRC 結果を削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-name <arg>` (オプション): 削除する DRC 結果の名前を指定します。名前は、`report_drc` コマンドの `-name` オプションで指定されたものです。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、メモリおよび GUI から指定した結果を削除しています。

```
reset_drc -name DRC1
```

関連項目

- [report_drc](#)

reset_drc_check

1 つまたは複数の DRC チェックをデフォルトにリセットします。

構文

```
reset_drc_check [-quiet] [-verbose] [<checks>...]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<checks>]	リセットするチェックを指定します。

カテゴリ

DRC、Object (オブジェクト)

説明

指定の DRC チェックを Vivado Design Suite のデフォルトにリセットします。DRC チェックが、IS_ENABLED、SEVERITY プロパティも含め、デフォルト設定に戻ります。

DRC チェックの IS_ENABLED プロパティを変更すると、そのチェックが `report_drc` コマンドで直接またはルールデッキの一部として指定されている場合でも、実行されないようディスエーブルにできます。

SEVERITY プロパティは文字列プロパティで、`report_drc` コマンドで違反が検出された場合にその DRC ルールに関連付けられている重要度を変更できます。有効な値は、FATAL、ERROR、"CRITICAL_WARNING"、WARNING、ADVISORY です。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<checks>` (必須): デフォルト設定に戻す DRC ルール チェックを指定します。

例

次の例では、ROAS-1 ルールの IS_ENABLED プロパティを変更し、RFFC-1 ルールの SEVERITY プロパティを変更した後、すべてのチェックをデフォルト設定に戻しています。

```
set_property IS_ENABLED false [get_drc_checks ROAS-1]
set_property SEVERITY "Critical Warning" [get_drc_checks RFFC-1]
reset_drc_check [get_drc_checks]
```

関連項目

- [add_drc_checks](#)
- [get_drc_checks](#)
- [report_drc](#)
- [set_property](#)

reset_hw_axi

ハードウェア AXI コアのステートをリセットします。

構文

```
reset_hw_axi [-quiet] [-verbose] [<hw_axis>...]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_axis>]	ハードウェア AXI オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定の hw_axi オブジェクトまたは現在のデバイスの STATUS プロパティをリセットします。

`reset_hw_axi` コマンドは、現在のデバイス上の hw_axi コアを、AXI トランザクションの動作を開始する既知のステータスにリセットします。次の STATUS プロパティがあります。

- STATUS.AXI_READ_BUSY
- STATUS.AXI_READ_DONE
- STATUS.AXI_WRITE_BUSY
- STATUS.AXI_WRITE_DONE
- STATUS.BRESP: 書き込み応答チャネル応答。書き込み転送の結果を示します。
- STATUS.RRESP: 読み出し応答チャネル応答。読み出し転送の結果を示します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_axis> (必須): リセットする hw_axis オブジェクトを指定します。hw_axis は、`get_hw_axis` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、現在のデバイス上にある hw_axis をリセットし、初期設定に戻しています。

```
reset_hw_axis [get_hw_axis]
```

関連項目

- [delete_hw_axis_txn](#)
- [get_hw_axis](#)
- [get_hw_axis_txns](#)
- [refresh_hw_axis](#)

reset_hw_ila

ハードウェア ILA の制御プロパティをデフォルト値にリセットします。

構文

```
reset_hw_ila [-reset_compare_values <arg>] [-quiet] [-verbose]
[<hw_ilas>...]
```

使用法

名前	説明
<code>[-reset_compare_values]</code>	関連のハードウェア プロブ比較値をリセットします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_ilas>]</code>	ハードウェア ILA オブジェクトを指定します。デフォルトは、現在のハードウェア ILA です。

カテゴリ

Hardware (ハードウェア)

説明

指定の ILA デバッグ コアのトリガーおよびキャプチャ構成プロパティと、コアのデバッグ プロブの TRIGGER_COMPARE_VALUE および CAPTURE_COMPARE_VALUE プロパティをリセットします。

ハードウェア ILA (hw_ila) オブジェクトのプロパティは、run_hw_ila コマンドを使用してハードウェア デバイス (hw_device) 上の ILA コアを設定するための準備として、set_property コマンドを使用して設定します。このコマンドは、指定した hw_ila オブジェクト上のユーザー設定可能なプロパティをデフォルト設定に戻します。これらのプロパティの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

プロパティのデフォルト値は、次のとおりです。

- CONTROL.DATA_DEPTH: hw_ila オブジェクトの MAX_DATA_DEPTH。
- CONTROL.TRIGGER_POSITION 0
- CONTROL.WINDOW_COUNT 1
- CONTROL.TRIGGER_MODE BASIC_ONLY
- CONTROL.TRIGGER_CONDITION AND
- CONTROL.TRIG_OUT_MODE DISABLED
- CONTROL.CAPTURE_MODE ALWAYS
- CONTROL.CAPTURE_CONDITION AND
- TRIGGER_COMPARE_VALUE eq1'bX (ハードウェア プロブ (hw_probe) 上)

- `CAPTURE_COMPARE_VALUE eq1'bX (hw_probe 上)`

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-reset_compare_values [true | false]` (オプション): 指定した `hw_ila` オブジェクトに関連付けられている `hw_probe` の `TRIGGER_COMPARE_VALUE` および `CAPTURE_COMPARE_VALUE` プロパティをリセットします。これはブール引数で、デフォルトは `TRUE` です。`-reset_compare_values false` を使用すると、これらのプロパティはリセットされません。その場合、`hw_ila` はリセットされますが、`hw_probe` では現在設定が保持されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_ilas>` (オプション): リセットする `hw_ila` オブジェクトを 1 つ以上指定します。`hw_ila` は、`get_hw_ilas` または `current_hw_ila` コマンドを使用してオブジェクトとして指定するか、名前で指定できます。`hw_ila` オブジェクトを指定しない場合、現在のハードウェア ILA (`current_hw_ila`) がリセットされます。

例

次の例では、現在のデバイス上にあるすべての `hw_ila` デバッグ コアをリセットしています。

```
reset_hw_ila [get_hw_ilas]
```

関連項目

- [current_hw_ila](#)
- [get_hw_ilas](#)
- [run_hw_ila](#)
- [set_property](#)

reset_hw_vio_activity

指定のハードウェア VIO オブジェクトに関連付けられているハードウェア プロープに対し、ハードウェア VIO の ACTIVITY_VALUE プロパティをリセットします。

構文

```
reset_hw_vio_activity [-quiet] [-verbose] <hw_vios>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_vios></code>	ハードウェア VIO オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定した VIO デバッグ コア オブジェクト上のすべてのハードウェア プロープの ACTIVITY_VALUE プロパティをリセットします。ACTIVITY_VALUE プロパティは、VIO デバッグ コアの入カプロープの遷移を表すために Vivado IDE で使用されます。

VIO 入力プロープから値を読み出すだけでなく、VIO 入力プロープのアクティビティを監視することもできます。ACTIVITY_VALUE プロパティは、Vivado IDE の周期的なアップデートの間に VIO 入力の値がいつ変化したか示すために使用されます。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_vios>` (必須): リセットするハードウェア VIO (`hw_vio`) オブジェクトを 1 つ以上指定します。`hw_vio` は、`get_hw_vios` コマンドを使用してオブジェクトとして指定するか、名前で指定できます。

例

次の例では、VIO デバッグ コアの入力アクティビティ プロパティをリセットしています。

```
reset_hw_vio_activity [get_hw_vios]
```

関連項目

- [commit_hw_vio](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [program_hw_devices](#)
- [refresh_hw_vio](#)
- [reset_hw_vio_outputs](#)
- [set_property](#)

reset_hw_vio_outputs

ハードウェア VIO コアの出力を初期値にリセットします。

構文

```
reset_hw_vio_outputs [-quiet] [-verbose] <hw_vios>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_vios></code>	ハードウェア VIO オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

ハードウェア VIO デバッグ コアの出力を初期値にリセットします。

VIO (Virtual Input/Output) デバッグ コアは、ザイリンクス FPGA にプログラムされている内部信号をリアルタイムで監視および駆動できます。VIO コアは、ハードウェア プローブ (hw_probe) オブジェクトを使用してデバイス上の信号を監視および駆動します。入力プローブは、VIO コアへの入力として信号を監視します。出力プローブは、VIO コアから信号を指定の値に駆動します。

`reset_hw_vio_outputs` コマンドは、指定したハードウェア VIO (hw_vio) デバッグ コアの出力プローブの信号値を初期値に戻します。これはハードウェア デバイス (hw_device) の信号に影響しますが、hw_probe オブジェクトの OUTPUT_VALUE プロパティには影響しません。



ヒント: このコマンドは、hw_probe オブジェクトのプロパティをリセットせずに、hw_vio デバッグ コアの信号を初期値にリセットします。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_vios> (必須): リセットするハードウェア VIO (hw_vio) オブジェクトを 1 つ以上指定します。hw_vio は、`get_hw_vios` コマンドを使用してオブジェクトとして指定するか、名前で指定できます。

例

次の例では、VIO デバッグ コア の出力プローブを、FPGA が最初にコンフィギュレーションされ、起動したときのコアの初期値にリセットしています。

```
reset_hw_vio_outputs [get_hw_vios {hw_vio_1}]
```

関連項目

- [commit_hw_vio](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_probes](#)
- [get_hw_vios](#)
- [program_hw_devices](#)
- [refresh_hw_vio](#)
- [reset_hw_vio_activity](#)
- [set_property](#)

reset_methodology

設計手法レポートを削除します。

構文

```
reset_methodology [-name <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-name]</code>	削除する設計手法レポート結果の名前を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Methodology \(設計手法\)](#)、[Report \(レポート\)](#)

説明

指定の結果セットから設計手法レポートの結果を削除します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-name <arg>` (オプション): 削除する設計手法レポート結果の名前を指定します。名前は、`report_methodology` コマンドの `-name` オプションで指定されたものです。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、メモリおよび GUI から指定した結果を削除しています。

```
reset_methodology -name ultrafast_methodology_3
```

関連項目

- [get_methodology_violations](#)

- [report_methodology](#)
- [reset_methodology_check](#)

reset_methodology_check

1 つまたは複数の設計手法チェックをデフォルトにリセットします。

構文

```
reset_methodology_check [-quiet] [-verbose] [<checks>...]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<checks>]</code>	リセットするチェックを指定します。

カテゴリ

[Methodology \(設計手法\)](#)、[Object \(オブジェクト\)](#)

説明

指定の設計手法チェックを Vivado Design Suite のデフォルトにリセットします。チェックが、IS_ENABLED、SEVERITY プロパティも含め、デフォルト設定に戻ります。

設計手法チェックの IS_ENABLED プロパティを変更すると、そのチェックが `report_methodology` コマンドで指定されている場合でも、実行されないようディスエーブルにできます。

SEVERITY プロパティは列挙型のプロパティで、`report_methodology` コマンドで違反が検出された場合に、その設計手法チェックに関連付けられている重要度を変更できます。有効な値は、FATAL、ERROR、"CRITICAL_WARNING"、WARNING、ADVISORY です。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<checks>` (必須): デフォルト設定に戻す DRC ルール チェックを指定します。

例

次の例では、CHECK-4 設計手法チェックの IS_ENABLED および SEVERITY プロパティを変更し、チェックのプロパティをレポートして変更を確認し、その後設計手法チェックをデフォルト設定にリセットしています。

```
set_property IS_ENABLED false [get_methodology_checks CHECK-4]
set_property SEVERITY Warning [get_methodology_checks CHECK-4]
report_property [get_methodology_checks CHECK-4]
reset_methodology_check [get_methodology_checks CHECK-4]
report_property [get_methodology_checks CHECK-4]
```

関連項目

- [get_methodology_checks](#)
- [report_methodology](#)
- [reset_methodology](#)

reset_msg_config

set_msg_config コマンドで定義されたメッセージの制御ルールをリセットまたは削除します。

構文

```
reset_msg_config [-string <args>] [-id <arg>] [-severity <arg>] [-limit]
                 [-suppress] [-count] [-default_severity] [-regex] [-quiet] [-verbose]
```

使用法

名前	説明
[-string]	指定した文字列に一致するルールのみをリセットします。デフォルトは空です。
[-id]	指定した ID に一致するルールのみをリセットします。
[-severity]	指定した重要度に一致するルールのみをリセットします。
[-limit]	現在のプロジェクトで指定したメッセージの最大表示回数をリセットします。
[-suppress]	現在のプロジェクトで指定のメッセージの非表示設定を解除します。
[-count]	現在のプロジェクトで指定したメッセージのカウントをリセットします。メッセージが -limit 制御のために非表示になるのを、次にメッセージ カウントが指定の最大表示回数を超えるまで回避します。
[-default_severity]	現在のプロジェクトで指定のメッセージの重要度をデフォルト値にリセットします。
[-regex]	-string で指定した値が正規表現であることを示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

set_msg_config コマンドで設定したメッセージの最大表示回数またはメッセージの重要度を Vivado ツールのデフォルト設定に戻したり、メッセージの非表示設定を解除したりします。

1 つの reset_msg_config コマンドで、1 つリセット操作のみを実行できます。1 つの reset_msg_config コマンドで複数の操作を実行しようとすると、エラーが返されます。

reset_msg_config コマンドでリセットするメッセージは、-string、-id、および -severity で指定します。これらのオプションの指定すべてに一致するメッセージのみがリセットされます。

注記: 少なくとも 1 つを使用してコマンドを適用するメッセージを指定する必要があります。そうでないと、エラーが返されます。

メッセージに設定されている現在の制御ルールをレポートするには、get_msg_config コマンドを使用します。

引数

`-string <args>` (オプション): 指定した文字列を含むメッセージのみに選択した操作を適用します。文字列は波かっこ ({}) で囲んで指定します。複数の文字列はスペースで区切って指定します。

```
{{Vivado}} {{Synthesis}}
```

注記: 大文字/小文字が区別されます。

`-id <arg>` (オプション): 指定のメッセージ ID のメッセージをリセットします。メッセージ ID は、すべてのメッセージに含まれます。たとえば、「Common 17-54」、「Netlist 29-28」です。

注記: ワイルドカード (*) を指定すると、すべてのメッセージ ID のメッセージがリセットされます。

`-severity <arg>`: 指定の重要度のメッセージをリセットします。次の 5 つの重要度があります。

- **ERROR:** デザインの結果が使用不可となったり、ユーザーの操作がないと回避できないような問題が発生していることを示すエラー メッセージです。
- **{CRITICAL WARNING}:** 入力や制約に適用されないものがあったり、FPGA ファミリには適していないものがあることを示すクリティカル警告メッセージです。修正することが強く推奨されます。

注記: これは 2 単語の値なので、{} で囲む必要があります。

- **WARNING:** 制約または仕様が指定どおりに適用されず、デザインが最適な結果にならない可能性を示す警告メッセージです。修正するかどうかは、ユーザーが判断します。
- **INFO:** STATUS メッセージと同じですが、重要度とメッセージ ID タグが含まれる点が異なります。INFO メッセージには、必要に応じてアンサー データベースで確認できるようにメッセージ ID が含まれます。
- **STATUS:** デザインの処理に関するプロセスおよびフィードバックのステータスを示すステータス メッセージです。STATUS メッセージには、メッセージ ID は含まれません。

`-limit` (オプション): `-string`、`-id`、または `-severity` オプションの指定に一致するメッセージの最大表示回数をリセットします。

`-suppress` (オプション): `-string`、`-id`、または `-severity` オプションの指定に一致するメッセージの非表示設定を解除します。

`-count` (オプション): `-string`、`-id`、または `-severity` オプションの指定に一致するメッセージのカウントをリセットします。

`-default_severity` (オプション): `-string`、`-id`、または `-severity` オプションの指定に一致するメッセージの重要度をデフォルトにリセットします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したメッセージ ID の重要度をクリティカル警告に変更し、その後デフォルトの重要度に戻しています。

```
set_msg_config -id "Common 17-81" -new_severity "CRITICAL WARNING"
reset_msg_config -id "Common 17-81" -default_severity
```

次の例では、指定したメッセージ ID のメッセージの重要度を変更し、現在のメッセージ制御ルールを取得して、特定のルールをリセットする 2 つのコマンドを示しています。

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
-----
Message control rules currently in effect are:
Rule Name Rule Current
Message Count
1 set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
-----
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```



ヒント: 上記の例でメッセージをリセットするのに必要なのは、いずれか 1 つの `reset_msg_config` コマンドのみです。

関連項目

- [get_msg_config](#)
- [set_msg_config](#)

reset_msg_count

メッセージ カウントをリセットします。

構文

```
reset_msg_count [-quiet] [-verbose] <id>
```

戻り値

新しいメッセージ数

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><id></code>	リセットするメッセージ ID (「Common 17-99」など) を指定します。「reset_msg_count *」と指定すると、すべてのカウンターがリセットされます。

カテゴリ

[Report \(レポート\)](#)

説明

指定したメッセージ ID のメッセージ カウントを 0 にリセットします。これにより、メッセージ カウンターが新しく開始します。最大表示回数に近づいている特定のメッセージのカウントをリセットしたり、またはすべてのメッセージのカウントをリセットしたりできます。

ツールで表示される各メッセージには、アプリケーション サブシステムのコードとメッセージ識別子を含む固有のメッセージ ID が付けられています。次に、メッセージ ID の例を示します。

```
"Common 17-54"  
"Netlist 29-28"  
"Synth 8-3295"
```

特定のメッセージ ID の現在のカウントを取得するには、`get_msg_count` コマンドを使用します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<id> (必須): カウントを 0 にリセットするメッセージの ID を指定します。すべてのメッセージのカウントを 0 にリセットする場合は * を指定します。

例

次の例では、すべてのメッセージのカウントをリセットしています。

```
reset_msg_count *
```

関連項目

- [set_msg_config](#)

reset_operating_conditions

消費電力見積もりの動作条件をツールのデフォルトにリセットします。

構文

```
reset_operating_conditions [-voltage <args>] [-grade] [-process]
    [-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
    [-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
    [-design_power_budget] [-quiet] [-verbose]
```

使用法

名前	説明
[-voltage]	電圧値をリセットします。サポートされる電圧は、ファミリによって異なります。
[-grade]	温度グレードをリセットします。
[-process]	プロセスをリセットします。
[-junction_temp]	ジャンクション温度をリセットします。
[-ambient_temp]	周囲温度をリセットします。
[-thetaja]	ThetaJA をリセットします。
[-thetasa]	ThetaSA をリセットします。
[-airflow]	エアフローをリセットします。
[-heatsink]	ヒートシンクのサイズをリセットします。
[-thetajb]	ThetaJB をリセットします。
[-board]	ボード タイプをリセットします。
[-board_temp]	ボード温度をリセットします。
[-board_layers]	ボード レイヤーをリセットします。
[-design_power_budget]	デザインの消費電力バジェット (W) をリセットします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

電力、XDC

説明

指定の動作条件をデフォルト値にリセットします。動作条件を指定しない場合、すべての動作条件がデフォルト値にリセットされます。

動作条件は、set_operating_conditions コマンドを使用して設定できます。現在の値を確認するには、report_operating_conditions コマンドを使用します。デバイスの環境動作条件は、report_power コマンドによる消費電力解析では使用されますが、タイミング解析では使用されません。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

- voltage <args> (オプション): 電源をデフォルト値にリセットします。電源およびそのデフォルト値は、デバイスファミリによって異なります。
 - grade (オプション): 選択したデバイスの温度グレードをリセットします。デフォルト値は commercial です。
 - process (オプション): ターゲット デバイスの製造プロセスをリセットします。デフォルト値は typical です。
 - junction_temp (オプション): ターゲット デバイスのジャンクション温度をリセットします。デフォルト値は auto です。
 - ambient_temp (オプション): デザインの周囲温度をリセットします。デフォルト値は default です。
 - thetaja (オプション): Theta-JA 熱抵抗をリセットします。デフォルト値は auto です。
 - thetasa (オプション): Theta-SA 熱抵抗をリセットします。デフォルト値は auto です。
 - airflow (オプション): エアフロー (LFM (リニア フィート/分)) をリセットします。デフォルト値はデバイス ファミリによって異なります。
 - heatsink (オプション): ヒートシンクのプロファイルをリセットします。デフォルト値は medium です。
 - thetajb (オプション): Theta-JB 熱抵抗をリセットします。デフォルト値は auto です。
 - board (オプション): モデリングに使用されるボードのサイズをリセットします。デフォルト値は medium です。
 - board_temp (オプション): ボード温度をデフォルト値にリセットします。
 - board_layers (オプション): モデリングに使用されるボードの層数をデフォルトの 12to15 にリセットします。
 - design_power_budget (オプション): デザインの消費バジェットを unspecified (未指定) にリセットします。
 - quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。
- 注記:** コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。
- verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、デザインのすべての動作条件をデフォルト値にリセットしています。

```
reset_operating_conditions
```

次の例では、デザインのジャンクション温度、周囲温度、およびボード温度をデフォルト値にリセットしています。

```
reset_operating_conditions -junction_temp -ambient_temp -board_temp
```

次の例では、電源 Vccint をデフォルト値にリセットしています。

```
reset_operating_conditions -voltage Vccint
```

関連項目

- [report_operating_conditions](#)
- [report_power](#)
- [set_operating_conditions](#)

reset_param

パラメーターをリセットします。

構文

```
reset_param [-quiet] [-verbose] <name>
```

戻り値

元のパラメーター値

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	パラメーター名を指定します。

カテゴリ

[PropertyAndParameter](#) (プロパティおよびパラメーター)

説明

`set_param` コマンドで変更したユーザー定義可能なパラメーターをデフォルト値に戻します。

`report_param` コマンドを使用すると、現在定義されているパラメーターを表示できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 値をリセットするパラメーターの名前を指定します。このコマンドでリセットできるのは、一度に 1 つのパラメーターのみです。

例

次の例では、tcl.statsThreshold パラメーターの値がデフォルト値にリセットされます。

```
reset_param tcl.statsThreshold
```

関連項目

- [get_param](#)
- [list_param](#)
- [report_param](#)
- [set_param](#)

reset_project

現在開いているプロジェクトをリセットします。

構文

```
reset_project [-exclude_runs] [-exclude_ips] [-exclude_sim_runs] [-quiet]
              [-verbose]
```

使用法

名前	説明
<code>[-exclude_runs]</code>	run はリセットしません。
<code>[-exclude_ips]</code>	IP はリセットしません。
<code>[-exclude_sim_runs]</code>	シミュレーション実行はリセットしません。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Project (プロジェクト)

説明

現在のプロジェクトを、合成、シミュレーション、インプリメンテーション、および `write_bitstream` で生成されたさまざまな出力ファイルをクリーンアップして、開始時のソースおよび制約ファイルのみの状態にリセットします。プロジェクトのステートもデザイン フローの開始にリセットします。



ヒント: プロジェクト名前空間ではなくグローバル名前空間のユーザー定義 Tcl 変数は、このコマンドではリセットまたは消去されません。グローバル変数は Vivado の起動中は保持され、Vivado Design Suite を終了したときにのみ消去されます。特定の Tcl 変数を消去するには、`unset` コマンドを使用できます。

引数

`-exclude_runs` (オプション): リセット プロセスから `<project>.runs` フォルダを除外します。runs フォルダはそのまま保持され、ほかのプロジェクト データが削除されます。

`-exclude_ips` (オプション): リセット プロセスから `<project>.srcs/sources_1/ip` フォルダを除外します。IP コアと生成されたターゲットを含む IP フォルダはそのまま保持され、ほかのプロジェクト データが削除されます。

`-exclude_sim_runs` (オプション): リセット プロセスから `<project>.sim` フォルダを除外します。シミュレーション フォルダはそのまま保持され、ほかのプロジェクト データが削除されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、シミュレーション run データを保持して現在のプロジェクトをリセットし、メッセージの非表示設定にかかわらずすべてのメッセージを表示しています。

```
reset_project -exclude_sim_runs -verbose
```

関連項目

- [create_project](#)
- [current_project](#)

reset_property

オブジェクトのプロパティをリセットします。

構文

```
reset_property [-quiet] [-verbose] <property_name> <objects>...
```

戻り値

設定された値、エラーが発生した場合は ""

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><property_name></code>	リセットするプロパティの名前を指定します。
<code><objects></code>	プロパティをリセットするオブジェクトの名前を指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

指定のオブジェクトの指定のプロパティをデフォルト値にリセットします。プロパティにデフォルト値が定義されていない場合は、指定のオブジェクトにそのプロパティは割り当てられません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<property_name>` (必須): リセットするプロパティの名前を指定します。

`<objects>` (必須): プロパティをデフォルト値にリセットするオブジェクトを 1 つまたは複数指定します。

例

次の例では、指定したブロック RAM の DOB_REG プロパティを設定し、その後リセットしています。

```
set_property DOB_REG 1 [get_cells usbEngine1/usbEngineSRAM/  
snoopyRam_reg_19]  
reset_property DOB_REG [get_cells usbEngine1/usbEngineSRAM/  
snoopyRam_reg_19]
```

関連項目

- [create_property](#)
- [get_cells](#)
- [get_property](#)
- [list_property](#)
- [list_property_value](#)
- [report_property](#)
- [set_property](#)

reset_runs

既存の run をリセットします。

構文

```
reset_runs [-prev_step] [-from_step <arg>] [-quiet] [-verbose] <runs>
```

使用法

名前	説明
<code>[-prev_step]</code>	最後の run ステップをリセットします。
<code>[-from_step]</code>	リセットする最初のステップを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><runs></code>	変更する run を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

reset_simulation

既存のシミュレーション実行をリセットします。

構文

```
reset_simulation [-mode <arg>] [-type <arg>] [-quiet] [-verbose] [<simset>]
```

使用法

名前	説明
[-mode]	指定したモードの生成データを削除します。有効な値は behavioral、post-synthesis、post-implementation で、デフォルトは behavioral です。
[-type]	指定したタイプの生成データを削除します。このオプションは、-mode を post-synthesis または post-implementation に設定している場合にのみ有効です。有効な値は functional、timing です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<simset>]	リセットするシミュレーション ファイルセットの名前を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

指定したシミュレーション ファイルセットのコンパイルおよびシミュレーションで生成されたさまざまな出力ファイルをクリーンアップして、現在のシミュレーションを開始状態にリセットします。



重要: ローカル ファイルが警告なしでプロジェクト シミュレーション ファイルから削除されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-mode [behavioral | post-synthesis | post-implementation] (オプション): リセットするシミュレーション モードを指定します。有効な値は、behavioral (ビヘイビアー シミュレーション)、post-synthesis (合成後のシミュレーション)、および post-implementation (インプリメンテーション後のシミュレーション) です。デフォルト モードは behavioral です。

-type [functional | timing] (オプション): -mode behavioral オプションと共に使用することはできません。ネットリストのみの論理シミュレーションを指定するか、ネットリストと SDF ファイルを使用したタイミングシミュレーションを指定します。合成後のタイミングシミュレーションでは、synth_design コマンドからの SDF コンポーネント遅延が使用されます。インプリメンテーション後のタイミングシミュレーションでは、place_design および route_design コマンドからの SDF 遅延が使用されます。

注記: -type オプションを -mode behavioral と共に使用すると、エラーが返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<simset>` (オプション): 削除するシミュレーション ファイルセットを指定します。デフォルトは `sim_1` です。

例

次の例では、`sim_2` シミュレーション ファイルセットを削除して合成後のタイミング シミュレーションをリセットしています。

```
reset_simulation -mode post-synthesis -type timing sim_2
```

reset_ssn

メモリから SSN 結果を削除します。

構文

```
reset_ssn [-quiet] [-verbose] <name>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	結果のセット名を指定します。

カテゴリ

[Report \(レポート\)](#)

説明

指定の結果セットから SSN 結果を削除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 削除する結果の名前を指定します。

例

次の例では、メモリから指定した結果を削除しています。

```
reset_ssn SSN1
```

関連項目

- [report_ssn](#)

reset_switching_activity

指定したオブジェクトのスイッチング アクティビティをリセットします。

構文

```
reset_switching_activity [-default] [-type <args>] [-hier] [-all]
                        [-no_deassert_resets] [-quiet] [-verbose] [<objects>...]
```

使用法

名前	説明
[-default]	デフォルトのスタティック確率およびデフォルトのトグル レートをリセットします。
[-type]	指定のカテゴリのノードを指定します。有効なタイプは、io_output、io_bidir_enable、register、lut_ram、lut、dsp、bram_enable、bram_wr_enable、gt_txdata、gt_rxdata です。
[-hier]	<objects> で指定した階層セルのすべてのレベルでスイッチング アクティビティをリセットします。
[-all]	すべてのネットのスイッチング アクティビティをリセットします。
[-no_deassert_resets]	set_switching_activity -deassert_resets コマンドを使用して適用したリセットのデアサートを元に戻します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	スイッチング アクティビティをリセットするオブジェクトを指定します。

カテゴリ

電力、XDC

説明

デザインに含まれる指定したネット、ポート、ピン、およびセルのスイッチング アクティビティの属性をリセットします。

スイッチング アクティビティを定義するには、set_switching_activity コマンドを使用します。特定のポート、ピン、ネット、またはセルの現在定義されているスイッチング アクティビティを確認するには、report_switching_activity コマンドを使用します。

注記: reset_switching_activity コマンドは、指定したオブジェクトのスイッチング アクティビティをリセットするために使用します。現在のデザインのデフォルト値を変更またはリセットするには、set_switching_activity -default_toggle_rate または -default_static_probability コマンドを使用します。

このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-default` (オプション): 指定のオブジェクトのスタティック確率と信号レートをリセットします。

`-type <arg>` (オプション): 指定したタイプのロジック エンティティのスイッチング アクティビティをリセットします。デフォルトでは、現在のデザインの最上位または指定の `<objects>` に適用されます。`-type` オプションは、現在のデザインの最上位にあるロジック オブジェクトの指定のタイプにコマンドを適用します。`-all` または `-hier` オプションを使用して、コマンドを適用するオブジェクトの範囲を変更できます。有効なロジック タイプは次のとおりです。

- `io_output`: プライマリ出力。
- `io_bidir_enable`: 双方向ポートのイネーブル ピン。
- `register`: 指定のデザイン/階層のすべてのレジスタ出力。
- `lut`: 指定のデザイン/階層のすべての LUT 出力。
- `lut_ram`: 指定のデザイン/階層のすべての分散 RAM 出力。
- `dsp`: 指定のデザイン/階層のすべての DSP 出力。
- `bram_enable`: BRAM のイネーブル ピン (ENARDEN/ENBWREN)。
- `bram_wr_enable`: BRAM のライト イネーブル (WEA/WEBWE)。
- `gt_txdata`: すべての GT の出力 TX データ ピン。
- `gt_rxdata`: すべての GT の出力 RX データ ピン。

`-hier` (オプション): 指定の階層オブジェクトのすべてのレベルでスイッチング アクティビティをリセットします。`-hier` を使用しない場合、現在の階層レベルで指定の `<objects>` のスイッチング アクティビティがリセットされます。

`-all` (オプション): `-type` オプションと共に使用する必要があり、`-type` で指定したロジック オブジェクトのすべてのインスタンスに含まれるネットのスイッチング アクティビティをリセットします。

`-no_deassert_resets` (オプション): `set_switching_activity` コマンドで指定されていた `-deassert_resets` オプションをディスエーブルにします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): スwitching アクティビティをリセットするオブジェクトを指定します。オブジェクトを指定しない場合、すべてのオブジェクトのスイッチング アクティビティがリセットされます。

例

次の例では、すべての出力ポートの信号レートおよびスタティック確率をリセットしています。

```
reset_switching_activity -default [all_outputs]
```

関連項目

- [power_opt_design](#)
- [report_power](#)
- [report_switching_activity](#)
- [set_switching_activity](#)

reset_target

指定したソースのターゲット データをリセットします。

構文

```
reset_target [-quiet] [-verbose] <name> <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	リセットするターゲットを指定します。all を指定すると、生成されたターゲットがすべてリセットされます。
<code><objects></code>	データをリセットするオブジェクトを指定します。

カテゴリ

Project (プロジェクト)、IPFlow (IP フロー)

説明

指定した IP コアのターゲット データを削除します。これにより、指定したターゲットの生成中に生成されたファイルが削除されます。コアは現在のプロジェクトから削除されませんが、参照場所にある関連のターゲット データが削除されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): リセットするターゲット タイプの名前を指定します。有効な値は次のとおりです。

- `all`: 指定のコアのすべてのターゲットをリセットします。
- `synthesis`: 指定したコアの合成ネットリストをリセットします。指定したコアのネットリスト ファイルが削除されます。
- `simulation`: 指定したコアのシミュレーション ネットリストをリセットします。
- `instantiation_template`: 指定したコアのインスタンシエーション テンプレートをリセットします。

<objects> (必須): ターゲット データを削除する IP コア オブジェクトを指定します。

例

次の例では、指定した IP コアのインスタンシエーション テンプレートをリセットしています。

```
reset_target instantiation_template [get_ips blk_mem*]
```

関連項目

- [generate_target](#)

reset_timing

現在のデザインのタイミング情報をリセットします。

構文

```
reset_timing [-invalid] [-clock_reservation] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-invalid]</code>	有効なタイミング制約に加えて無効なタイミング制約もリセットします。
<code>[-clock_reservation]</code>	有効なタイミング制約に加え、自動派生クロックのクロック名の予約もリセットします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)、[Timing \(タイミング\)](#)

説明

現在のデザインのタイミング データおよび制約をリセットします。現在メモリにあるタイミング データおよび制約を消去し、タイミング エンジンでデザインを反復的にはなく包括的に再評価する場合に使用します。

インメモリ デバイスから制約を消去した後、`read_xdc` コマンドを使用して必要な制約を読み込み直す必要があります。Vivado ツールでは制約は自動的に読み込み直されません。



ヒント: メモリのタイミング データは削除されますが、タイミング レポートは削除されません。レポートされたタイミング情報を削除するには、`delete_timing_results` コマンドを使用します。

引数

`-invalid` (オプション): デザインをリセットする際に、有効なタイミング制約だけでなく無効なタイミング制約も削除します。無効な制約は、エラーが含まれるか、見つからないデザイン オブジェクトに割り当てられており、XDC ファイルが読み込まれたときに Vivado タイミング エンジンで無視されるので、タイミング結果には影響しません。無効な制約をリセットすると、インメモリ デザインから削除されるので、制約ファイルに保存していない場合は失われます。

`-clock_reservation` (オプション): 有効なタイミング制約に加え、自動派生クロックのクロック名もリセットします。これにより、以前の予約名にかかわらず、Vivado タイミング エンジンで自動生成クロックの名前を再生成できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、メモリから無効なタイミング制約を含む現在のタイミング データを消去しています。

```
reset_timing -invalid
```

関連項目

- [delete_timing_results](#)
- [report_timing](#)
- [report_timing_summary](#)

resize_net_bus

現在のデザインに含まれるネット バスのサイズを変更します。

構文

```
resize_net_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]  
               <net_bus_name>...
```

使用法

名前	説明
[-from]	新しい開始バス インデックスを指定します。
[-to]	新しい終了バス インデックスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<net_bus_name>	サイズを変更するネット バスの名前を指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

既存のバス ネットのサイズを変更するか (拡大または縮小)、現在のインデックス範囲の番号を付け直します。1 つのコマンドでは、拡大、縮小、または番号の付け直しのいずれかのみを実行可能です。

- バスを拡大するには、現在のインデックス範囲を超える新しいインデックス範囲を指定します。接続されている既存のビットは、そのまま保持されます。
- バスを縮小するには、現在のインデックス範囲未満の新しいインデックス範囲を指定します。ビットを削除するため接続が解除されますが、残りのビットは現在の接続が保持されます。
- 現在のバス インデックスの番号を付け直すには、現在の範囲と同じ幅の新しいインデックス範囲を指定します。バス ビットの番号は変更されますが、接続は変更されません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-from <arg>` (オプション): 指定のバス ネットの新しい開始インデックスを指定します。

`-to <arg>` (オプション): 指定のバスの新しい終了インデックスを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<net_bus_name> (必須): 既存のバスネットの名前を指定します。

例

次の例では、新しい 24 ビット バスを作成し、負のインデックスを含むようにバス インデックスの番号を付け直し、その後 8 ビット バスにサイズを変更しています。

```
create_net tempBus -from 23 -to 0
resize_net_bus tempBus -from -12 -to 11
resize_net_bus tempBus -from 0 -to 7
```

関連項目

- [connect_net](#)
- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [disconnect_net](#)
- [get_nets](#)
- [remove_net](#)
- [resize_pin_bus](#)
- [resize_port_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

resize_pblock

Pblock のサイト範囲制約を移動、サイズ変更、追加、削除します。

構文

```
resize_pblock [-add <args>] [-remove <args>] [-from <args>] [-to <args>]
               [-replace] [-locs <arg>] [-quiet] [-verbose] <pblock>
```

使用法

名前	説明
[-add]	サイト範囲を追加します。
[-remove]	サイト範囲を削除します。
[-from]	移動するサイト範囲を指定します。
[-to]	サイト範囲の移動先を指定します。
[-replace]	既存の範囲をすべて削除します。
[-locs]	ロックの処理方法を指定します。デフォルトは keep_all です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pblock>	サイズを変更する Pblock を指定します。

カテゴリ

Floorplan (フロアプラン)、XDC

説明

指定した Pblock を配置、サイズ変更、移動、削除します。Pblock は、create_pblock コマンドを使用して作成されている必要があります。

Pblock は、独立したまたはオーバーラップした 1 つまたは複数の矩形で定義されたセルのグループで構成されます。次に定義されているさまざまなオプションを使用して、矩形にサイトを追加したり、矩形からサイトを削除したり、既存の Pblock に関連付ける新しい矩形を定義したりできます。

引数

-add <arg> (オプション): Pblock に指定の範囲のサイトを追加します。範囲は、矩形の 1 つの角からその対角線上の角までで指定します。Pblock の範囲を定義するのに使用可能なオブジェクトには、SLICE、DSP48、RAMB18、RAMB36、CLOCKREGION などがあります。たとえば、「SLICE_X0Y0:SLICE_X20Y12」のように指定します。

注記: 複数のサイト タイプは、個別の矩形として追加します。

-remove <arg> (オプション): Pblock から指定の範囲のサイトを削除します。Pblock からサイトを削除すると、Pblock を 1 つまたは複数の矩形で定義するという要件に従うため、Pblock が複数の矩形に分割される場合があります。

`-from <args>` (オプション): `-from` と `-to` オプションはペアとして指定し、1つの場所から別の場所に移動するサイトまたはサイト範囲を指定します。

`-to <args>` (オプション): `-from` と `-to` オプションはペアとして指定し、1つの場所から別の場所に移動するサイトまたはサイト範囲を指定します。

`-locs <args>` (オプション): Pblock を移動またはサイズ変更したときに、Pblock の配置されているロジックをどのように処理するかを指定します。有効な値は次のとおりです。

- `keep_all`: すべての配置を現在設定されているとおり保持します。`-locs` を指定しない場合、これがデフォルトです。Pblock の外に配置されているロジックは、Pblock に割り当てられなくなります。
- `keep_only_fixed`: ユーザーが配置したロジック (固定) のみを保持します。自動的に配置された未固定のロジックは配置が解除されます。
- `keep_none`: すべてのロジックの配置を解除します。
- `move`: すべての配置を Pblock の座標に相対して移動します。
- `move_unfixed`: 未固定の要素のみを移動するよう指定します。ユーザーが配置した (固定) ロジックは移動しません。
- `trim`: 新しい Pblock の境界外となるロジックの配置を解除します。配置されているロジックで Pblock の境界内になるものは、配置が保持されます。
- `trim_unfixed`: 新しい Pblock の境界外となる未固定の要素のみの配置を解除します。

`-replace` (オプション): Pblock に関連付けられているすべての矩形を削除します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pblock>` (必須): サイズ変更、移動、または削除する Pblock の名前を指定します。

例

次の例では、Pblock にスライス範囲を追加し、別のスライスを削除して、Pblock のサイズを変更しています。すべてのインスタンスの配置は変更しません。

```
resize_pblock block3 -add SLICE_X6Y67:SLICE_X11Y71 \
    -remove SLICE_X6Y71:SLICE_X7Y71 -locs keep_all
```

次の例では、Pblock 領域を作成し、CLOCKREGION の範囲を追加して Pblock のエリアを定義しています。

```
create_pblock pblock_1
resize_pblock pblock_1 -add {CLOCKREGION_X0Y10:CLOCKREGION_X1Y11}
```

次の例では、指定した Pblock にスライス範囲を追加し、既存のスライス範囲を削除して、新しい Pblock の境界外となるロジックを削除しています。その後、新しいスライス範囲とブロック RAM を別の矩形として追加します。

```
resize_pblock block3 -add SLICE_X3Y8:SLICE_X10Y3 \  
    -remove SLICE_X6Y67:SLICE_X11Y71 -locs trim  
resize_pblock block3 -add {SLICE_X6Y67:SLICE_X11Y71 \  
    RAMB18_X0Y2:RAMB18_X1Y4}
```

関連項目

- [add_cells_to_pblock](#)
- [create_pblock](#)
- [place_pblocks](#)

resize_pin_bus

現在のデザインに含まれるピン バスのサイズを変更します。

構文

```
resize_pin_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]  
               <pin_bus_name>...
```

使用法

名前	説明
[-from]	新しい開始バス インデックスを指定します。
[-to]	新しい終了バス インデックスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<pin_bus_name>	サイズを変更するピン バスの名前を指定します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

既存のバス ピンのサイズを変更するか (拡大または縮小)、現在のピン インデックス範囲の番号を付け直します。1 つのコマンドでは、拡大、縮小、または番号の付け直しのいずれかのみを実行可能です。

- バスを拡大するには、現在のピン インデックス範囲を超える新しいインデックス範囲を指定します。接続されている既存のピンは、そのまま保持されます。
- バスを縮小するには、現在のピン インデックス範囲未満の新しいインデックス範囲を指定します。バス ピンを削除するため接続が解除されますが、残りのピンは現在の接続が保持されます。
- 現在のバス インデックスの番号を付け直すには、現在の範囲と同じ幅の新しいピン インデックス範囲を指定します。ピン インデックスの番号は変更されますが、接続は変更されません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-from <arg> (オプション): 指定のバス ピンの新しい開始インデックスを指定します。

-to <arg> (オプション): 指定のバス ピンの新しい終了インデックスを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<bus_pin_name>` (必須): 変更するバス ピンの名前を指定します。ピン名は、ピンを割り当てるセル インスタンスを基準に階層で指定する必要があります。デザインの最上位に作成されたピンはポートであり、`resize_port_bus` コマンドを使用してサイズを変更する必要があります。

例

次の例では、ブラック ボックス セルを作成し、指定の階層セルに 24 ビットの双方向バスを作成し、バス ピンの幅を 32 ビットに拡張して、負のバス インデックスを含むように番号を付け直しています。

```
create_cell -reference dmaBlock -black_box usbEngine0/myDMA
create_pin -direction INOUT -from 0 -to 23 usbEngine0/myDMA/dataBus
resize_pin_bus -from 0 -to 31 usbEngine0/myDMA/dataBus
resize_pin_bus -from -16 -to 15 usbEngine0/myDMA/dataBus
```

関連項目

- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [get_pins](#)
- [remove_pin](#)
- [resize_net_bus](#)
- [resize_port_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

resize_port_bus

現在のデザインに含まれるポート バスのサイズを変更します。

構文

```
resize_port_bus [-from <arg>] [-to <arg>] [-quiet] [-verbose]  
                <port_bus_name>...
```

使用法

名前	説明
[-from]	新しい開始バス インデックスを指定します。
[-to]	新しい終了バス インデックスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<port_bus_name>	サイズを変更するポート バスの名前を指定します。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

既存のバス ポートのサイズを変更するか (拡大または縮小)、現在のポート インデックス範囲の番号を付け直します。1 つのコマンドでは、拡大、縮小、または番号の付け直しのいずれかのみを実行可能です。

- バスを拡大するには、現在のポート インデックス範囲を超える新しいインデックス範囲を指定します。接続されている既存のポートは、そのまま保持されます。
- バスを縮小するには、現在のポート インデックス範囲未満の新しいインデックス範囲を指定します。バス ポートを削除するため接続が解除されますが、残りのポートは現在の接続が保持されます。
- 現在のバス インデックスの番号を付け直すには、現在の範囲と同じ幅の新しいポート インデックス範囲を指定します。ポート インデックスの番号は変更されますが、接続は変更されません。

ネットリストを編集すると、現在のデザインのネットリストのメモリにあるビューが変更されます。ソース ファイルセットのファイルおよびディスク上のデザインは変更されません。ネットリストに加えた変更は、`write_checkpoint` コマンドを使用してデザイン チェックポイントとして保存するか、`write_*` コマンドを使用して Verilog、VHDL、または EDIF 形式のネットリスト ファイルにエクスポートできます。

注記: エラボレート済み RTL デザインでは、ネットリストを編集することはできません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-from <arg>` (オプション): 指定のバス ポートの新しい開始インデックスを指定します。

`-to <arg>` (オプション): 指定のバス ポートの新しい終了インデックスを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<bus_port_name>` (必須): 変更するバス ポートの名前を指定します。

例

次の例では、32 ビットの出力バス ポートを作成し、負のバス インデックスを含めるためポートの番号を付け直し、バス幅を 32 ビットから 16 ビットに縮小します。

```
create_port -direction out -from 0 -to 31 outPorts
resize_port_bus -from -16 -to 15 outPorts
resize_port_bus -from -8 -to 7 outPorts
```

関連項目

- [create_net](#)
- [create_pin](#)
- [create_port](#)
- [get_ports](#)
- [remove_port](#)
- [resize_net_bus](#)
- [resize_pin_bus](#)
- [write_checkpoint](#)
- [write_edif](#)
- [write_verilog](#)
- [write_vhdl](#)

restart

シミュレーションをデザインを読み込んだ後の状態に戻し、時間を 0 に設定します。

構文

```
restart [-quiet] [-verbose]
```

戻り値

ブレークポイント、Tcl フォース、波形ビューアーの設定は保持し、その他の Tcl コマンドの結果をすべてクリーンアップ

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションを時間を 0 にリセットすることにより、デザインを再読み込みしたのと同様の初期状態に戻します。

`restart` コマンドではブレークポイント、Tcl フォース、波形設定ウィンドウは保持されますが、すべてのシミュレーション値がリセットされ、Tcl コマンドのその他の結果がすべてクリーンアップされます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のシミュレーションを初期状態に戻しています。

```
restart
```

関連項目

- [current_time](#)
- [run](#)
- [stop](#)

resume_hw_hbm_amon

指定したハードウェア HBM に対して一時停止していたアクティビティ モニターの実行を再開します。

構文

```
resume_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

`resume_hw_hbm_amon` コマンドは、Vivado ハードウェア マネージャーで `pause_hw_hbm_amon` コマンドを使用して一時停止していた HBM アクティビティ モニターの実行を再開します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (必須): 再開する HBM アクティビティ モニター オブジェクト (`hw_hbm`) を指定します。

例

次の例では、指定した HBM コアに関連付けられている HBM アクティビティ モニターを再開しています。

```
resume_hw_hbm_amon [get_hw_hbms *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)

- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [run_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

route_design

現在のデザインを配線します。

構文

```
route_design [-unroute] [-release_memory] [-nets <args>] [-physical_nets]
             [-pins <arg>] [-directive <arg>] [-tns_cleanup] [-no_timing_driven]
             [-preserve] [-delay] [-auto_delay] -max_delay <arg> -min_delay <arg>
             [-timing_summary] [-finalize] [-ultrathreads] [-quiet] [-verbose]
```

使用法

名前	説明
[-unroute]	デザイン全体の配線を解除するか、-nets または -pins を指定した場合は指定のネット/ピンの配線を解除します。
[-release_memory]	配線のメモリを解放します。ほかのオプションと互換性はありません。
[-nets]	指定したネットに対して操作を実行します。
[-physical_nets]	すべての物理ネットに対して操作を実行します。
[-pins]	指定したピンに対して操作を実行します。
[-directive]	コマンドのモードを指定します。有効な値は、「引数」セクションを参照してください。デフォルトは Default です。
[-tns_cleanup]	TNS クリーンアップを実行します。
[-no_timing_driven]	タイミングドリブン モードで実行しません。
[-preserve]	既存の配線を保持します。
[-delay]	-nets または -pins オプションと共に使用し、遅延ドリブン モードで配線します。
[-auto_delay]	-nets または -pins オプションと共に使用し、制約ドリブン モードで配線します。
-max_delay	-pins オプションと共に使用し、ピンに最大遅延制約を指定します。このオプションを指定すると、-delay も指定されます。
-min_delay	-pins オプションと共に使用し、ピンに最大遅延制約を指定します。このオプションを指定すると、-delay も指定されます。
[-timing_summary]	配線後のサインオフ タイミング サマリをイネーブルにします。
[-finalize]	インタラクティブ モードの route_design を完了します。
[-ultrathreads]	ターボ モード配線をイネーブルにします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Tools \(ツール\)](#)

説明

現在のデザインに含まれるネットを配線し、ターゲット パーツでのロジック接続を完成させます。

定義済みの配線ストラテジを `route_design -directive` コマンドを使用して選択するか、または配線オプションを設定して独自の配線ストラテジを定義できます。

配線は、`route_design` コマンドを使用してすべて自動的に実行するか、`route_design` コマンドのさまざまなオプションを使用して反復実行し、配線を完了させてタイミング クロージャを達成できます。配線を反復的に実行すると、まずクリティカルなネットを配線してからそれほどクリティカルでないネットを配線するなどして配線プロセスを制御したり、エフォート レベルやタイミング アルゴリズムを調整したりできます。

配線はインプリメンテーションの 1 つの段階です。Vivado ツールをプロジェクト モードで実行している場合は、インプリメンテーションは `launch_runs` コマンドを使用してすべてを自動的に実行できます。

非プロジェクト モードでは、インプリメンテーション プロセスを `opt_design`、`place_design`、`phys_opt_design`、`power_opt_design`、および `route_design` の各コマンドを使用して手動で実行する必要があります。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。



ヒント: `route_design` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメーターの設定に関する詳細は、`set_param` コマンドを参照してください。

配置および配線はどちらも、インクリメンタル インプリメンテーション フローを使用して、デザイン チェックポイント ファイル (DCP) に保存されている結果に基づいて、インクリメンタルに実行できます。インクリメンタル配置および配線の詳細は、`read_checkpoint` コマンドまたは『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。

このコマンドを実行するには、配置済みデザインが必要であり、`opt_design` コマンドを実行しておくことをお勧めします。

引数

`-unroute <arg>` (オプション): デザインのネットの配線を解除します。引数が指定されていない場合は、すべてのネットの配線が解除されます。`-unroute` オプションを指定した場合、`route_design` コマンドでネットは配線されません。

- `-nets` オプションと共に使用すると、配線を解除するネットを指定できます。
- `-pins` オプションと共に使用すると、指定のピンから最も近いネットの分岐までの配線が解除されます。
- `-physical_nets` オプションと共に使用すると、論理 1 および論理 0 のネットの配線を解除できます。

`-release_memory` (オプション): 後続の配線実行用に配線のメモリ リソースを解放します。このオプションを使用すると配線は実行されず、配線機能により使用されているメモリを解放して配線初期化を削減します。この後配線を実行するには、デザインを読み込み直す必要があります。

`-nets <args>` (オプション): 指定したネット オブジェクトのみを配線または配線解除します。ネット オブジェクトは、`get_nets` コマンドを使用して指定する必要があります。



ヒント: `-nets`、`-physical_nets`、または `-pins` オプションを指定すると、指定のネットに対して配線ソリューションを見つけるのにクイック配線が使用され、タイミング遅延は無視されます。遅延が最も少ない配線を見つける際に `-delay` を使用してください。

`-physical_nets` (オプション): 論理 0 または論理 1 のネットのみを配線または配線解除します。

`-pins <args>` (オプション): 指定したピンまで配線するか、またはその配線を解除します。セル入力を指定する必要があります。指定したピンが複数のファンアウト ネットで駆動されている場合、ネットとピンの間の配線セグメントのみに操作が実行されます。

`-directive <arg>` (オプション): 特定のデザイン目標を達成するよう配線します。1つの `route_design` コマンドに対して 1つのモードのみを指定できます。値では大文字/小文字が区別されます。有効な値は次のとおりです。

- `Explore`: 初期配線の後、タイミングに基づいて異なるクリティカル パス配線を試します。
- `AggressiveExplore`: 元のタイミング バジェットを維持しながら、さらに多くのクリティカル パス配線を試します。タイミング制約を満たすためにより積極的な最適化しきい値が使用されるので、`Explore` 指示子の場合よりも実行時間が大幅に長くなる可能性があります。
- `NoTimingRelaxation`: 配線を完了するためにタイミングを緩和しないようにします。配線でタイミング満たすのが困難である場合、元のタイミング制約を満たすため実行時間が長くなります。
- `MoreGlobalIterations`: 最終段階だけでなく、すべての段階で詳細なタイミング解析を使用し、タイミングの向上が少しであってもグローバル 反復を実行します。
- `HigherDelayCost`: 配線の内部コスト関数を調整して反復実行で遅延に焦点を置き、実行時間が長くなる代わりにパフォーマンスを向上します。
- `AdvancedSkewModeling`: 配線のすべての段階でより正確なスキュー モデルを使用します。スキューの大きいクロック ネットワークでデザインのパフォーマンスが向上する場合があります。
- `AlternateCLBRouting` (UltraScale のみ): 実行時間は長い配線の密集を緩和する代替配線アルゴリズムを選択します。
- `RuntimeOptimized`: 反復回数を少なくし、デザイン パフォーマンスよりも実行時間を短縮することを優先します。
- `Quick`: 最も高速な、タイミングドリブンでない、有効なデザインを得るために最低限の配線を実行します。
- `Default`: `route_design` をデフォルト設定で実行します。



重要: `-directive` オプションは全体的な配線ストラテジを制御するもので、`-preserve` および `-tns_cleanup` 以外の `route_design` コマンドのオプションとは互換性がありません。`-quiet` および `-verbose` とも使用できます。再利用の高いデザインおよび `read_checkpoint -incremental` コマンドで定義されたインクリメンタル インプリメンテーション フローでは、`Explore`、`Quick`、および `Default` 指示子のみを使用できます。`-directive` オプションの使用法の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

`-tns_cleanup` (オプション): デフォルトでは、実行時間を短縮するため、トータル ネガティブ スラック (TNS) パスではなくワースト ネガティブ スラック (WNS) パスを最適化するよう配線が実行されます。このオプションを使用すると、配線の最後に、TNS パス (WNS パス以外のタイミングが満たされていないパス) の修正が試みられます。このオプションを使用すると TNS は削減されますが、実行時間が増加する可能性があります。WNS には影響しません。配線後に `phys_opt_design` コマンドを実行して WNS パスに焦点を置き、配線で修正可能な TNS パスに時間を費やさないようにする場合に、`-tns_cleanup` オプションの使用をお勧めします。このオプションは、`-directive` オプションと共に使用できます。

`-no_timing_driven` (オプション): デフォルトのタイミングドリブン配線アルゴリズムをディスエーブルにします。このオプションを使用すると短時間で配線結果を得ることができますが、タイミング制約は無視されます。

`-preserve` (オプション): 完了している既存の配線を保持し、配線し直しません。IS_ROUTE_FIXED または FIXED_ROUTE プロパティを使用して固定されている配線は配線し直されないの、このオプションは適用されません。配線は、現在の `route_design` コマンドでのみ保持されます。



ヒント: 部分的に配線済みのネットは、接続を完了させるために配線し直されます。部分的に配線済みのネットの配線を保持する場合は、配線の保持する部分に FIXED_ROUTE プロパティを適用する必要があります。

`-delay` (オプション): `-nets` または `-pins` オプションを指定している場合にのみ使用できます。デフォルトでは、`-nets` または `-pins` オプションを指定すると、ネットの配線は配線の実行時間が最短になるようにタイミング制約を無視して実行されます。`-delay` オプションを指定すると、配線インターコネクト遅延が最小になるように配線されますが、タイミング制約は無視されます。



重要: `-delay` オプションを使用して同時に配線する複数のネットを指定できますが、配線リソースの競合が発生することがあります。`-delay` オプションでは指定されたすべてのネットに対して最短配線を達成するた配線リソースが再利用されるので、ネットどうしが近くにあると、Vivado 配線でノード オーバーラップ エラーが表示されることがあります。そのため、`-delay` オプションを使用して最もクリティカルなものから順に 1 つずつネットおよびピンを指定して配線することをお勧めします。

`-auto_delay` (オプション): `-nets` または `-pins` オプションを指定している場合にのみ使用できます。`-auto_delay` オプションは配置済みデザインに使用し、指定するネットまたはピンの数を 100 未満に制限することをお勧めします。`-auto_delay` オプションを指定すると、定義されたタイミング制約を使用して、セットアップおよびホールド クリティカル パスを優先するよう配線されます。

`-max_delay <arg>` (オプション): `-pins` オプションを使用している場合にのみ有効で、インターコネクト遅延が指定の遅延 (ps) 以下になるよう配線します。このオプションを指定すると、`-delay` オプションも指定されます。

注記: `-max_delay` および `-min_delay` オプションはインターコネクト遅延の範囲を指定するもので、ロジックまたはサイト内の遅延は指定しません。配線は、インターコネクトの遅延制限を満たすよう実行されます。

`-min_delay <arg>` (オプション): `-pins` オプションを使用している場合にのみ有効で、インターコネクト遅延が指定の遅延 (ps) 以上になるよう配線します。このオプションを指定すると、`-delay` オプションも指定されます。

`-timing_summary` (オプション): デフォルトでは、Vivado 配置の内部見積もりタイミングに基づく最終的なタイミング サマリはログに出力されます。この内部見積もりタイミングは、遅延見積もりでの不必要に悪い見積もりのため、実際の配線タイミングとは多少異なります。`-timing_summary` オプションを使用すると、Vivado スタティック タイミング解析機能が起動され、実際の配線遅延に基づくタイミング サマリがレポートされます。ただし、スタティック タイミング解析のために実行時間が長くなります。タイミング サマリには、ワースト ネガティブ スラック (WNS)、トータル ネガティブ スラック (TNS)、ワースト ホールド スラック (WHS)、およびトータル ホールド スラック (THS) が含まれます。値は、配線後のデザインに対して `report_timing_summary` コマンドを実行した場合と同じになります。



ヒント: `-directive Explore` オプションを使用すると、Vivado スタティック タイミング解析も起動されます。

`-finalize` (オプション): 配線をインタラクティブに実行している場合、`route_design -finalize` を指定して部分的に配線された接続を完了できます。

`-ultrathreads` (オプション): 結果の再現性を低下させて、配線の実行時間を短縮します。このオプションを使用すると、配線は高速に実行されますが、同一の run であっても配線に多少のバリエーションが発生します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、デザイン全体を配線し、クリティカル パス遅延を向上するために複数のアルゴリズムを試しています。

```
route_design -directive Explore
```

次の例では、インターコネクト遅延が最小になるようにタイミング クリティカル ネット \$criticalNets を配線した後、そのネットを IS_ROUTE_FIXED プロパティを使用して固定し、デザインの残りの部分を RuntimeOptimized モードを使用して配線しています。

```
route_design -delay -nets $criticalNets
set_property IS_ROUTE_FIXED 1 $criticalNets
route_design -directive RuntimeOptimized
```

次の例では、指定のネットを最短の実行時間で配線しています。

```
route_design -nets [get_nets ctrl0/ctr*]
```

次の例では、指定のネットをインターコネクト遅延が最短になるように配線しています。

```
route_design -nets [get_nets ctrl0/ctr*] -delay
```

次の例では、指定のピンをまでを配線しています。

```
route_design -pins [get_pins ctrl0/reset_reg/D ctrl0/ram0/ADDRARDADDR]
```

次の例では、指定のピンの遅延が 500 ps 以下になるよう配線しています。

```
route_design -pins [get_pins ctrl0/reset_reg/D] -max_delay 500
```

次の例では、指定のピンの遅延が 200 ps より大きくなるよう配線しています。

```
route_design -pins [get_pins ctrl0/ram0/ADDRARDADDR] -min_delay 200
```

関連項目

- [get_nets](#)
- [get_pins](#)
- [launch_runs](#)
- [opt_design](#)
- [phys_opt_design](#)
- [place_design](#)
- [power_opt_design](#)
- [read_checkpoint](#)
- [set_property](#)
- [write_checkpoint](#)

run

シミュレーションを指定した時間実行します。

構文

```
run [-all] [-quiet] [-verbose] [<time>] [<unit>]
```

使用法

名前	説明
[-all]	シミュレーションをブレークポイント、例外、またはキューにイベントがなくなるまで実行します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<time>]	シミュレーション時間を指定します。
[<unit>]	時間の単位を指定します。有効な値は fs、ps、ns、us、ms、sec です。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションを現在の時間から指定の時間またはシミュレーションが停止するまで実行します。

実行中のシミュレーションは、あらかじめ指定された時間、HDL ソース コードの特定のブレークポイント、TRUE 条件の発生、イベントがなくなるまで回路を評価、または範囲外の値などのランタイム エラーの発生により停止できます。

run コマンドでは既存のシミュレーションを指定の時間またはイベントがなくなるまで実行します。時間は、現在のシミュレーションの単位または指定の単位で、浮動小数点値で指定します。

引数

-all: イベント キューにイベントがなくなるまで、ブレークポイントまたは有効な条件が発生するまで、またはランタイム例外が発生するまでシミュレーションを実行します。

注記: ほかのオプションを指定しない場合、これがデフォルトです。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<time> (オプション): シミュレーションの実行時間を指定します。時間は浮動小数点値で指定するか、キーワード `all` を指定します。

注記: キーワード `all` を使用するの、`-all` を使用するのと同じです。

<unit> (オプション): 実行時間の単位を指定します。指定可能な単位は `fs`、`ps`、`ns`、`us`、`ms`、または `sec` です。時間を指定する数値と単位の間スペースを入れても入れなくてもかまいません。デフォルトは `time_unit` の値です。

例

次の例では、既存のシミュレーションをデフォルトの単位 (`ns`) で指定したシミュレーション時間実行します。

```
run 1000
```

次の例では、既存のシミュレーションを `300 μs` 間実行します。

```
run 300 us
```

次の例では、現在のシミュレーションをイベント キューにイベントがなくなるまで、ブレークポイントまたは有効な条件が発生するまで、またはシミュレーション ランタイム例外が発生するまで実行します。

```
run -all
```

関連項目

- [add_bp](#)
- [add_condition](#)
- [restart](#)
- [step](#)
- [stop](#)

run_hw_axi

ハードウェア AXI の読み出し/書き込みトランザクションを実行し、ハードウェア AXI オブジェクトのトランザクション ステータスをアップデートします。

構文

```
run_hw_axi [-queue] [-quiet] [-verbose] <hw_axi_txns>...
```

使用法

名前	説明
<code>[-queue]</code>	トランザクションをキューに配置します。デフォルト: 0
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_axi_txns></code>	AXI バスに対して実行するハードウェア AXI トランザクション オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定した JTAG to AXI Master コアに定義されている AXI トランザクションを実行します。

AXI トランザクションは `create_hw_axi_txns` コマンドで作成します。

指定のハードウェア AXI 読み出し/書き込みトランザクションを実行し、関連付けられているハードウェア AXI (`hw_axi`) オブジェクトのトランザクション ステータスをアップデートします。

引数

`-queue` (オプション): キュー モードで指定した `hw_axi` トランザクションを実行します。キュー モードでは、JTAG to AXI Master FIFO に 16 個の読み出しトランザクションと 16 個の書き込みトランザクションを待機させ、トランザクション間のレイテンシを短くし、パフォーマンスを向上するため、連続発行できます。キューに挿入しないトランザクションは、送信されると同時に実行されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_axi_txns> (必須): AXI バス上で実行するハードウェア AXI トランザクション オブジェクトを指定します。オブジェクトは `get_hw_axi_txns` コマンドを使用して指定できます。

例

次の例では、指定の hw_axi オブジェクトで現在定義されている AXI トランザクションを実行しています。

```
run_hw_axi [get_hw_axi_txns [get_hw_axis]]
```

次の例では、AXI トランザクションをキュー モードで実行しています。

```
run_hw_axi -queue [get_hw_axi_txns txn_1] [get_hw_axi_txns txn_2] \
[get_hw_axi_txns txn_3] [get_hw_axi_txns txn_4]
```

次の例では、AXI 読み出しおよび書き込みトランザクションを作成し、hw_axi を実行して、その結果をレポートしています。

```
create_hw_axi_txn wr_txn [lindex [get_hw_axis] 0] -address 80000000 \
-data {11112222 33334444 55556666 77778888} -len 4 -type write
create_hw_axi_txn rd_txn [lindex [get_hw_axis] 0] -address 80000000 \
-len 4 -type read

run_hw_axi [get_hw_axi_txns wr_txn]
set wr_report [report_hw_axi_txn wr_txn -w 32]
puts $wr_report

run_hw_axi [get_hw_axi_txns rd_txn]
set rd_report [report_hw_axi_txn rd_txn -w 32]
puts $rd_report

close_hw_target;
disconnect_hw_server;
```

関連項目

- [delete_hw_axi_txn](#)
- [get_hw_axis](#)
- [get_hw_axi_txns](#)
- [refresh_hw_axi](#)
- [reset_hw_axi](#)

run_hw_hbm_amon

指定したハードウェア HBM に対してアクティビティ モニターの実行をイネーブルにします。

構文

```
run_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

HBM アクティビティ モニターは、統合高帯域幅メモリ (HBM) コントローラーを含む特定のザイリンクス UltraScale + デバイスのパフォーマンス モニターおよび温度センサーにリアルタイムにアクセスするために使用できます。このコアの詳細は、『LogiCORE IP 製品ガイド: AXI High Bandwidth Memory Controller』(PG276) を参照してください。HBM コントローラーおよびメモリ スタックにはパフォーマンス カウンターと温度センサーの両方が含まれ、ザイリンクス Vivado ハードウェア マネージャー内から HBM アクティビティ モニターを使用してアクセスできます。HBM アクティビティ モニターは、読み出し、書き込み、全体的なデータのスループットのほか、デバイス温度を表示します。アクティビティ モニターを実行すると、データを表示、収集、CSV ファイルにエクスポートできます。

`run_hw_hbm_amon` コマンドは、`add_hw_hbm_pc` コマンドを使用して設定した HBM アクティビティ モニターを Vivado ハードウェア マネージャーで実行します。

このコマンドを実行すると、収集されたデータが記述された CSV ファイルのファイル パスと名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (必須): 実行する HBM アクティビティ モニター オブジェクト (`hw_hbm`) を指定します。

例

次の例では、指定した HBM コアに関連付けられている HBM アクティビティ モニターを実行しています。

```
run_hw_hbm_amon [get_hw_hbms *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)
- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [pause_hw_hbm_amon](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [stop_hw_hbm_amon](#)

run_hw_ila

ハードウェア ILA をトリガー待機状態にします。

構文

```
run_hw_ila [-trigger_now] [-compile_only] [-file <arg>] [-force] [-quiet]
           [-verbose] [<hw_ilas>...]
```

使用法

名前	説明
<code>[-trigger_now]</code>	即座にトリガーおよびキャプチャを実行します。
<code>[-compile_only]</code>	コンパイルトリガー ステート マシン ファイルをテストするだけで、アップロードしません。
<code>[-file]</code>	スタートアップ ファイル名でトリガーします。ILA コアはトリガー待機状態になりません。
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<hw_ilas>]</code>	ハードウェア ILA を指定します。デフォルトは、現在のハードウェア ILA です。

カテゴリ

Hardware (ハードウェア)

説明

現在のハードウェア デバイス上の指定したハードウェア ILA デバッグ コアをトリガー待機状態にし、実行します。

ILA (Integrated Logic Analyzer) デバッグ コアを使用すると、ザイリンクス FPGA にプログラムされているインプリメント済みデザイン (デザイン ビットストリーム) のインシステム デバッグを実行できます。ILA コアには、ブールトリガー論理式、エッジ遷移トリガーなど、現代のロジック アナライザーで提供される多数のアドバンス機能が含まれています。ILA コアを使用すると、デザインの特定の信号のプロープ、プログラムされたハードウェア イベントでのトリガー、ザイリンクス FPGA からのリアルタイムでのデータ サンプル キャプチャを実行できます。ILA コアの詳細は、『LogiCORE IP Integrated Logic Analyzer 製品ガイド』 (PG172) を参照してください。

ILA デバッグ コアは、デザインの RTL ソース ファイルに追加するか、`create_debug_core` コマンドを使用して合成済みネットリストに追加できます。デザインへのデバッグ コアおよび信号プロープの追加に関する詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。

デバッグ コアとプロープは `write_debug_probes` コマンドを使用してプロープ ファイル (.ltx) に記述され、ハードウェア デバイス (hw_device) オブジェクトの PROBES.FILE および PROGRAM.FILE プロパティを使用してビットストリーム ファイル (.bit) と共にハードウェア デバイスに関連付けます。ハードウェア デバイスをプログラムするには、`program_hw_devices` コマンドを使用します。デザインの ILA デバッグ コアは、`get_hw_ilas` コマンドを使用してハードウェア マネージャーからアクセスできます。ILA デバッグ コアに割り当てられているデバッグ プロープは、`get_hw_probes` コマンドを使用して取得できます。

ILA デバッグ コアを使用したハードウェアでのデザインのデバッグ手順は、次のとおりです。

1. `connect_hw_server` および `open_hw_target` コマンドを使用して、ハードウェア サーバーとターゲットを接続します。
2. `program_hw_devices` コマンドを使用して、FPGA をビットストリーム ファイル (.bit) およびプローブ ファイル (.ltx) でプログラムします。
3. `set_property` コマンドを使用して ILA のプロパティを設定することにより、ILA デバッグ コアのトリガー イベントおよびキャプチャ条件を設定します。
4. `run_hw_ila` コマンドを使用して、サイリンクス FPGA 上の ILA デバッグ コア トリガーを待機状態にします。トリガー イベントが発生するか、キャプチャ条件が満たされると、データが ILA キャプチャ バッファに挿入されます。
5. `upload_hw_ila_data` コマンドを使用して、サンプリングされたデータを `hw_device` から Vivado ロジック解析のハードウェア ILA データ (`hw_ila_data`) オブジェクトに読み込みます。
6. `display_hw_ila_data` コマンドを使用して、Vivado ロジック解析機能の波形ウィンドウでキャプチャされたデータを表示します。

ILA デバッグ コアは、ハードウェア ILA (`hw_ila`) オブジェクトの `CONTROL` プロパティを使用して、デバッグ プローブでの特定のイベントまたは条件でトリガーされ、特定の条件化でデータがキャプチャされるよう設定できます。これらのプロパティは、`set_property` コマンドを使用して設定します。デザインを監視するためにデバッグ コアおよび信号プローブを設定するプロパティの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』 (UG908) を参照してください。



推奨: Vivado IDE では、`hw_ila` およびハードウェア プローブ (`hw_probe`) オブジェクトのトリガーおよびキャプチャ条件を設定するグラフィカル インターフェイスが提供されています。Vivado IDE を使用して、`hw_ila` オブジェクトを設定および実行するのに必要なプロパティを表示できます。

デバッグ コアを設定するのに使用する `hw_ila` オブジェクトのプロパティは、次のとおりです。

- `CONTROL.DATA_DEPTH`: デフォルトでは、ILA デバッグ コアを作成またはデザインに挿入したときに設定される `MAX_DATA_DEPTH` に設定されます。データ深さは、`hw_ila` オブジェクトでデータ ウィンドウでキャプチャ可能なデータ サンプルの数を定義します。1 から `hw_ila` の最大データ深さ (`MAX_DATA_DEPTH`) の間の 2 のべき乗で整数として設定します。

注記: `DATA_DEPTH` の値と `CONTROL.WINDOW_COUNT` とには、 $DATA_DEPTH * WINDOW_COUNT = MAX_DATA_DEPTH$ という関係があります。

- `CONTROL.WINDOW_COUNT`: ILA コアの `MAX_DATA_DEPTH` を複数のデータ ウィンドウに分割し、複数のトリガー イベントからのサンプル データを格納できるようにします。データ ウィンドウが 10 個の場合、最初のトリガー イベントで最初のウィンドウにデータが格納され、その後のトリガー イベントまたはキャプチャ条件で各ウィンドウに順にデータが格納されます。
- `CONTROL.TRIGGER_POSITION`: `DATA_DEPTH` に関連する整数値です。サンプル データ バッファでのトリガー イベントの位置を指定します。`DATA_DEPTH` が 1024 の場合、位置 0 は最初 (最も左) のデータ バッファ、位置 1023 は最後 (最も右) のデータ バッファを示します。`TRIGGER_POSITION` を指定すると、トリガー イベントの前のサンプル データをキャプチャできます。たとえば、`DATA_DEPTH` が 256 の場合、`TRIGGER_POSITION` を 100 に設定すると、トリガー前の 100 個のデータ サンプルとトリガー時およびトリガー後の 155 個のデータ サンプルをキャプチャできます。

注記: `run_hw_ila -trigger_now 1` の場合、`TRIGGER_POSITION` は Vivado ロジック解析機能の波形ウィンドウでトリガー マークの位置を移動するだけです。これは、トリガー イベントは直ちに発生するので、トリガー前のデータ サンプルをキャプチャする時間がないからです。

- `CONTROL.TRIGGER_MODE`: 有効な値は次のとおりです。
 - 。 `BASIC_ONLY`: トリガー条件は、関連の各 ILA プローブ上の値を評価する `TRIGGER_CONDITION` を使用したブール式の結果です。

- BASIC_OR_TRIG_IN: ILA コアは、プローブ値を考慮するブール式またはコアの TRIG_IN ポートによりトリガーされます。
- ADVANCED_ONLY: ILA コアは、ユーザー定義のトリガー ステート マシン (TSM) で定義されたアドバンス トリガー機能を持つよう設定されます。
- ADVANCED_OR_TRIG_IN: ILA コアは、TMS またはコアの TRIG_IN ポートによりトリガーされます。
- TRIG_IN_ONLY: ILA コアは、コアの TRIG_IN ポートによりトリガーされます。
- CONTROL.TRIGGER_CONDITION: ILA デバッグ コアのトリガー条件の評価に使用されるプローブのコンパレータを評価するブール式を定義します。条件が真の場合、BASIC トリガー モードが使用されます。有効な値は、次のとおりです。
 - AND: すべてのプローブ コンパレータが真の場合にトリガー条件が真となり、それ以外の場合は偽となります。
 - NAND: 少なくとも 1 つのプローブ コンパレータが偽の場合にトリガー条件が真となり、それ以外の場合は偽となります。
 - OR: 少なくとも 1 つのプローブ コンパレータが真の場合にトリガー条件が真となり、それ以外の場合は偽となります。
 - NOR: すべてのプローブ コンパレータが偽の場合にトリガー条件が真となり、それ以外の場合は偽となります。

注記: トリガー条件の評価に使用するプローブは、hw_probe オブジェクト (get_hw_probes コマンドで指定) の TRIGGER_COMPARE_VALUE プロパティで判断されます。TRIGGER_COMPARE_VALUE が X の場合、トリガー条件の評価には使用されません。

- CONTROL.TSM_FILE: アドバンス トリガー処理に使用するトリガー有限ステート マシン (TSM) を定義するファイルへのパスを指定します。
- CONTROL.TRIG_OUT_MODE: ほかの ILA コア上の TRIG_IN ポートを駆動するのに使用する ILA コアの TRIG_OUT ポートを遷移します。有効な値は、次のとおりです。
 - DISABLED: ILA コアの TRIG_OUT ポートをディスエーブルにします。
 - TRIGGER_ONLY: トリガー条件が満たされたときに TRIG_OUT ポートを遷移します。
 - TRIG_IN_ONLY: TRIG_IN 信号が遷移したときに TRIG_OUT ポートを遷移します。ILA コアのチェーンでトリガー イベントを伝搬する際に使用します。
 - TRIGGER_OR_TRIG_IN: トリガー条件が満たされたとき、または TRIG_IN が遷移したときに TRIG_OUT ポートを遷移します。
- CONTROL.CAPTURE_MODE: 有効な値は ALWAYS および BASIC です。ALWAYS に設定すると、デバッグ コアのデータ サンプルが常にキャプチャおよび格納されます。BASIC に設定すると、CAPTURE_CONDITION が真の場合にのみデータ サンプルがキャプチャおよび格納されます。
- CONTROL.CAPTURE_CONDITION: ILA デバッグ コアのデータ キャプチャ条件を満たすために真と評価される必要のあるプローブ コンパレータのブール式を定義します。キャプチャ条件が真の場合、BASIC トリガー モードが使用されます。有効な値は、次のとおりです。
 - AND: すべてのプローブ コンパレータが真の場合にキャプチャ条件が真となり、それ以外の場合は偽となります。
 - NAND: 少なくとも 1 つのプローブ コンパレータが偽の場合にキャプチャ条件が真となり、それ以外の場合は偽となります。
 - OR: 少なくとも 1 つのプローブ コンパレータが真の場合にキャプチャ条件が真となり、それ以外の場合は偽となります。
 - NOR: すべてのプローブ コンパレータが偽の場合にキャプチャ条件が真となり、それ以外の場合は偽となります。

注記: キャプチャ条件の評価に使用するプローブは、hw_probe オブジェクト (get_hw_probes コマンドで指定) の CAPTURE_COMPARE_VALUE で判断されます。CAPTURE_COMPARE_VALUE が X の場合、キャプチャ条件の評価には使用されません。



ヒント: ILA コアにはコアの動作を決定するプロパティがほかにもありますが、ユーザーが設定することはできません。

ILA コアを設定したら、run_hw_ila コマンドを使用してターゲット パーツ上の ILA コアをトリガー待機状態にできます。このコマンドを実行すると、hw_ila および hw_probe オブジェクトで定義されているトリガー設定がターゲット ザイリンクス FPGA (hw_device) に記述され、デバイス上の ILA コアがトリガー待機状態になります。

hw_ila がトリガー待機状態で実行中の場合、wait_on_hw_ila コマンドを使用してデータ サンプル バッファにキャプチャされたデータが挿入されるまで Tcl スクリプトを停止できます。物理 hw_device 上の ILA コアのメモリがフルになったら、wait_on_hw_ila コマンドから戻り、Tcl スクリプトが再開します。

キャプチャされたデータを hw_device の物理メモリから Vivado ロジック解析機能の hw_ila_data オブジェクトにアップロードするには、upload_hw_ila_data コマンドを使用します。ILA データは、display_hw_ila_data コマンドを使用して Vivado ロジック解析機能の波形ウィンドウに表示でき、write_hw_ila_data コマンドを使用して外部ツール用にデータを保存できます。

トリガー イベントまたはキャプチャ条件が発生するまで待つのではなく、-trigger_now オプションを使用して hw_device 上のプローブを直ちにトリガーしてデバイスからデータをキャプチャすることもできます。

ILA デバッグ コアの CONTROL プロパティを元に戻し、プローブ コンパレータ値を X にリセットするには、reset_hw_ila を使用します。

引数

-trigger_now (オプション): トリガー条件にかかわらず、ILA デバッグ コアを直ちにトリガーし、デバッグ ポートのサンプル データをキャプチャします。これはブール値のオプションであり、使用するとイネーブルになります。

-compile_only (オプション): トリガー ステート マシン ファイル (CONTROL.TSM_FILE) をコンパイルし、構文チェックを実行してコンパイル エラーおよび警告をレポートします。このオプションでは、ハードウェア デバイスにファイルは読み込まれず、トリガー待機状態にはなりません。このオプションは、アドバンス トリガー モード (CONTROL.TRIGGER_MODE ADVANCED_ONLY) でのみサポートされ、それ以外の場合はエラーが返されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

-file <arg> (オプション): TRIGGER_MODE を BASIC_ONLY に設定した場合に、現在のトリガーおよびプローブ設定を含むファイルをエクスポートします。トリガー ファイルは、apply_hw_ila_trigger コマンドでスタートアップまたはデバイスのブート時に使用する hw_device のトリガーを定義する際に使用されます。-file オプションを使用すると、run_hw_ila コマンドでハードウェア デバイス上の ILA コアがトリガー待機状態になりません。

注記: 拡張子を指定しない場合、デフォルトの拡張子 .tas が使用されます。

-force (オプション): 指定のトリガー ファイルが存在する場合に上書きします。デフォルトでは、既存のファイルは上書きされません。

<hw_ilas> (オプション): 実行する hw_ila オブジェクトを 1 つ以上指定します。hw_ila は、get_hw_ilas または current_hw_ila コマンドを使用してオブジェクトとして指定するか、名前で指定できます。hw_ila オブジェクトを指定しない場合、現在のハードウェア ILA (current_hw_ila) が実行されます。



重要: -file オプションを使用する場合は、hw_ila オブジェクトを 1 つのみ指定できます。

例

次の例では、現在の hw_ila オブジェクトのトリガーおよびキャプチャ プロパティを設定し、指定した名前の hw_ila オブジェクトを実行して、hw_device をトリガー待機状態にしてバッファにデータが挿入されるのを待ちます。

```
current_hw_ila [get_hw_ilas hw_ila_1]
set_property CONTROL.DATA_DEPTH 1024 [current_hw_ila]
set_property CONTROL.TRIGGER_MODE BASIC_ONLY [current_hw_ila]
set_property CONTROL.TRIGGER_CONDITION AND [current_hw_ila]
set_property CONTROL.CAPTURE_MODE ALWAYS [current_hw_ila]
run_hw_ila -trigger_now hw_ila_1
wait_on_hw_ila hw_ila_1
```

次の例では、現在の hw_ila オブジェクトのトリガー構成ファイルを記述し、指定のファイルに保存しますが、現在の hw_device をトリガー待機状態にはしません。

```
run_hw_ila -file C:/Data/trigger_config1
```

注記: 指定のファイル名にデフォルトの拡張子 .tas が使用されます。

関連項目

- [apply_hw_ila_trigger](#)
- [current_hw_device](#)
- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [get_hw_probes](#)
- [reset_hw_ila](#)
- [set_property](#)
- [upload_hw_ila_data](#)
- [wait_on_hw_ila](#)
- [write_debug_probes](#)
- [write_hw_ila_data](#)

run_hw_sio_scan

ハードウェア SIO スキャンを実行します。

構文

```
run_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_scans></code>	ハードウェア SIO スキャン

カテゴリ

Hardware (ハードウェア)

説明

指定したシリアル I/O 解析リンク スキャン オブジェクトを実行します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用のアイ スキャン ハードウェアを使用してリンクのスキャンを実行すると有益です。Vivado シリアル I/O 解析機能では、リンク スキャンを作成、実行、および保存できます。

このコマンドは、`run_hw_sio_scan` コマンドで指定のリンクまたは GT レシーバーに対して解析を実行するのに使用可能なリンク スキャン オブジェクトを作成し、返します。`write_hw_sio_scan` コマンドを使用して、スキャンをディスクに保存することもできます。

指定のスキャン オブジェクトに対して解析を実行します。Tcl スクリプトで実行する場合、`wait_on_hw_sio_scan` コマンドを使用して、スキャンを完了するまでスクリプトを一時停止できます。実行中のスキャンを停止するには、`stop_hw_sio_scan` コマンドを使用します。

スキャンをディスクに保存するには、`write_hw_sio_scan` コマンドを使用します。

作成したスキャン オブジェクトを削除するには、`remove_hw_sio_scan` コマンドを使用します。

このコマンドを実行すると、作成したハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_scans>` (必須): 実行する `hw_sio_scan` オブジェクトを 1 つ以上指定します。`hw_sio_scans` は、`create_hw_sio_scan` または `get_hw_sio_scans` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したシリアル I/O 解析スキャンを実行しています。

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_3}] 0]
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

run_hw_sio_sweep

ハードウェア SIO スイープを実行します。

構文

```
run_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_sweeps></code>	ハードウェア SIO スイープ

カテゴリ

Hardware (ハードウェア)

説明

ある値の範囲で複数のスキャンを実行するシリアル I/O 解析リンク スイープ スキャンを実行します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用の機能を使用してリンクのスキャンを実行すると有益です。また、リンクに対して GT の異なる設定を使用して複数のスキャンを実行するのも有益です。デザインにどの設定が最適かを判断するのに役立ちます。Vivado シリアル I/O 解析機能では、リンク スイープ (ある値の範囲で実行するリンク スキャンのグループ) を定義、実行、および保存できます。

指定のスイープ スキャン オブジェクトに対して解析を実行します。Tcl スクリプトで実行する場合、`wait_on_hw_sio_sweep` コマンドを使用して、スイープを完了するまでスクリプトを一時停止できます。実行中のスイープを停止するには、`stop_hw_sio_sweep` コマンドを使用します。

`write_hw_sio_sweep` コマンドを使用すると、スイープをディスクに保存できます。

作成したスキャン オブジェクトを削除するには、`remove_hw_sio_sweep` コマンドを使用します。

このコマンドを実行すると、作成したハードウェア SIO スイープ (`hw_sio_sweep`) オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_sio_sweeps> (必須): 実行する hw_sio_sweep オブジェクトを 1 つ以上指定します。hw_sio_sweep は、`get_hw_sio_sweeps` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定したスイープ スキャンを実行しています。

```
run_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

関連項目

- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

run_state_hw_jtag

指定した遷移の安定ステートに変更します。

構文

```
run_state_hw_jtag [-state <args>] [-quiet] [-verbose] <stable_state>
```

戻り値

ハードウェア JTAG

使用法

名前	説明
[-state]	安定ステートに遷移する際に通過する有効なステート パス シーケンスを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<stable_state>	有効な安定ステートを指定します。有効な安定ステートは、IDLE、RESET、IRPAUSE、および DRPAUSE です。

カテゴリ

[Hardware](#) (ハードウェア)、[Object](#) (オブジェクト)

説明

現在のハードウェア ターゲットのハードウェア JTAG (hw_jtag) オブジェクトを指定の TAP 安定ステートに遷移します。

hw_jtag オブジェクトは、open_hw_target -jtag_mode コマンドを使用してハードウェア ターゲットを JTAG モードで開いたときに、Vivado Design Suite のハードウェア マネージャーにより作成されます。

run_state_hw_jtag コマンドは、次を指定します。

- 遷移先のターゲット (最終) TAP 安定ステート。
- 現在のステートからターゲット ステートに遷移するまでに通過するステート パス リスト (オプション)。

オプションの -state オプションを使用してパス リストを定義する場合は、ステート リストには安定ステートに達するまでに通過するすべてのステートを含める必要があります。そうでないと、エラーが返されます。ステート パス リストを定義しない場合は、次の表で定義されたステート 遷移パスに基づいて現在のステートからターゲット ステートに遷移します。

Current State	Target State	State Transition Path
DRPAUSE	RESET	DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT -> IRSELECT-> RESET
DRPAUSE	IDLE	DRPAUSE -> DREXIT2 -> DRUPDATE -> IDLE
DRPAUSE	DRPAUSE	DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT ->

DRPAUSE	IRPAUSE	DRCAPTURE -> DREXIT1 -> DRPAUSE
IDLE	RESET	DRPAUSE -> DREXIT2 -> DRUPDATE -> DRSELECT ->
IDLE	IDLE	IRSELECT -> IRCAPTURE -> IREXIT12 -> IRPAUSE
IDLE	DRPAUSE	IDLE -> DRSELECT -> IRSELECT -> RESET
DRPAUSE		IDLE
IDLE	IRPAUSE	IDLE -> DRSELECT -> DRCAPTURE -> DREXIT1 ->
		IDLE -> DRPAUSE -> IRSELECT -> IRCAPTURE ->
IRPAUSE	RESET	IREXIT1 -> IRPAUSE
		IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
IRPAUSE	IDLE	IRSELECT -> RESET
IRPAUSE	DRPAUSE	IRPAUSE -> IREXIT2 -> IRUPDATE -> IDLE
		IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
IRPAUSE	IRPAUSE	DRCAPTURE -> DREXIT1 -> DRPAUSE
		IRPAUSE -> IREXIT2 -> IRUPDATE -> DRSELECT ->
RESET	RESET	IRSELECT -> IRCAPTURE -> IREXIT1 -> IRPAUSE
RESET	IDLE	RESET
RESET	DRPAUSE	RESET -> IDLE
		RESET -> IDLE -> DRSELECT -> DRCAPTURE ->
RESET	IRPAUSE	DREXIT1 -> DRPAUSE
		RESET -> IDLE -> DRSELECT -> IRSELECT ->
		IRCAPTURE -> IREXIT1 -> IRPAUSE

このコマンドを実行すると、正常に実行された場合はターゲット安定ステートが返され、正常に実行されなかった場合はエラーが返されます。

引数

-state <args> (オプション): hw_jtag オブジェクトを現在のステートからターゲット安定ステート (<stable_state>) に遷移するまでに通過するステートの有効なパス シーケンスを指定します。有効なステートは、次のとおりです。

- IDLE
- RESET
- DRSELECT、DRCAPTURE、DRSHIFT、DRPAUSE、DREXIT1、DREXIT2、DRUPDATE
- IRSELECT、IRCAPTURE、IRSHIFT、IRPAUSE、IREXIT1、IREXIT2、IRUPDATE

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<stable_state> (必須): 有効なターゲット安定ステート (最終ステート) を指定します。有効なターゲット ステートは、次のとおりです。

- IDLE
- RESET
- DRPAUSE
- IRPAUSE

例

次の例では、さまざま TAP 安定ステートに遷移しています。

```
// Go to state RESET
run_state_hw_jtag RESET

// From current state RESET, go to DRPAUSE
run_state_hw_jtag DRPAUSE

// From DRPAUSE, go to IDLE state transitioning through
// the specified states
run_state_hw_jtag -state {DREXIT2 DRUPDATE IDLE} IDLE

// From IDLE, go to RESET, through the specified states
// note that specified path starts with an extra TCK
// clock cycle in the IDLE state
run_state_hw_jtag RESET -state {IDLE DRSELECT IRSELECT RESET}
```

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [open_hw_target](#)
- [runtest_hw_jtag](#)
- [scan_dr_hw_jtag](#)
- [scan_ir_hw_jtag](#)

runtest_hw_jtag

IEEE 1149.1 TAP ステート マシンを指定した待機期間安定ステートに強制します。

構文

```
runtest_hw_jtag [-wait_state <arg>] [-end_state <arg>] [-sec <arg>]
                [-max_wait <arg>] [-tck <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-wait_state]	有効な安定ステートを指定します。有効な安定ステートは、IDLE、RESET、IRPAUSE、および DRPAUSE です。
[-end_state]	有効な安定ステートを指定します。有効な安定ステートは、IDLE、RESET、IRPAUSE、および DRPAUSE です。
[-sec]	待機ステートにとどまる時間を秒で指定します。
[-max_wait]	待機ステートにとどまる最大時間を秒で指定します (最大タイムアウト)。
[-tck]	待機ステートにとどまる時間を TCK サイクル数で指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Hardware (ハードウェア)、Object (オブジェクト)

説明

ハードウェア JTAG (hw_jtag) オブジェクトのステート マシンの待機操作を指定します。次を定義します。

- 待機操作を実行する TAP 安定ステート。
- 次で指定した待機時間:
 - TCK の n サイクル (n は 32 ビットの符号なし 10 進数値)。
 - 秒で指定した待機ステートにとどまる最小時間、およびオプションで最大時間 (符号なしの整数または実数値)。
- 待機操作が完了した後に遷移する TAP 安定ステート。

-wait_state および -end_state オプションのデフォルト値は IDLE です。IDLE 以外の待機ステートまたは最終ステートが指定された場合、hw_jtag オブジェクトは待機操作を開始する前にまず指定の待機ステートに遷移します。待機時間が経過すると、hw_jtag オブジェクトは指定の最終ステートに遷移します。runtest_hw_jtag コマンドで待機ステートまたは最終ステートを指定すると、変更しない限り、その後のコマンドでも同じ待機ステートと最終ステートが使用されます。

このコマンドを実行すると、最終安定ステートが返され、正常に実行されなかった場合はエラーが返されます。

注記: 指定の待機時間を満たすことができない場合は、例外が発行されます。この例外は、Tcl の catch コマンドで取得できます。

引数

`-wait_state <arg>` (オプション): 待機操作を実行する際に遷移するステートを指定します。TAP 安定ステート IDLE、RESET、IRPAUSE、および DRPAUSE のいずれかを指定できます。デフォルトは IDLE です。

`-end_state <arg>` (オプション): 待機操作を完了した後に遷移するステートを指定します。TAP 安定ステート IDLE、RESET、IRPAUSE、および DRPAUSE のいずれかを指定できます。デフォルトは IDLE です。

`-sec <arg>` (オプション): 待機する最小時間 (秒) を 32 ビットの 10 進整数値で指定します。

`-max_wait <arg>` (オプション): 待機ステートにとどまる最大時間 (秒) を指定します。

`-tck <arg>` (オプション): 待機する JTAG クロック サイクル数を 32 ビットの 10 進整数値で指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、さまざまな待機ステートおよび最終ステートを指定して `runtest_hw_jtag` コマンドを複数回実行しています。

```
// Wait in default IDLE state for 1000 TCKs,
// then go to end_state DRPAUSE
runtest_hw_jtag -tck 1000 -end_state DRPAUSE

// Go from DRPAUSE (end_state defined in previous
// runtest_hw_jtag command) to IDLE and wait for
// 500 TCK clock cycles before going to DRPAUSE again
runtest_hw_jtag -tck 500

// Go from DRPAUSE to IDLE and wait for
// 1,000,000 TCKs or at least
// 5 seconds before transitioning to DRPAUSE
runtest_hw_jtag -tck 1000000 -sec 5

// Go from DRPAUSE to IDLE and wait for
// at least 1 millisecond and at most 50 milliseconds
// before remaining in IDLE state
runtest_hw_jtag -sec 1.0E-3 -max_wait 50.0E-3 -end_state IDLE

// Go from IDLE to DRPAUSE and wait for at least
// 85 milliseconds before returning to IDLE state
runtest_hw_jtag -wait_state DRPAUSE -sec 85E-3

// Go from IDLE to DRPAUSE state and wait for
// at least 1 second before returning to IDLE state
```

```
runtest_hw_jtag -sec 1

// Go to wait_state IDLE (note: current end_state is IDLE),
// wait for at least 10 milliseconds, then stay in IDLE state.
runtest_hw_jtag -wait_state IDLE -sec 1E-2
```



ヒント: 指定して変更しない限り、最初の `runtest_hw_jtag` コマンドからの待機ステートまたは最終ステートがその後のコマンドでも使用されます。

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [open_hw_target](#)
- [run_state_hw_jtag](#)
- [scan_dr_hw_jtag](#)
- [scan_ir_hw_jtag](#)

save_bd_design

既存の IP サブシステム デザインをディスク ファイルに保存します。

構文

```
save_bd_design [-quiet] [-verbose] [<name>]
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	保存するデザインの名前を指定します。

カテゴリ

[IPIntegrator \(IP インテグレーター\)](#)

説明

Vivado Design Suite の IP インテグレーターの現在の IP サブシステムまたは指定した IP サブシステムに変更を保存します。

このコマンドが正常に実行された場合は TCL_OK が返され、正常に実行されなかった場合は TCL_ERROR が返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<name>: 保存する IP サブシステム デザインの名前を指定します。名前を指定しない場合は、現在の IP サブシステム デザインが保存されます。

例

次の例では、現在のプロジェクトの現在の IP サブシステム デザインを保存しています。

```
save_bd_design
```

関連項目

- [close_bd_design](#)
- [create_bd_design](#)
- [current_bd_design](#)
- [get_bd_designs](#)
- [open_bd_design](#)

save_bd_design_as

既存の IP サブシステム デザインを別の名前を指定してファイルに保存します。生成された出力ファイルは保存されません。

構文

```
save_bd_design_as [-dir <arg>] [-ignore_comments] [-force] [-quiet]  
                  [-verbose] [<name>]
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-dir]	作成し、管理するリモート BD へのディレクトリ パスを指定します。名前を指定しない場合、このオプションは必須です。
[-ignore_comments]	ユーザー コメントを保存しません。
[-force]	指定した名前のファイルが既に存在する場合に上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<name>]	デザインの名前を指定します。ディレクトリを指定しない場合、このオプションは必須です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

IP インテグレーターからの既存のブロック デザインのコピーを別の名前で保存します。ブロック デザインの出力ファイルは、新しいブロック デザインには保存されません。

注記: ロックされている IP を含むブロック デザインのコピーを作成することはできません。IP がロックされている場合、このコマンドでエラーが返されます。

引数

```
save_bd_design_as  [-dir <arg>] [-ignore_comments] [-force] [-quiet] [-  
verbose]  
                  [<name>]
```

-dir <arg> (オプション): 新しいブロック デザイン ファイルを保存するディレクトリを指定します。ブロック デザインは、指定したディレクトリのブロック デザインと同じ名前のフォルダーに保存されます。-dir オプションのみを指定した場合は、ブロック デザインは新しいディレクトリに現在の名前でコピーされます。



重要: `-dir` の指定はオプションですが、`-dir` または `<name>` のいずれかを指定する必要があります。

`-ignore_comments` (オプション): 現在のブロック デザインのユーザー定義のコメントはコピーしません。

`-force` (オプション): 同じ名前のブロック デザインが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>`: 保存するブロック デザインの名前を指定します。`<name>` を指定しない場合は、`-dir` オプションを指定する必要があります。`-dir` オプションを指定せずに `<name>` のみを指定すると、ブロック デザインは現在のディレクトリに新しい名前で作成されます。

例

次の例では、現在のブロック デザインを指定したディレクトリに指定した名前で保存しています。

```
save_bd_design_as -dir C:/Data new_Block
```

関連項目

- [close_bd_design](#)
- [create_bd_design](#)
- [current_bd_design](#)
- [get_bd_designs](#)
- [open_bd_design](#)
- [save_bd_design](#)

save_constraints

現在のデザインの制約を保存します。

構文

```
save_constraints [-force] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-force]</code>	制約を強制的に保存します。必要であれば、ターゲットおよびソース XDC を上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

アクティブな制約セットの制約ファイルに変更を保存します。このコマンドにより、制約ファイルへのすべての変更がハード ドライブのプロジェクト データに書き込まれ、作業中の変更がすべて保存されます。

引数

`-force` (オプション): 変更されているかどうかにかかわらず、アクティブな制約ファイルを保存します。現在のターゲット制約ファイルが上書きされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ファイルが変更されているかどうかにかかわらず、アクティブな制約セットの制約ファイルが保存されます。

```
save_constraints -force
```

関連項目

- [save_constraints_as](#)

save_constraints_as

現在のデザインの制約を新しい制約ファイルのセットとして保存します。

構文

```
save_constraints_as [-dir <arg>] [-target_constrs_file <arg>] [-quiet]
                  [-verbose] <name>
```

使用法

名前	説明
[-dir]	制約を保存するディレクトリを指定します。
[-target_constrs_file]	新しいファイルセットのターゲット制約ファイルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	新しい制約ファイルセットの名前を指定します。

カテゴリ

Project (プロジェクト)

説明

アクティブな制約セットとその制約セットに含まれる制約ファイルのローカル コピーを、新しい制約セットにコピーします。コピーした制約セットのターゲットとして使用する制約ファイルも指定できます。

このコマンドは、現在の制約ファイルを変更せずに、デザインの制約への変更を保存するために使用します。これにより、さまざまな条件のデザイン制約を試すことができます。

注記: `save_constraints_as` コマンドで作成された新しい制約セットは、参照されますが、アクティブにはなりません。この制約セットをアクティブにするには、特定の run で `constset` プロパティを新しい制約セットに設定する必要があります。例を参照してください。

引数

`-dir <arg>` (オプション): 制約ファイルを保存するディレクトリを指定します。ディレクトリを指定しない場合、新しい制約セットはプロジェクト ソース ディレクトリに保存されます。アクティブな制約セットからの制約ファイルは、指定したディレクトリにコピーされます。

`-target_constrs_file <arg>` (オプション): 新しい制約ファイルセットのターゲット制約ファイルを指定します。パスをファイル名の一部として指定しない場合は、ファイルセットのディレクトリにファイルが作成されます。

注記: 拡張子 `.xdc` を指定する必要があります。指定しないと、ファイル タイプが無効であり、ターゲット制約ファイルとして設定できないという警告メッセージが表示されます。この場合、既存のターゲット制約ファイルがターゲットとして使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 新しく作成する制約セットの名前を指定します。

例

次の例では、アクティブな制約セットを `constrs_2` という新しい制約セットに保存し、制約ファイルを指定したディレクトリにコピーし、新しい制約セットのターゲット制約ファイルを指定しています。

```
save_constraints_as -dir C:/Data/con1 \
  -target_constrs_file rev1.xdc constrs_2
```

次の例では、アクティブな制約セットを `newCon2` という新しい制約セットに保存し、制約ファイルをプロジェクトソースの下の `newCon2` 制約ディレクトリにコピーして、指定した合成 run およびインプリメンテーション run の `constrset` プロパティを新しい制約セットに設定しています。

```
save_constraints_as newCon2
set_property CONSTRSET newCon2 [get_runs synth_1]
set_property CONSTRSET newCon2 [get_runs impl_1]
```

注記: 制約セットは、現在の run に対してアクティブに設定されない限り、アクティブにはなりません。

関連項目

- [save_constraints](#)

save_project_as

現在のプロジェクトを新しい名前で保存します。

構文

```
save_project_as [-scan_for_includes] [-exclude_run_results]
                [-include_local_ip_cache] [-force] [-quiet] [-verbose] <name> [<dir>]
```

戻り値

保存されたプロジェクト オブジェクト

使用法

名前	説明
<code>[-scan_for_includes]</code>	インクルード ファイルをスキャンし、新しいプロジェクトに追加します。
<code>[-exclude_run_results]</code>	新しいプロジェクトに run の結果を含めません。
<code>[-include_local_ip_cache]</code>	新しいプロジェクトに IP キャッシュの結果を含めます。
<code>[-force]</code>	既存のプロジェクト ディレクトリを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	保存するプロジェクトの新しい名前を指定します。
<code>[<dir>]</code>	プロジェクト ファイルを保存するディレクトリを指定します。デフォルトは . です。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在開いているプロジェクト ファイルを別の名前で指定のディレクトリまたはディレクトリを 指定しない場合は現在の作業ディレクトリに保存します。

Vivado Design Suite プロジェクト ファイル (.xpr) または Vivado Lab Edition 用のプロジェクト ファイル (.lpr) を指定したディレクトリに保存します。

このコマンドを実行すると、保存されたプロジェクトの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-scan_for_includes` (オプション): すべてのソース ファイルをスキャンし、参照されている Verilog の `'include` ファイルをプロジェクト構造にインポートします。

`-exclude_run_results` (オプション): 新しいプロジェクトに run の結果を含めません。プロジェクトから現在の run が削除され、プロジェクト ソースのコピーが追加のデータなしで作成されます。

`-include_local_ip_cache` (オプション): 新しいプロジェクトに IP キャッシュの結果を含めます。これにより、アウト オブ コンテキスト合成キャッシュを新しいプロジェクトに保持できます。

`-force` (オプション): 既存のプロジェクトを上書きします。指定したディレクトリに指定のプロジェクト名が既に存在する場合、`-force` オプションを使用して既存のプロジェクトを上書きする必要があります。

注記: 既存プロジェクトを開いている場合、新しいプロジェクトでディスクの既存プロジェクトが上書きされますが、両方のプロジェクトがツールで開いたままになります。この場合、`create_project` を実行する前に `close_project` コマンドを実行しておくことをお勧めします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): 新しいプロジェクトの名前を指定します。この引数はディレクトリの前に指定します。このコマンドにはパラメーターがないので、最初の引数は `<name>`、2 つ目の引数は `<dir>` として認識されます。プロジェクト ファイルは、`<dir>` で指定したディレクトリに、Vivado Design Suite では `<name> .xpr` という名前、Vivado Lab Edition では `<name> .lpr` という名前で保存されます。

`<dir>` (オプション): 新しいプロジェクト ファイルを保存するディレクトリ名を指定します。指定したディレクトリが存在しない場合は、その名前の新しいディレクトリが作成されます。ディレクトリを完全なパスで指定すると、その指定したパス名が使用されます。`<dir>` をパスなしで指定すると、現在の作業ディレクトリまたはツールの起動ディレクトリでそのディレクトリが検索されるか、作成されます。

注記: プロジェクトを GUI モードで作成すると、ディレクトリ名 `<dir>` にファイル名 `<name>` を付けた `<dir>/<name>` という名前のプロジェクト ディレクトリが作成され、新しいプロジェクト ファイルとプロジェクト データ フォルダが保存されます。

例

次の例では、アクティブ プロジェクトが `myProjectDir` というディレクトリに `myProject` という新しいプロジェクト名で保存されます。

```
save_project_as myProject myProjectDir
```

注記: `<dir>` にはフォルダー名のみが指定されているので、プロジェクトは現在の作業ディレクトリかツールの起動ディレクトリに作成されます。

次の例では、現在のプロジェクトが `C:/Designs/myProjectDir` というディレクトリに `myProject` という新しいプロジェクト名で保存されます。`-force` オプションを使用すると、指定したディレクトリに同名のプロジェクトがある場合はそれが上書きされます。

```
save_project_as myProject C:/Designs/myProjectDir -force
```

関連項目

- [create_project](#)
- [current_project](#)

- [open_project](#)

save_wave_config

指定または現在の波形設定オブジェクトを指定のファイル名で保存します。

構文

```
save_wave_config [-object <args>] [-quiet] [-verbose] [<filename>]
```

戻り値

保存された波形設定オブジェクト

使用法

名前	説明
[-object]	保存する波形設定 (WCFG) を指定します。デフォルトは、現在の波形設定です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<filename>]	指定または現在の波形設定オブジェクトを保存するファイルの名前を指定します。

カテゴリ

Waveform (波形)

説明

指定または現在の波形設定オブジェクトを指定のファイル名で保存します。

波形設定オブジェクトが保存されておらず、FILE_PATH プロパティ 値がない場合はファイル名を指定する必要があります。波形設定オブジェクトの NAME プロパティが指定した名前に変更されます。

指定した波形設定オブジェクトが以前に保存されており、FILE_PATH プロパティがある場合は、オブジェクトは現在の場所に記述され、ファイル名を指定する必要はありません。

FILE_PATH プロパティがある波形設定オブジェクトにファイル名を指定した場合、波形設定オブジェクトは新しいファイル名に保存され、オブジェクト名がその名前に変更されます。

引数

-object <arg> (オプション): 保存する波形設定オブジェクトを指定します。波形設定オブジェクトがファイルに保存されていない場合は、<filename> を指定する必要があります。-object を指定しない場合は、現在の波形設定オブジェクト (current_wave_config で返される) が指定の <filename> で保存されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<filename>` (オプション): 指定または現在の開く波形設定オブジェクトのパスとファイル名を指定します。ファイル名 (`<filename>`) の拡張子は `.wcfg` である必要があり、拡張子を付けない場合は自動的に付けられます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定の波形設定オブジェクトを新しいファイル名で保存しています。

```
save_wave_config -object [get_wave_configs test.wcfg] \
C:/Data/project/newTest
```

注記: 拡張子が指定されていないので、波形設定ファイルに `.wcfg` が拡張子として付けられます。

関連項目

- [create_wave_config](#)
- [current_wave_config](#)
- [get_wave_configs](#)
- [open_wave_config](#)

scan_dr_hw_jtag

ハードウェア JTAG で Shift-DR を実行します。

構文

```
scan_dr_hw_jtag [-tdi <arg>] [-tdo <arg>] [-mask <arg>] [-smask <arg>]
                [-quiet] [-verbose] <length>
```

戻り値

ハードウェア TDO

使用法

名前	説明
[-tdi]	ターゲットに挿入する 16 進数を指定します。
[-tdo]	スキャンされた値と比較する 16 進数を指定します。
[-mask]	TDO 値を比較する際に適用する 16 進数のマスクを指定します。
[-smask]	TDI 値に適用する 16 進数のマスクを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<length>	スキャンするビット数を指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

scan_dr_hw_jtag コマンドは、JTAG インターフェイス ターゲットのデータ レジスタに挿入するスキャン パターンを指定します。

このコマンドは、open_hw_target -jtag_mode コマンドを使用してハードウェア ターゲット (hw_target) を JTAG モードで開いたときに作成されたハードウェア JTAG (hw_jtag) オブジェクトをターゲットとします。

scan_dr_hw_jtag コマンドで指定したスキャン パターンをシフトする前に hw_jtag オブジェクトをターゲットとすると、最後に定義されたヘッダー プロパティ (HDR) が指定したデータ パターンの先頭に追加され、最後に定義されたトレーラー プロパティ (TDR) がデータ パターンの末尾に追加されます。

オプションはどの順序で指定してもかまいませんが、1 回のみ指定可能です。-tdi、-tdo、-mask、または -smask オプションで指定する 16 進数文字列で表されたビット数は、<length> で指定した最大値以下にする必要があります。16 進数文字列で表されたビット数が <length> よりも小さい場合は、16 進数文字列の先頭は 0 であると想定されます。

データ ビットをターゲット データ レジスタにシフトする際、次のステート遷移の表に従って、`scan_dr_hw_jtag` コマンドにより JTAG TAP が現在の安定ステートから DRSHIFT ステートに移動されます。

Current State	Transitions to get to DRSHIFT state
RESET	IDLE -> DRSELECT -> DRCAPTURE -> DRSHIFT
IDLE	DRSELECT -> DRCAPTURE ->
DRSHIFT	
IRPAUSE	IREXIT2 -> IRUPDATE -> DRSELECT -> DRCAPTURE ->
DRSHIFT	
DRPAUSE	DREXIT2 ->
DRSHIFT	
DRPAUSE*	DREXIT2 -> DRUPDATE -> DRSELECT -> DRCAPTURE -> DRSHIFT

注記: * `-force_update` オプションが設定されている場合。

最後のビットがターゲット データ レジスタにシフトされた後、`scan_dr_hw_jtag` コマンドで JTAG TAP が IDLE ステートまたは `run_state_hw_jtag` コマンドで定義された安定ステートに移動されます。

`scan_dr_hw_jtag` コマンドを実行すると、`hw_jtag` からキャプチャされた TDO データを含む 16 進配列が返されるか、正常に実行されなかった場合はエラーが返されます。

`hw_jtag` からの TDO データが `-tdo` オプションで指定した値に一致しない場合はエラーが返され、そのエラーは Tcl の `catch` コマンドでキャプチャできます。



ヒント: `-tdo` および `-mask` オプションを指定した場合、2 つを比較する前に `-tdo` オプションと返された `hw_jtag` TDO データにマスクが適用されます。

引数

`-tdi <arg>` (オプション): ターゲットに挿入する値を 16 進数値で指定します。このオプションを指定しない場合、最後の `scan_dr_hw_jtag` コマンドからの `-tdi` 値が使用されます。`-tdi` オプションは、`scan_dr_hw_jtag` コマンドを初めて実行する場合、および `<length>` を変更した場合には必ず指定する必要があります。

`-tdo <arg>` (オプション): `hw_jtag` データ レジスタからスキャンされた実際の TDO 値と比較するデータ値を 16 進文字列で指定します。このオプションを指定しない場合、比較は実行されません。`-tdo` オプションを指定しない場合、`-mask` オプションは無視されます。

`-mask <arg>` (オプション): `hw_jtag` からスキャンされた実際の TDO 値と `-tdo` で指定した値を比較する際に適用するマスクを指定します。特定のビット位置が 1 の場合はそのビット値が比較され、0 の場合はそのビット値は比較されません。`-mask` オプションを指定しない場合、最後の `scan_dr_hw_jtag` コマンドからの `-mask` 値が使用されます。



重要: `<length>` を変更して `-mask` オプションを指定しない場合、使用される `-mask` パターンはすべて 1 です。

`-smask <arg>` (オプション): `-tdi` データに適用するマスクを指定します。特定のビット位置が 1 の場合はそのビット位置の TDI データが重要であることを示し、0 の場合は重要でないことを示します。`-smask` オプションは `-tdi` オプションを指定していない場合でも適用されます。`-smask` オプションを指定しない場合、最後の `scan_dr_hw_jtag` コマンドからの `-smask` 値が使用されます。



重要: `<length>` を変更して `-smask` オプションを指定しない場合、使用される `-smask` パターンはすべて 1 です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<length>` (必須): データ レジスタからスキャンするビット数を指定します。0 より大きい値を 32 ビット符号なし 10 進数整数で指定します。

例

次の例では、JTAG データ レジスタから 24 ビット値をスキャンしています。

```
scan_dr_hw_jtag 24
```

次の例では、24 ビット値 0x00_0010 (LSB が先頭) を TDI に送信し、データ出力 TDO をキャプチャして、0xF3_FFFF のマスクを適用して返された TDO 値を `-tdo` で指定した値 0x81_8181 と比較しています。

```
scan_dr_hw_jtag 24 -tdi 000010 -tdo 818181 -mask F3FFFF -smask 0
```

次の例では、指定の TDI 値 0x3f の先頭に 0x が追加されます。

```
scan_dr_hw_jtag 24 -tdi 3f
```

長いデータ レジスタシフトを複数の SDR シフトに分割するには、最終ステートを DRPAUSE に設定します。これにより、最初の `scan_dr_hw_jtag` コマンドが DRPAUSE 安定ステートで終了し、次の `scan_dr_hw_jtag` コマンドが DRSHIFT に戻る前に DREXIT2 ステートになります。

```
run_state_hw_jtag DRPAUSE
scan_dr_hw_jtag 16 -tdi aabb
scan_dr_hw_jtag 16 -tdi ccdd
scan_dr_hw_jtag 16 -tdi ceff
scan_ir_hw_jtag 6 -tdi 05
```

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [open_hw_target](#)
- [run_state_hw_jtag](#)
- [runttest_hw_jtag](#)
- [scan_ir_hw_jtag](#)

scan_ir_hw_jtag

ハードウェア JTAG で Shift-IR を実行します。

構文

```
scan_ir_hw_jtag [-tdi <arg>] [-tdo <arg>] [-mask <arg>] [-smask <arg>]
                [-quiet] [-verbose] <length>
```

戻り値

ハードウェア TDO

使用法

名前	説明
[-tdi]	ターゲットに挿入する 16 進数を指定します。
[-tdo]	スキャンされた値と比較する 16 進数を指定します。
[-mask]	TDO 値を比較する際に適用する 16 進数のマスクを指定します。
[-smask]	TDI 値に適用する 16 進数のマスクを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<length>	スキャンするビット数を指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)、[Object \(オブジェクト\)](#)

説明

scan_ir_hw_jtag コマンドは、JTAG インターフェイス ターゲットの命令レジスタに取り込むスキャン パターンを指定します。

このコマンドは、open_hw_target -jtag_mode コマンドを使用してハードウェア ターゲット (hw_target) を JTAG モードで開いたときに作成されたハードウェア JTAG (hw_jtag) オブジェクトをターゲットとします。

scan_ir_hw_jtag コマンドで指定したスキャン パターンをシフトする前に hw_jtag オブジェクトをターゲットとすると、最後に定義されたヘッダー プロパティ (HIR) が指定したデータ パターンの先頭に追加され、最後に定義されたトレーラー プロパティ (TIR) がデータ パターンの末尾に追加されます。

オプションはどの順序で指定してもかまいませんが、1 回のみ指定可能です。-tdi、-tdo、-mask、または -smask オプションで指定する 16 進数文字列で表されたビット数は、<length> で指定した最大値以下にする必要があります。16 進数文字列で表されたビット数が <length> よりも小さい場合は、16 進数文字列の先頭は 0 であると想定されます。

ビットをターゲット命令レジスタにシフトする際、次のステート遷移の表に従って、`scan_ir_hw_jtag` コマンドにより JTAG TAP が現在の安定ステートから IRSHIFT ステートに移動されます。

Current State	Transitions to get to IRSHIFT state
RESET	IDLE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT
IDLE	IRSELECT > IRCAPTURE > IRSHIFT
DRPAUSE	DREXIT2 > DRUPDATE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT
IRPAUSE	IREXIT2 > IRSHIFT
IRPAUSE*	IREXIT2 > IRUPDATE > DRSELECT > IRSELECT > IRCAPTURE > IRSHIFT

注記: * `-force_update` オプションが設定されている場合。

最後のビットがターゲット データ レジスタにシフトされた後、`scan_ir_hw_jtag` コマンドで JTAG TAP が IDLE ステートまたは `run_state_hw_jtag` コマンドで定義された安定ステートに移動されます。

`scan_ir_hw_jtag` コマンドを実行すると、`hw_jtag` からキャプチャされた TDO データを含む 16 進配列が返されるか、正常に実行されなかった場合はエラーが返されます。

`hw_jtag` からの TDO データが `-tdo` オプションで指定した値に一致しない場合はエラーが返され、そのエラーは Tcl の `catch` コマンドでキャプチャできます。



ヒント: `-tdo` および `-mask` オプションを指定した場合、2 つを比較する前に `-tdo` オプションと返された `hw_jtag` TDO データにマスクが適用されます。

引数

`-tdi <arg>` (オプション): ターゲットに挿入する値を 16 進数値で指定します。このオプションを指定しない場合、最後の `scan_ir_hw_jtag` コマンドからの `-tdi` 値が使用されます。`-tdi` オプションは、`scan_ir_hw_jtag` コマンドを初めて実行する場合、および `<length>` を変更した場合には必ず指定する必要があります。

`-tdo <arg>` (オプション): `hw_jtag` 命令レジスタからスキャンされた実際の TDO 値と比較するデータ値を 16 進文字列で指定します。このオプションを指定しない場合、比較は実行されません。`-tdo` オプションを指定しない場合、`-mask` オプションは無視されます。

`-mask <arg>` (オプション): `hw_jtag` からスキャンされた実際の TDO 値と `-tdo` で指定した値を比較する際に適用するマスクを指定します。特定のビット位置が 1 の場合はそのビット値が比較され、0 の場合はそのビット値は比較されません。`-mask` オプションを指定しない場合、最後の `scan_ir_hw_jtag` コマンドからの `-mask` 値が使用されます。



重要: `<length>` を変更して `-mask` オプションを指定しない場合、使用される `-mask` パターンはすべて 1 です。

`-smask <arg>` (オプション): `-tdi` データに適用するマスクを指定します。特定のビット位置が 1 の場合はそのビット位置の TDI データが重要であることを示し、0 の場合は重要でないことを示します。`-smask` オプションは `-tdi` オプションを指定していない場合でも適用されます。`-smask` オプションを指定しない場合、最後の `scan_ir_hw_jtag` コマンドからの `-smask` 値が使用されます。



重要: `<length>` を変更して `-smask` オプションを指定しない場合、使用される `-smask` パターンはすべて 1 です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<length>` (必須): 命令レジスタからスキャンするビット数を指定します。0 より大きい値を 32 ビット符号なし 10 進数整数で指定します。

例

次の例では、JTAG 命令レジスタから 24 ビット値をスキャンしています。

```
scan_ir_hw_jtag 24
```

次の例では、24 ビット値 0x00_0010 (LSB が先頭) を TDI に送信し、TDO 出力をキャプチャして、0xF3_FFFF のマスクを適用して返された TDO 値を `-tdo` で指定した値 0x81_8181 と比較しています。

```
scan_ir_hw_jtag 24 -tdi 000010 -tdo 818181 -mask F3FFFF -smask 0
```

次の例では、指定の TDI 値 0x3f の先頭に 0x が追加されます。

```
scan_ir_hw_jtag 24 -tdi 3f
```

長い命令レジスタシフトを複数の SDR シフトに分割するには、最終ステートを IRPAUSE に設定します。これにより、最初の `scan_ir_hw_jtag` コマンドが IRPAUSE 安定ステートで終了し、次の `scan_ir_hw_jtag` コマンドが IRSHIFT に戻る前に IREXIT2 ステートになります。

```
run_state_hw_jtag IRPAUSE
scan_ir_hw_jtag 8 -tdi aa
scan_ir_hw_jtag 8 -tdi bb
scan_ir_hw_jtag 8 -tdi cc
scan_dr_hw_jtag 128 -tdi 0
```

関連項目

- [connect_hw_server](#)
- [current_hw_target](#)
- [open_hw_target](#)
- [run_state_hw_jtag](#)
- [runtest_hw_jtag](#)

select_objects

GUI でオブジェクトを選択します。

構文

```
select_objects [-add] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
[-add]	既存の選択リストに追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	選択するオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

GUI モードで開いている適切なビューで、指定のオブジェクトを選択します。このコマンドは、表示目的のためにのみ使用されます。ほかのコマンドで使用するために選択されているオブジェクトを返すには、`get_selected_objects` コマンドを使用します。

`select_objects` コマンドでは、指定されたプライマリ オブジェクトに加えてセカンダリ オブジェクトが選択される場合があります。セカンダリ オブジェクトは、[Tools]→[Settings] をクリックし、[Settings] ダイアログ ボックスの [Selection Rules] ページで定義します。選択規則の設定方法は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

オブジェクトの選択を解除するには、`unselect_objects` コマンドを使用します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): 選択するオブジェクトを指定します。

例

次の例では、デバイス上の指定したサイトが選択されます。

```
select_objects [get_sites SLICE_X56Y214]
```

関連項目

- [get_selected_objects](#)
- [highlight_objects](#)
- [mark_objects](#)
- [unselect_objects](#)

select_wave_objects

1 つまたは複数の項目に対するヘルプを表示します。

構文

```
select_wave_objects [-quiet] [-verbose] <items>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><items></code>	波形オブジェクトを選択します。

カテゴリ

Waveform (波形)

説明

Vivado IDE の波形ウィンドウで指定のオブジェクトを選択します。波形ウィンドウで表示されているオブジェクトを選択するためのコマンドで、Vivado IDE の `select_objects` コマンドに似ています。

注記: 開いているシミュレーションまたは現在のシミュレーション (`current_sim`) のシミュレーション オブジェクトを選択するには、`get_hdl_objects` コマンドを使用します。

オブジェクトの選択を解除するには、`select_wave_objects` コマンドを空の文字列を指定して実行します。

```
select_wave_objects ""
```

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<items>` (必須): 選択する波形ウィンドウの 1 つまたは複数のオブジェクトを指定します。オブジェクトは名前で指定します。空の文字列を指定すると、現在選択されている波形オブジェクトの選択が解除されます。

例

次の例では、デバイス上の指定したサイトが選択されます。

```
select_wave_objects {sys_clk_p sys_clk_n}
```

関連項目

- [select_objects](#)
- [unselect_objects](#)

set_bus_skew

バス スキューを定義します。

構文

```
set_bus_skew [-from <args>] [-rise_from <args>] [-fall_from <args>]
             [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
             [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
             <value>
```

使用法

名前	説明
[-from]	バスの始点またはクロックを指定します。
[-rise_from]	始点またはクロックから立ち上がるバスに適用します。
[-fall_from]	始点またはクロックから立ち下がるバスに適用します。
[-to]	バスの終点またはクロックを指定します。
[-rise_to]	終点またはクロックで立ち上がるバスに適用します。
[-fall_to]	終点またはクロックで立ち下がるバスに適用します。
[-through]	通過ピン、セル、またはネットを指定します。
[-rise_through]	ピン、セル、またはネットを立ち上がりで通過するバスに適用します。
[-fall_through]	ピン、セル、またはネットを立ち下がりで通過するバスに適用します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<value>	制約の値を指定します。

カテゴリ

XDC

説明

クロック ドメインをまたがるバス信号のバス スキュー要件を設定します。バス スキュー制約では、バスの最高速の信号と最低速の信号の間の最大スキュー分散を定義します。全体的なデータパス遅延は考慮されません。Vivado 配線は、set_bus_skew 制約が満たされるよう実行されます。バス スキュー制約を使用する例には、グレイ コード ポインタのクロック乗せ換え、マルチプレクサー制御およびマルチプレクサー データ ホールド CDC バスなどがあります。



ヒント: set_bus_skew は特定のバスではなくバスの信号間に設定される制約なので、バス スキュー制約がクロック グループ、最大遅延、フォルス パスにより無効になることはありません。

set_bus_skew 制約は、set_max_delay 制約と共に使用して、良い結果を得ることができます。set_bus_skew 制約では、絶対データパス遅延は考慮されず、ソースとデスティネーションのクロック スキューを含めたデータのデスティネーションでの相対到着時間のみが考慮されます。set_max_delay -datapath-only <SRC_CLK> を使用しても、set_bus_skew に有益です。この制約は、Vivado 配置でソース レジスタとデスティネーション レジスタが遠くに配置されすぎないようにし、配線で set_bus_skew 制約を満たしやすくなるようにします。詳細は、『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903) を参照してください。

スキュー要件を厳しくしすぎないようにするため、バス スキュー値は2つのクロック ドメインの小さい方の周期に近い値にする必要があります。これにより、デスティネーション クロック ドメインで複数のデータがキャプチャされるのを避けることができます。

set_bus_skew コマンドでは、-from および -to オプション (または -fall_from および -rise_to オプションなどのバリエーション) を使用してタイミング パスを定義する必要があります。-through オプションを使用して、パスをさらに特定することも可能です。クロック ドメイン全体を指定するのではなく、-from/-to オプションを指定して明示的に信号パスを指定してください。

- set_bus_skew -from [get_pins <hierarchy/C>] -to [get_pins <hierarchy/D>] <value>
- set_bus_skew -from [get_clocks <clock name>] -to get_clocks <clock name>] <value>



ヒント: バス スキュー制約は、タイミング解析が実行される同期クロック ドメインには設定しないでください。

report_bus_skew コマンドを使用すると、現在のデザインのパスに算出されたスキューをレポートできます。

set_bus_skew コマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-from <args> (オプション): バス スキューを設定するタイミング パスの始点を指定します。ピンまたはセルをタイミング パスの始点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが始点として指定されます。



重要: -from、-to、-through、およびそれらのバリエーションはオプションですが、-from および -to オプションの形式を使用してパスを定義する必要があります。

-rise_from <args> (オプション): 指定した始点で立ち上がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジで駆動されるパスのみが始点として考慮されます。

-fall_from <args> (オプション): 指定した始点で立ち下がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジで駆動されるパスのみが始点として考慮されます。

-to <args> (オプション): バス スキューを定義するタイミング パスの終点 (デスティネーション オブジェクト) を指定します。ポートまたはネットを終点として指定できます。また、クロック オブジェクトを指定すると、そのクロックが供給されるオブジェクトが終点として指定されます。

-rise_to <args> (オプション): 指定した終点で立ち上がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。クロック オブジェクトを指定すると、そのクロックの立ち上がりエッジでキャプチャされるパスのみが終点として考慮されます。

-fall_to <args> (オプション): 指定した終点で立ち下がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。クロック オブジェクトを指定すると、そのクロックの立ち下がりエッジでキャプチャされるパスのみが終点として考慮されます。

`-through <args>` (オプション): 指定したピン、セル インスタンス、またはネットを通過するパスのみにバス スキューを設定します。個別の `-through` (または `-rise_through` および `-fall_through`) 点を順次指定し、デザインを通過する特定のパスを定義できます。パスを定義するには、通過点の指定順序が重要です。通過点を複数のオブジェクトで指定することもできます。この場合、指定の通過オブジェクトのいずれかを通過するタイミング パスが考慮されます。

`-rise_through <args>` (オプション): 指定した通過点で立ち上がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。

`-fall_through <args>` (オプション): 指定した通過点で立ち下がりエッジ遷移が発生するタイミング パスにバス スキューを設定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<value>`: 許容可能なバス スキューをナノ秒で指定します。

例

次の例では、グレイ コードの読み出しポインターと書き込みポインターの間のバス スキューを定義しています。

```
set_bus_skew -from [get_pins gray_coded_read_ptr[*]/C] \
  -to [get_pins gray_coded_write_ptr[*]/D] 2.5
```

関連項目

- [report_bus_skew](#)
- [report_cdc](#)
- [report_datasheet](#)
- [set_data_check](#)
- [set_max_delay](#)

set_case_analysis

入力を 1、0、rising、falling のいずれかに指定します。

構文

```
set_case_analysis [-quiet] [-verbose] <value> <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><value></code>	ピンの論理値を指定します。有効な値は 0、1、rising、falling、zero、one、rise、fall です。
<code><objects></code>	ポートまたはピンを指定します。

カテゴリ

SDC、XDC

説明

ピンまたはポートを 1、0、rising、falling のいずれかの一定したステートに指定します。

このコマンドは通常、ポートの値を強制的に設定し、解析空間、ランタイム、メモリ使用量を削減するために使用します。信号値が一定である場合、それが Vivado タイミング エンジンで認識されるよう指定することが重要です。これは、機能しないパスおよび無関係なパスがレポートされないようにするためにも重要です。

ピンにケース値を設定すると、そのピンを通過するタイミング解析はディスエーブルになり、そのピンを通過するタイミング パスはレポートされません。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<value>` (必須): タイミング解析用にポートまたはピンに適用する値を指定します。有効な値は、0 または zero、1 または one、rise または rising、fall または falling です。rise/rising または fall/falling を指定した場合、指定のピンまたはポートは指定の遷移でのみタイミング解析で考慮され、ほかの遷移は考慮されません。デフォルト値は 1 です。

<objects> (必須): <value> を適用するポートまたはピンを 1 つ以上指定します。

例

次の例では、BUFGMUX (clock_sel) の入力ピンに 2 つのクロックが作成されていますが、選択ピン S に定数値 1 が設定されているため、clk_B のみが出力ピンを介して伝搬されます。

```
create_clock -name clk_A -period 10.0 [get_pins clock_sel/I0]  
create_clock -name clk_B -period 15.0 [get_pins clock_sel/I1]  
set_case_analysis 1 [get_pins clock_sel/S]
```

関連項目

- [create_clock](#)
- [report_timing](#)

set_clock_groups

排他的または非同期のクロック グループを設定します。

構文

```
set_clock_groups [-name <arg>] [-logically_exclusive]
                 [-physically_exclusive] [-asynchronous] [-group <args>] [-quiet]
                 [-verbose]
```

使用法

名前	説明
[-name]	クロック グループの名前を指定します。
[-logically_exclusive]	論理的に排他的なクロック グループを指定します。
[-physically_exclusive]	物理的に排他的なクロック グループを指定します。
[-asynchronous]	非同期クロック グループを指定します。
[-group]	クロック リストを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

デザインのほかのクロックとは排他的または非同期のクロックまたはクロックのグループを定義します。排他的クロックは同時にはアクティブにならないので、それらのクロック間のパスはタイミング解析で無視できます。非同期クロックとは、通常は同じプライマリ クロックを共有していなかったり、共通周期がないために、位相関係が不明のクロックのことです。

このコマンドは、排他的または非同期のクロック ドメイン間を移動するデータパスにフォルス パス制約を定義するのと似ています。詳細は、『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903) を参照してください。

1 つのグループのみを指定した場合、そのグループのクロックはデザインのその他のクロックすべてと非同期または排他的になりますが、そのグループ内のクロックどうしは非同期または排他的にはなりません。

set_clock_groups コマンドを使用した後にクロックを新しく作成した場合、そのクロックもそのグループに対して非同期になります。

このコマンドは、両方のクロックが同時にアクティブになることはないので、1 つの BUFGMUX から派生する複数のクロックにも使用できます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-name <group_name>` (オプション): 作成するクロック グループの名前を指定します。指定しない場合、名前が自動的に付けられます。

`-logically_exclusive` (オプション): 指定のクロックが論理的に排他的であることを示します。

注記: `-logically_exclusive`、`-physically_exclusive`、および `-asynchronous` オプションは、いずれか 1 つのみ使用できます。

`-physically_exclusive` (オプション): 指定のクロックが物理的に排他的であり、デザインに同時に存在できないことを示します。

`-asynchronous` (オプション): 指定のクロックがお互いに非同期であることを示します。

`-group <args>` (オプション): クロック グループに含めるクロックを指定します。クロックの各グループは、その他すべてのグループで指定されたクロックとは排他的または非同期です。



ヒント: クロックのグループを 1 つのみ指定している場合、そのグループはデザインのその他のクロックすべてと排他的または非同期です。クロックは、名前指定するか、または `get_clocks` コマンドを使用してオブジェクトとして指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`src_clk` および `sync_clk` で駆動されるすべてのエレメントを個別のクロック グループにグループ化しています。クロック グループどうしは非同期になります。

```
set_clock_groups -group src_clk -group sync_clk -asynchronous
```

次の例では、指定したクロックの生成クロックを含め、それらをクロック グループに追加しています。

```
set_clock_groups -group [get_clocks -include-generated_clocks src_clk] \
-group [get_clocks -include-generated_clocks sync_clk] -asynchronous
```

注記: 上記の例では、`src_clk` と `sync_clk`、およびそれらのすべての生成クロックは非同期になります。このようにしない場合、生成クロックにはそれらどうしおよびその他のマスター クロックに対するタイミングが適用されます。

次の例では、指定したクロックをグループ化し、デザインのその他のクロックすべてに対して非同期にしています。

```
set_clock_groups -async -group [get_clocks {J_CLK U_CLK}]
```

関連項目

- [get_clocks](#)

- [set_false_path](#)

set_clock_latency

クロック レイテンシを設定します。

構文

```
set_clock_latency [-clock <args>] [-rise] [-fall] [-min] [-max] [-source]
                  [-late] [-early] [-quiet] [-verbose] <latency> <objects>
```

使用法

名前	説明
[-clock]	指定のオブジェクトに割り当てられたレイテンシに関連付けるクロックを指定します。
[-rise]	クロックの立ち上がりレイテンシを指定します。
[-fall]	クロックの立ち下がりレイテンシを指定します。
[-min]	クロックの立ち上がりおよび立ち下がりの最小状態のレイテンシを指定します。
[-max]	クロックの立ち上がりおよび立ち下がりの最大状態のレイテンシを指定します。
[-source]	クロックの立ち上がりおよび立ち下がりのソース レイテンシを指定します。
[-late]	指定の遅延がクロック エッジがどれだけ遅れて到着するかを示すよう指定します。
[-early]	指定の遅延がクロック エッジがどれだけ早く到着するかを示すよう指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<latency>	レイテンシ値を指定します。
<objects>	クロック、ポート、またはピンを指定します。

カテゴリ

[SDC](#)、[XDC](#)

説明

指定したクロック、ポート、またはピンに対して、クロックのソース レイテンシまたはネットワーク レイテンシを定義します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

ソース レイテンシは、クロック信号が波形の始点からデザインのクロック定義点まで伝搬されるのにかかる時間 (ns) です。たとえば、クロックがシステム ボード上のソース (オシレーター) から FPGA 入力ポートまで伝搬されるまでの遅延です。

ネットワーク レイテンシは、クロック信号がデザインの定義点からタイミング パス上のレジスタ クロック ピンに伝搬されるのにかかる時間です。レジスタ クロック ピンの合計クロック レイテンシは、クロックのソース レイテンシとネットワーク レイテンシを加算したものです。

引数

`-clock <args>` (オプション): 指定の `<objects>` に割り当てられた `<latency>` に関連付けるクロックを指定します。 `-clock` オプションを使用しない場合、`<latency>` は指定のピンおよびポートを通過するすべてのクロックに適用されます。

`-rise` (オプション): 立ち上がりクロック エッジのレイテンシを定義します。

`-fall` (オプション): 立ち下がりクロック エッジのレイテンシを定義します。

`-min` (オプション): マルチ コーナー解析用に指定のクロックの最小レイテンシを指定します。

`-max` (オプション): マルチ コーナー解析用に指定のクロックの最大レイテンシを指定します。

注記: `-min` と `-max` オプションを同時に使用することはできません。

`-source` (オプション): 指定の `<latency>` をソース レイテンシとして定義します。クロック ソース レイテンシは、クロック オブジェクトおよびクロック ソース ピンにのみ指定できます。

注記: `-source` オプションを指定しない場合、`<latency>` はネットワーク レイテンシとして使用されます。

`-late` (オプション): `-latency` で指定した遅延がクロック エッジがどれだけ遅れて到着するかを示すよう指定します。

`-early` (オプション): `-latency` で指定した遅延がクロック エッジがどれだけ早く到着するかを示すよう指定します。

注記: `-early` と `-late` オプションを同時に使用することはできず、`-source` オプションを指定する場合にのみ使用できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<latency>` (必須): 適用するクロック レイテンシの量をナノ秒で指定します。

`<objects>` (必須): レイテンシを適用するクロック、ポート、またはピン オブジェクトを指定します。ピンまたはポート オブジェクトを指定すると、そのピンまたはポートの推移的なファンアウトのレジスタ クロックすべてにレイテンシが適用されます。`-clock` を使用した場合、指定のクロックのレジスタ クロック ピンすべてにレイテンシが適用されます。

注記: `<objects>` にクロックを指定した場合、`-clock` オプションは不要で、無視されます。

例

次の例では、CLK_A の立ち上がりエッジ早く到着するレイテンシ量を設定しています。

```
set_clock_latency -source -rise -early 0.4 [get_ports CLK_A]
```

関連項目

- [report_timing](#)

set_clock_sense

ポートまたはピンのクロック極性を設定します。

構文

```
set_clock_sense [-positive] [-negative] [-stop_propagation]
                [-clocks <args>] [-quiet] [-verbose] <pins>
```

使用法

名前	説明
<code>[-positive]</code>	正ユネイト (反転なし) のクロック極性を指定します。
<code>[-negative]</code>	負ユネイト (反転) のクロック極性を指定します。
<code>[-stop_propagation]</code>	指定したピンからのクロック伝搬を停止します。
<code>[-clocks]</code>	クロックのリスト
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><pins></code>	ポート/ピンを指定します。

カテゴリ

SDC、XDC

説明

ポートまたはピンのクロック極性を設定します。これは、クロック オブジェクトに対してピンでの正または負のユネイト性を定義するために使用します。ただし、指定のユネイト性はクロック ネットワークの非ユネイト ポイント、つまりクロック信号が特定できないポイントにのみ適用されます。クロック信号が特定されないで、指定のピンから定義したクロック極性が伝搬されます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-positive` (オプション): 正ユネイト (反転なし) のクロック極性を指定します。

`-negative` (オプション): 負ユネイト (反転) のクロック極性を指定します。

`-stop_propagation` (オプション): 指定したピンまたはポートからの `-clocks` オプションのクロックの伝搬を停止します。クロックおよびデータの伝搬が停止されます。

注記: `-positive`、`-negative`、および `-stop_propagation` は、いずれか 1 つのみ使用できます。

`-clocks <args>` (オプション): 指定のピンおよびポートに対してクロック極性を適用するクロックを指定します。`-clocks` オプションを使用しない場合、クロック極性は指定のピンおよびポートを通過するすべてのクロックに適用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<pins>` (必須): クロック極性を伝搬するポートおよびピンを指定します。

例

次の例では、元のクロックと比較して、正ユネイトパスのみが XOR ゲートの出力ピンを介して伝搬されるよう指定します。

```
set_clock_sense -positive [get_pins xor_a.z]
```

関連項目

- [create_clock](#)
- [get_pins](#)

set_clock_uncertainty

クロックのばらつきを設定します。

構文

```
set_clock_uncertainty [-setup] [-hold] [-from <args>] [-rise_from <args>]
  [-fall_from <args>] [-to <args>] [-rise_to <args>] [-fall_to <args>]
  [-quiet] [-verbose] <uncertainty> [<objects>]
```

使用法

名前	説明
[-setup]	セットアップ チェック用にクロックのばらつきを指定します。
[-hold]	ホールド チェック用にクロックのばらつきを指定します。
[-from]	クロック間のばらつきのソース クロックを指定します。
[-rise_from]	クロック間のばらつきのソース クロックを立ち上がりエッジで指定します。
[-fall_from]	クロック間のばらつきのソース クロックを立ち下がりエッジで指定します。
[-to]	クロック間のばらつきのデスティネーション クロックを指定します。
[-rise_to]	クロック間のばらつきのデスティネーション クロックを立ち上がりエッジで指定します。
[-fall_to]	クロック間のばらつきのデスティネーション クロックを立ち下がりエッジで指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<uncertainty>	クロック ネットワークのばらつきを指定します。
[<objects>]	クロック、ポート、またはピンを指定します。

カテゴリ

SDC、XDC

説明

デザインのクロックのばらつきを追加します。デフォルトのジッター計算を無効にはしません。ユーザー クロックのばらつきとして参照されます。set_clock_uncertainty コマンドは、クロックの定義および関係を変更せずに、デザインの一部のクロックの制約を厳しくするのに便利です。-setup および -hold オプションを使用すると、セットアップ パスとホールド パスを個別に制約できます。

クロックのばらつきは、1 つのクロック ドメイン内またはクロック ドメイン間で、2 つのクロック エッジがレジスタに到着する時間の最大差 (ns) です。

クロックのばらつきはセットアップおよびホールド解析で使用され、解析で使用されるクロック エッジとクロック ツリートポロジに基づいて、各タイミング パスに対してばらつきが算出されます。たとえば、始点と終点と同じクロック ネットに接続されているパスの場合、どちらのソースおよびデスティネーションにも同じクロック エッジが使用されるので、`set_clock_uncertainty` コマンドで最小遅延解析にばらつきを追加しない場合、クロックのばらつきはヌルです。Vivado タイミング エンジンでは、クロックのばらつきは次の式でスラックを算出する際に使用されます。

- セットアップ スラック = セットアップ パス要件 - データ遅延 - クロックのばらつき + クロック スキュー

クロックのばらつきはジッターの異なる要素の関数であり、`report_timing_summary` または `report_timing` コマンドで返される次の式で算出されます。

- クロックのばらつき = $(\sqrt{(T_{sj}^2 + D_j^2)})/2 + PE + UU$

説明:

- T_{sj} = システム ジッターを使用して算出されるトータル システム ジッター。 `set_system_jitter` を参照してください。
- D_j = ディスクリット ジッター。 MMCM や PLL などのハードウェア プリミティブにより追加されるジッター量です。 ディスクリット ジッターは MMCM で生成されるクロックの特性であり、プライマリ クロックで定義される入力ジッターが含まれます。 `set_input_jitter` を参照してください。
- PE = 位相エラー。 MMCM/PLL デバイス モデルからのエラーです。
- UU = ユーザー指定のばらつき。 この `set_clock_uncertainty` コマンドで指定されるユーザー指定のばらつきです。



ヒント: SYSTEM_JITTER はクロックのプロパティとしてレポートされますが、デザインに含まれるすべてのクロックに適用されます。 INPUT_JITTER もプライマリ クロックのプロパティです。これらのプロパティは、`get_property` または `report_property` コマンドを使用して返すことができます。ジッターおよびクロックのばらつきは、`report_timing_summary` および `report_timing` コマンドでレポートされます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

-setup (オプション): セットアップ チェックで適用するクロックのばらつきを指定します。

-hold (オプション): ホールド チェックで適用するクロックのばらつきを指定します。

-from <source_clock_name> (オプション): クロック間のばらつきのソース クロックを指定します。

-rise_from <source_clock_name> (オプション): クロック間のばらつきのソース クロックを立ち上がりエッジで指定します。

-fall_from <source_clock_name> (オプション): クロック間のばらつきのソース クロックを立ち下がりエッジで指定します。

-to <destination_clock_name> (オプション): クロック間のばらつきのデスティネーション クロックを指定します。

-rise_to <destination_clock_name> (オプション): クロック間のばらつきのデスティネーション クロックを立ち上がりエッジで指定します。

-fall_to <destination_clock_name> (オプション): クロック間のばらつきのデスティネーション クロックを立ち下がりエッジで指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<uncertainty> (必須): クロック ネットワークのばらつきをナノ秒で指定します。

<objects> (オプション): ばらつきを定義するクロックを指定します。

例

次の例では、すべてのクロック ドメイン間のばらつきを定義しています。

```
set_clock_uncertainty 0.225 -from [get_clocks] -to [get_clocks]
```

次の例では、wbClk クロック ドメインのセットアップおよびホールドのばらつきを定義しています。

```
set_clock_uncertainty -setup 0.213 [get_clocks wbClk]
set_clock_uncertainty -hold 0.167 [get_clocks wbClk]
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [get_clocks](#)
- [set_input_jitter](#)
- [set_system_jitter](#)

set_data_check

データ間チェックを作成します。

構文

```
set_data_check [-from <args>] [-to <args>] [-rise_from <args>]
               [-fall_from <args>] [-rise_to <args>] [-fall_to <args>] [-setup]
               [-hold] [-clock <args>] [-quiet] [-verbose] <value>
```

使用法

名前	説明
[-from]	データ間チェックの開始ピン/ポートを指定します。
[-to]	データ間チェックの終了ピン/ポートを指定します。
[-rise_from]	データ間チェックの開始ピン/ポートの立ち上がりエッジを指定します。
[-fall_from]	データ間チェックの開始ピン/ポートの立ち下がりエッジを指定します。
[-rise_to]	データ間チェックの終了ピン/ポートの立ち上がりエッジを指定します。
[-fall_to]	データ間チェックの終了ピン/ポートの立ち下がりエッジを指定します。
[-setup]	データ間チェックのセットアップ タイムを指定します。
[-hold]	データ間チェックのホールド タイムを指定します。
[-clock]	チェックの関連ピン/ポートのクロック ドメインを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<value>	定義するチェックのセットアップまたはホールド タイムを指定します。

カテゴリ

SDC、XDC

説明

データ ピンのセットアップ チェックおよびホールド チェックを別のデータ ピンに対して実行します。これは、クロック ピンに対して実行する従来のセットアップ チェックおよびホールド チェックとは異なります。

このコマンドは、セットアップ (最大) およびホールド (最小) タイミング チェックと同様に、2 つの終点間の最小および最大要件を定義します。セットアップ チェックおよびホールド チェックは、-from で指定された関連ピンから -to で指定された制約されたピンまで実行されます。関連ピンは、従来のセットアップ チェックおよびホールド チェックのクロック ピンと似ています。タイミング解析では、指定した 2 つの終点間の到着時間が比較されます。タイミングを満たすためには、到着時間の差が set_data_check <value> 要件未満である必要があります。

set_data_check コマンドの制限は、次のとおりです。

- デスティネーション クロック遅延の変動は無視されます。
- このコマンドはタイミングの目的でのみ使用され、Vivado 配置または配線では考慮されません。

注記: このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-from <value>` (オプション): 指定したデータ ピン、ポート、またはネットからのデータパスをチェックします。 `-from` オプションは、関連ピンを指定します。

`-to <value>` (オプション): 指定したデータ ピン、ポート、またはネットまでのデータパスをチェックします。 `-to` オプションは、制約されたピンを指定します。

`-rise_from <value>` (オプション): 指定したデータ ピン、ポート、またはネットの立ち上がりエッジからのデータパスをチェックします。

`-fall_from <value>` (オプション): 指定したデータ ピン、ポート、またはネットの立ち下がりエッジからのデータパスをチェックします。

`-rise_to <value>` (オプション): 指定したデータ ピン、ポート、またはネットの立ち上がりエッジまでのデータパスをチェックします。

`-fall_to <value>` (オプション): 指定したデータ ピン、ポート、またはネットの立ち下がりエッジまでのデータパスをチェックします。

`-setup <value>` (オプション): セットアップ データ チェックのみを実行します。デフォルトでは、セットアップ チェックとホールド チェックの両方が実行されます。

`-hold <value>` (オプション): ホールド データ チェックを実行します。デフォルトでは、セットアップ チェックとホールド チェックの両方が実行されます。

`-clock <value>` (オプション): チェックの関連ピンまたはポートのクロック ドメインを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<value>` (必須): 定義されたデータ チェックのセットアップまたはホールド タイムをナノ秒 (ns) で定義します。

例

次の例では、ピン A_IN からピン C_IN までのセットアップ違反を調べるデータ チェックを定義します。

```
set_data_check -from A_IN -to C_IN -setup 2.0
```

この例では、A_IN が関連ピンで C_IN が制約されるピンで、この制約により A_IN に対する C_IN のセットアップ チェックが実行されます。C_IN のデータは、A_IN のエッジより 2.0 ns 前に到着する必要があります。

関連項目

- [report_timing](#)
- [set_min_delay](#)
- [set_max_delay](#)

set_delay_model

タイミング解析用のインターコネクト遅延モデルを設定します。

構文

```
set_delay_model [-interconnect <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-interconnect]</code>	タイミング解析に使用するインターコネクト遅延モデルを指定します。有効な値は estimated、actual (デフォルト)、none です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)

説明

タイミング解析用のインターコネクト遅延モデルを設定します。インターコネクト遅延モデルには、actual、estimated、および none の 3 つの設定があります。

- actual を選択すると、タイミング解析に配線済みインターコネクトからの実際の遅延が使用されます。デザインの一部のみが配線されている場合は、配線された部分には実際の遅延が使用され、未配線の部分には見積もり遅延が使用されます。タイミング レポートには、算出された遅延のソースに関する詳細が含まれます。
- estimated に設定すると、インプリメンテーション前のデバイス上のデザインの配置および接続に基づくインターコネクト遅延の見積もりがタイミング解析に含まれます。デザインが完全に配線されている場合でも、見積もり遅延を指定できます。
- none に設定すると、タイミング遅延にインターコネクト遅延は含まれず、ロジック遅延のみが使用されます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-interconnect [actual | estimated | none]` (オプション): 使用する遅延モデルを指定します。デフォルトは actual です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、見積もり値であるタイミング遅延モデルが使用されます。

```
set_delay_model -interconnect estimated
```

関連項目

- [report_timing](#)

set_disable_timing

タイミング アークをディスエーブルにします。

構文

```
set_disable_timing [-from <arg>] [-to <arg>] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
[-from]	セル上の開始ピンを指定します。
[-to]	セル上の終了ピンを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	セルまたはピン、ポート、ライブラリ セル、ライブラリ ピン、ライブラリ セル/セル タイミング アークを指定します。

カテゴリ

SDC、XDC、Timing (タイミング)

説明

指定したセル内の出力ピンまでのタイミング アークをディスエーブルにします。クロック ポートとセルの出力の間にある I/O パスのみがディスエーブルになります。

タイミング アークをディスエーブルにするのは、アークのタイミング解析を実行しないようにするためです。

<cell> を指定すると、そのセルのすべてのタイミング アークがディスエーブルになります。-from および -to オプションを指定した場合、これらのオプションで定義されたタイミング アークがディスエーブルになります。-from オプションのみを指定すると、そのピンからのタイミング アークがすべてディスエーブルになります。-to オプションのみを指定すると、そのピンまでのタイミング アークがすべてディスエーブルになります。

<port> を指定すると、指定した入力ポートからのタイミング パスすべて、または指定した出力ポートまでのタイミング パスすべてがディスエーブルになります。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

-from <pin_name> (オプション): オブジェクト セルのソース ピンを指定します。<pin_name> は名前のみで指定し、階層セル名は必要ありません。階層セル名は <object> で定義されます。

-to <pin_name> (オプション): オブジェクト セルのデスティネーション ピンを指定します。<pin_name> は名前のみで指定し、階層セル名は必要ありません。階層セル名は <object> で定義されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): タイミング アークをディスエーブルにする 1 つまたは複数のオブジェクトを指定します。
`get_cells` またはその他の適切な Tcl コマンドを使用して Vivado オブジェクトとして指定する必要があります。セル、ポート、ピン、ライブラリ セル、ライブラリ ピン、ライブラリ セル/セルのタイミング アークのいずれかを指定できます。

例

次の例では、LUT `div_dec_ff_i/U0/count_i_1` の I0 と O ピンの間のタイミング アークをディスエーブルにし、組み合わせループを切断しています。

```
set_disable_timing -from I0 -to O [get_cells div_dec_ff_i/U0/count_i_1]
```

次の例では、BRAM セルの指定した入力ピンから指定した出力ピンまでのタイミング アークをディスエーブルにしています。

```
set_disable_timing -from WEBWE[3] -to CLKMEM [get_cells \
ldpc_dout360_channel/U_AP_FIFO_ldpc_dout360_channel_ram/mem_reg_0]
```

次の例では、指定したセルのタイミング アークすべてをディスエーブルにしています。

```
set_arcs [get_timing_arcs -of_objects [get_cells \
ldpc_dout360_channel/U_AP_FIFO_ldpc_dout360_channel_ram/mem_reg_0]]
set_disable_timing $arcs
```

関連項目

- [get_cells](#)
- [get_timing_arcs](#)
- [report_timing](#)

set_external_delay

外部遅延を設定します。

構文

```
set_external_delay -from <args> -to <args> [-min] [-max] [-add] [-quiet]
[-verbose] <delay_value>
```

使用法

名前	説明
-from	出力ポートを指定します。
-to	入力ポートを指定します。
[-min]	最小遅延を指定します。
[-max]	最大遅延を指定します。
[-add]	既存の外部遅延に追加します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<delay_value>	外部 (フィードバック) 遅延値を指定します。

カテゴリ

XDC、Timing (タイミング)

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

出力ポートと入力ポート間の外部 (フィードバック) 遅延をナノ秒 (ns) で設定します。外部遅延は、外部フィードバックを使用する PLL/MMCM の PLL/MMCM 補正遅延を算出するために使用されます。

最小値または最大値を指定できます。デフォルトでは、指定した値は最小 (ホールド) および最大 (セットアップ) 補正遅延の両方に適用されます。

このコマンドを実行すると、指定した遅延が返されます。

引数

-from <arg> (必須): 出力ポート名を指定します。

-to <arg> (必須): 入力ポート名を指定します。

-min (オプション): 遅延値がホールド タイム解析用の最小遅延値であることを指定します。

-max (オプション): 遅延値がセットアップ解析用の最大遅延値であることを指定します。

-add (オプション): 指定の遅延を既存の外部遅延に追加します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<delay_value> (必須): 外部遅延値をナノ秒 (ns) で指定します。デフォルト値は 0 です。

例

次の例では、ClkOut と ClkFb 間の外部フィードバック遅延を 1.0 ns に設定しています。

```
set_external_delay -from [get_ports ClkOut] -to [get_ports ClkFb] 1.0
```

関連項目

- [report_timing](#)
- [set_input_delay](#)
- [set_output_delay](#)

set_false_path

フォルス パスを定義します。

構文

```
set_false_path [-setup] [-hold] [-rise] [-fall] [-reset_path]
               [-from <args>] [-rise_from <args>] [-fall_from <args>] [-to <args>]
               [-rise_to <args>] [-fall_to <args>] [-through <args>]
               [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
```

使用法

名前	説明
[-setup]	パスのセットアップ タイミング解析を除外します。
[-hold]	パスのホールド タイミング解析を除外します。
[-rise]	指定のパスの立ち上がり遅延のみを除外します。
[-fall]	指定のパスの立ち下がり遅延のみを除外します。
[-reset_path]	フォルス パスを設定する前にこのパスをリセットします。
[-from]	パスの始点またはクロックを指定します。
[-rise_from]	始点またはクロックから立ち上がるパスに適用します。
[-fall_from]	始点またはクロックから立ち下がるパスに適用します。
[-to]	パスの終点またはクロックを指定します。
[-rise_to]	終点またはクロックで立ち上がるパスに適用します。
[-fall_to]	終点またはクロックで立ち下がるパスに適用します。
[-through]	通過ピン、セル、またはネットを指定します。
[-rise_through]	ピン、セル、またはネットを立ち上がりで通過するパスに適用します。
[-fall_through]	ピン、セル、またはネットを立ち下がりで通過するパスに適用します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

タイミング解析で無視するフォルス タイミング パスを設定します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

-setup (オプション): 指定のタイミング パスに対してセットアップ タイミング解析を実行しないようにします。

-hold (オプション): 指定のタイミング パスに対してホールドタイミング解析を実行しないようにします。

-rise (オプション): 指定のタイミング パスの立ち上がり遅延を除外します。

-fall (オプション): 指定のタイミング パスの立ち下がり遅延を除外します。

-reset_path (オプション): フォルス パスを設定する前にタイミング パスをリセットします。これにより、指定のタイミング パスから除外ベースのタイミング制約がすべて消去されます。

-from <element_name> (オプション): パスの始点またはクロックを指定します。有効な始点は、クロック オブジェクト、順序ロジックのクロック ピン、入力または入出力プライマリ ポートです。

-rise_from <element_name> (オプション): 始点またはクロックから立ち上がるパスに適用します。

-fall_from <element_name> (オプション): 始点またはクロックから立ち下がるパスに適用します。

-to <element_name> (オプション): パスの終点またはクロックを指定します。

-rise_to <element_name> (オプション): 終点またはクロックで立ち上がるパスに適用します。

-fall_to <element_name> (オプション): 終点またはクロックで立ち下がるパスに適用します。

-through <element_name> (オプション): 通過するピン、セル、またはネットを指定します。

-rise_through <element_name> (オプション): ピン、セル、またはネットを立ち上がりで通過するパスに適用します。

-fall_through <element_name> (オプション): ピン、セル、またはネットを立ち上がりで通過するパスに適用します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、bftClk からのパスのセットアップ タイミング解析を除外します。

```
set_false_path -setup -from bftClk
```

次の例では、タイミング解析から指定した 2 つのクロックの間にあるパスを除外します。

```
set_false_path -from [get_clocks GT0_RXUSRCLK2_OUT] \
    -to [get_clocks DRPCLK_OUT]
```

関連項目

- [get_clocks](#)
- [get_pins](#)
- [get_ports](#)
- [report_timing](#)

set_hierarchy_separator

階層区切り文字を設定します。

構文

```
set_hierarchy_separator [-quiet] [-verbose] [<separator>]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<separator>]</code>	階層区切り文字を指定します。デフォルトはスラッシュ (/) です。

カテゴリ

SDC、XDC

説明

デザインで使用する階層区切り文字を設定します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`<separator>` (オプション): 階層区切り文字として使用する文字を指定します。有効な文字は、/、@、^、#、.、および | です。デフォルトはスラッシュ (/) で、`<separator>` を指定しない場合に使用されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、階層区切り文字を | に変更しています。

```
set_hierarchy_separator |
```

次の例では、デフォルトの階層区切り文字 (/) に戻しています。

```
set_hierarchy_separator
```

関連項目

- [get_hierarchy_separator](#)

set_hw_sysmon_reg

システム モニターのレジスタ値を設定します。

構文

```
set_hw_sysmon_reg [-quiet] [-verbose] <hw_sysmon> <hexaddress> <hexdata>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sysmon></code>	hw_sysmon オブジェクトを指定します。
<code><hexaddress></code>	書き込む 16 進数アドレスを指定します。
<code><hexdata></code>	書き込む値を 16 進数で指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定のアドレスにあるシステム モニター レジスタを指定の 16 進数値に設定します。このコマンドは、現在のデバイスのハードウェア システム モニター (hw_sysmon) 上にあるレジスタをその 16 進アドレス値により特定し、そのレジスタに指定の 16 進データ値に設定します。



重要: システム モニター上の一部のレジスタは読み出し専用であり、直接設定することはできません。システム モニター上の読み出し専用レジスタの値を設定しようとしても、何も実行されません。

システム モニター (SYSMON) Analog-to-Digital Converter (ADC) は、ハードウェア デバイス (hw_device) のダイの温度および電圧を測定するために使用されます。システム モニターは、オンチップ温度および電源センサーを使用して物理環境を監視します。ADC は最大で 17 の外部アナログ入力チャンネルにアクセスできます。

システム モニターのデータは、専用レジスタであるステータス レジスタおよび制御レジスタに保存され、`get_hw_sysmon_reg` および `set_hw_sysmon_reg` コマンドを使用してアクセスできます。システム モニター レジスタのアドレスの詳細は、『UltraScale アークテクチャ システム モニター ユーザー ガイド』(UG580) の「SYSMON のレジスタ インターフェイス」の章または『7 シリーズ FPGA および Zynq-7000 SoC XADC デュアル 12 ビット 1MSPS アナログ - デジタル コンバーター』(UG480) を参照してください。

`set_hw_sysmon_reg` コマンドを使用すると、システム モニターのレジスタに指定の 16 進データ値を直接書き込むことができますが、`set_property` コマンドを使用して hw_sysmon オブジェクトのプロパティ値をアップデートし、その後 `commit_hw_sysmon` コマンドを使用してプロパティ 値を書き込むのが推奨される方法です。

`set_hw_sysmon_reg` コマンドを実行すると、hw_sysmon オブジェクト上の指定のアドレスにあるハードウェア システム モニター レジスタ (hw_sysmon_reg) オブジェクトに指定の 16 進数値が書き込まれるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sysmon>` (必須): レジスタを設定する `hw_sysmon` オブジェクトを指定します。`hw_sysmon` は、`get_hw_sysmon` コマンドを使用してオブジェクトとして指定する必要があります。

`<hexaddress>`: システム モニター上の値を設定するステータス レジスタの 16 進アドレスを指定します。

`<hexaddress>`: システム モニター上の値を設定するステータス レジスタの 16 進アドレスを指定します。

例

次の例では、現在のハードウェア デバイスシステム モニター上の指定の 16 進アドレスにあるレジスタに指定の 16 進データ値を設定しています。

```
set_hw_sysmon_reg [current_hw_device] 00 9D28
```

関連項目

- [commit_hw_sysmon](#)
- [connect_hw_server](#)
- [current_hw_device](#)
- [get_hw_sysmons](#)
- [get_hw_sysmon_reg](#)
- [open_hw_target](#)
- [refresh_hw_sysmon](#)

set_input_delay

ポートの入力遅延を設定します。

構文

```
set_input_delay [-clock <args>] [-reference_pin <args>] [-clock_fall]
                [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included]
                [-source_latency_included] [-quiet] [-verbose] <delay> <objects>
```

使用法

名前	説明
[-clock]	相対クロックを指定します。
[-reference_pin]	相対ピンまたはポートを指定します。
[-clock_fall]	遅延がクロックの立ち下がりエッジを基準にすることを示します。
[-rise]	立ち上がり遅延を指定します。
[-fall]	立ち下がり遅延を指定します。
[-max]	最大遅延を指定します。
[-min]	最小遅延を指定します。
[-add_delay]	既存の入力遅延を削除しません。
[-network_latency_included]	クロックのネットワーク レイテンシが既に含まれていることを示します。
[-source_latency_included]	クロックのソース レイテンシが既に含まれていることを示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<delay>	遅延値を指定します。
<objects>	ポートを指定します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

デザインのインターフェイスのクロック エッジに対するプライマリ入力ポートの外部システム レベル遅延を指定します。入力遅延値はナノ秒 (ns) で指定し、デバイスのインターフェイスにおけるクロックおよびデータの相対位相によって、正または負の値を指定できます。

ザイリンクス FPGA デザインのシステム レベル タイミングを正確にモデリングするには、FPGA 外部のオブジェクトのタイミング遅延をデザインのプライマリ入力または出力ポートに割り当てる必要があります。これらの遅延は、`set_input_delay` および `set_output_delay` コマンドを使用して定義します。



重要: 入力ポートに `set_max_delay` 制約も設定されている場合は、指定した入力遅延値は `max_delay` の算出の一部として考慮されます。つまり、入力遅延は入力ポートを含むタイミング パス上の最大遅延の一部を占めるということです。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-clock <arg>` (オプション): 入力遅延が指定のクロックを基準にしていることを指定します。デフォルトでは立ち上がりエッジが使用されます。`-clock_fall` を使用すると、立ち下がりエッジを使用するよう指定できます。

`-reference_pin <arg>` (オプション): 遅延がクロックではなく指定のピンに現れるクロックのアクティブ エッジを基準としていることを指定します。

`-clock_fall` (オプション): 遅延が指定のクロックの立ち上がりエッジではなく立ち下がりエッジを基準にすることを示します。

`-rise` (オプション): 入力遅延が指定したポートの立ち上がり遷移に適用されることを指定します。デフォルトでは、立ち上がり遷移と立ち下がり遷移の両方に適用されます。

`-fall` (オプション): 入力遅延が指定したポートの立ち下がり遷移に適用されることを指定します。デフォルトでは、立ち上がり遷移と立ち下がり遷移の両方に適用されます。

`-max` (オプション): 指定の入力遅延が最大パス遅延の算出にのみ使用されるよう指定します。

`-min` (オプション): 指定の入力遅延が最小パス遅延の算出にのみ使用されるよう指定します。

`-add_delay` (オプション): ポートに指定の遅延制約を追加し、ポートに既に定義されているほかの `set_input_delay` 制約と共存させます。デフォルトでは、既存の遅延が置き換えられます。

`-network_latency_included` (オプション): 遅延値に基準クロックのクロック ネットワーク レイテンシが含まれることを示します。指定の入力または出力遅延値にソース レイテンシまたはネットワーク レイテンシが含まれていない場合、Vivado タイミング エンジンでは、クロック エッジがクロック レイテンシの後にキャプチャ フリップフロップに到達すると考慮されます。

`-source_latency_included` (オプション): 指定の遅延値に相対クロックのソース レイテンシが含まれることを示します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<delay>` (必須): 指定のポートに適用する入力遅延をナノ秒 (ns) で指定します。有効な値は浮動小数点値で、デフォルト値は 0 です。

<objects> (必須): 遅延値を適用するポートを 1 つまたは複数指定します。

例

次の例では、ポート DIN 上の入力遅延を指定しています。入力遅延は 3 で、クロック clk1 の立ち上がりエッジを基準としています。

```
set_input_delay -clock clk1 3 DIN
```

次の例では、ポート DIN 上の入力遅延を指定しています。入力遅延は 2 で、クロック clk1 の立ち下がりエッジを基準としています。

```
set_input_delay -clock_fall -clock clk1 2 DIN
```

次の例では、ポート reset 上の入力遅延を指定しています。入力遅延は 2 で、クロック wbClk から派生する wbClk_IBUF_BUFG_inst/O ピンに現れるクロックの立ち上がりエッジを基準としています。

```
set_input_delay -clock wbClk 2 -reference_pin \
[get_pin wbClk_IBUF_BUFG_inst/O] reset
```

次の例では、clk_dds という名前のクロックを作成し、デバイス外にあるクロックの立ち上がりエッジおよび立ち下がりエッジの両方でデータが送信されてから、クロックの立ち上がりエッジおよび立ち下がりエッジの両方で動作する内部フリップフロップのデータ入力までの入力遅延制約を作成しています。

```
create_clock -name clk_dds -period 6 [get_ports DDR_CLK_IN]
set_input_delay -clock clk_dds -max 2.1 [get_ports DDR_IN]
set_input_delay -clock clk_dds -max 1.9 [get_ports DDR_IN] -clock_fall -
add_delay
set_input_delay -clock clk_dds -min 0.9 [get_ports DDR_IN]
set_input_delay -clock clk_dds -min 1.1 [get_ports DDR_IN] -clock_fall -
add_delay
```

注記: -add_delay オプションが使用されているので、新しい最小および最大遅延制約は同じポートの最初の遅延と共存します。

次の例では、デザインのクロック以外の入力ポートすべてに入力遅延を指定しています。all_inputs ではクロックポートを含むデザインのすべてのポートが返されますが、set_input_delay ではクロックポートに入力遅延を設定していません。入力遅延は 1 で、クロック wbClk の立ち上がりエッジを基準としています。

```
set_input_delay -clock wbClk 1 [all_inputs]
```

次の例では、reset および wbDataForInput に対して、クロック wbClk の立ち上がりエッジを基準とした入力遅延を 4 に設定しています。

```
set_input_delay -clock wbClk 4 [list reset wbDataForInput]
```

関連項目

- [all_clocks](#)
- [all_inputs](#)
- [check_timing](#)
- [create_clock](#)
- [get_ports](#)
- [report_timing](#)

- [set_max_delay](#)
- [set_output_delay](#)

set_input_jitter

クロック オブジェクトの入力ジッターを設定します。

構文

```
set_input_jitter [-quiet] [-verbose] <clock> <input_jitter>
```

戻り値

clock

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><clock></code>	クロック
<code><input_jitter></code>	入力ジッター値を指定します。有効な値は 0 以上です。

カテゴリ

XDC

説明

特定のプライマリ クロックに追加のジッターを指定するには、`set_input_jitter` コマンドを使用します。

入力ジッターとは、理想的なクロック到達時間と比較した連続クロック エッジ間のばらつきです。このコマンドは、指定したプライマリ クロック (`create_clock` コマンドで定義) の入力ジッターをナノ秒 (ns) で設定します。このコマンドでは 1 つのクロックしか指定できないので、各プライマリ クロックにジッターを個別に設定する必要があります。

`set_input_jitter` コマンドでは、プライマリ クロックの入力ジッターのみを設定可能です。生成クロックまたは自動派生クロックの入力ジッターは設定できません。入力ジッターは、MMCM および PLL を除き、マスター クロックから生成クロックに伝搬されます。

入力ジッターは、ディスクリート ジッターの計算に使用されます。このジッターは MMCM や PLL などのハードウェア プリミティブにより追加されるジッター量で、MMCM により生成されるクロックの特性です。

`set_clock_uncertainty` を参照してください。

合成中は `set_input_jitter` コマンドは無視されます。



ヒント: INPUT_JITTER はプライマリ クロックのプロパティであり、`get_property` または `report_property` コマンドを使用して返すことができます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<clock>` (必須): `create_clock` コマンドで定義されたプライマリ クロックの名前を指定します。

`<input_jitter>` (必須): 指定のクロック オブジェクトの入力ジッターを指定します。有効な値は 0 以上です。値はナノ秒 (ns) で指定します。

例

次の例では、2 つのクロック `sysClk` および `procClk` の入力ジッターを 0.3 ns に設定しています。ジッター値は同じですが、`set_input_jitter` コマンドでは 1 つのクロックしか指定できないので、クロックごとに個別に設定する必要があります。

```
set_input_jitter sysClk 0.3
set_input_jitter procClk 0.3
```

次の例では、プライマリ クロック `sysClk` を定義し、このプライマリ クロックを 2 で分周したクロック `sysClkDiv2` を生成した後、プライマリ クロックの入力ジッターを 0.15 ns に設定しています。入力ジッターは、生成クロックに自動的に伝搬されます。

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_generated_clock -name sysClkDiv2 -source [get_ports sysClk] \
    -divide_by 2 [get_pins clkgen/sysClkDiv/Q]
set_input_jitter sysClk 0.15
```

注記: この例では、`sysClkDiv2` はフリップフロップでインプリメントされた分周器で生成されるので、入力ジッターはプライマリ クロックから伝搬されます。

関連項目

- [all_clocks](#)
- [check_timing](#)
- [create_clock](#)
- [create_generated_clock](#)
- [report_clocks](#)
- [report_timing](#)
- [set_clock_uncertainty](#)
- [set_clock_latency](#)
- [set_system_jitter](#)

set_load

ポートおよびネットの容量を設定します。

構文

```
set_load [-rise] [-fall] [-max] [-min] [-quiet] [-verbose] <capacitance>
        <objects>
```

使用法

名前	説明
<code>[-rise]</code>	立ち上がり容量値を指定します (ポートのみ)。
<code>[-fall]</code>	立ち下がり容量値を指定します (ポートのみ)。
<code>[-max]</code>	最大容量値を指定します。
<code>[-min]</code>	最小容量値を指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><capacitance></code>	容量値を指定します。
<code><objects></code>	ポートまたはネットを指定します。

カテゴリ

SDC、XDC

説明

出力ポートの負荷容量を指定の値に設定します。負荷容量は、`report_power` コマンドによる消費電力解析では使用されますが、タイミング解析では使用されません。



ヒント: キャパシタンスのデフォルト単位はピコファラド (pF) ですが、`set_units` コマンドを使用して変更できます。

このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-max` (オプション): SDC 互換性のために提供されています。Vivado ツールでは無視されます。

`-min` (オプション): SDC 互換性のために提供されています。Vivado ツールでは無視されます。

`-rise` (オプション): SDC 互換性のために提供されています。Vivado ツールでは無視されます。

`-fall` (オプション): SDC 互換性のために提供されています。Vivado ツールでは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<capacitance>` (必須): 負荷容量の値を指定します。値は、0 以上の浮動小数点値で指定します。デフォルトは 0 です。

`<objects>` (必須): 負荷容量を適用する出力ポート オブジェクトを 1 つまたは複数指定します。デザインのすべての出力を取得するには、`all_outputs` コマンドを使用します。

例

次の例では、すべてのポートの負荷容量が指定の値に設定されます。

```
set_load 5.5 [all_outputs]
```

次の例では、指定の出力ポートの立ち上がりエッジおよび立ち下がりエッジ負荷容量が設定されます。

```
set_load -rise -fall 8 [get_ports wbOutput*]
```

関連項目

- [all_outputs](#)
- [get_ports](#)
- [report_power](#)

set_logic_dc

ポート/ピンをドントケアに設定します。

構文

```
set_logic_dc [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ポートまたはピンを指定します。

カテゴリ

[SDC](#)、[XDC](#)

説明

指定の入力ポートまたは入力ピンを論理値 X (ドントケア) に設定します。このコマンドは合成ではサポートされません。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 入力ポートまたはピンを 1 つまたは複数指定します。

例

次の例では、指定したポートを X に設定しています。

```
set_logic_dc [get_ports reset]
```

関連項目

- [all_inputs](#)

- [get_pins](#)
- [get_ports](#)
- [set_logic_one](#)
- [set_logic_unconnected](#)
- [set_logic_zero](#)

set_logic_one

ポート/ピンを論理 1 に設定します。

構文

```
set_logic_one [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ポートまたはピンを指定します。

カテゴリ

SDC、XDC

説明

指定の入力ポートまたは入力ピンを論理 1 に設定します。このコマンドは合成ではサポートされません。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 入力ポートまたはピンを 1 つまたは複数指定します。

例

次の例では、指定の入力ポートを論理 1 に設定しています。

```
set_logic_one [get_ports reset]
```

次の例では、入力ポート `reset` および `wbDataForInput` を論理 1 に設定しています。

```
set_logic_one [list [get_ports reset] [get_ports wbDataForInput]]
```

次の例では、インスタンス reset_IBUF の入力ピン I を論理 1 に設定しています。

```
set_logic_one [get_pins reset_IBUF_inst/I]
```

関連項目

- [all_inputs](#)
- [get_pins](#)
- [get_ports](#)
- [set_logic_dc](#)
- [set_logic_unconnected](#)
- [set_logic_zero](#)

set_logic_unconnected

ポート/ピンを未接続に設定します。

構文

```
set_logic_unconnected [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ポートまたはピンを指定します。

カテゴリ

[XDC](#)

説明

指定した出力ポートまたはピンを未接続に指定します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 出力ポートまたはピンを 1 つまたは複数指定します。

例

次の例では、指定したポートを未接続に設定しています。

```
set_logic_unconnected [get_ports OUT1]
```

関連項目

- [set_logic_dc](#)

- [set_logic_one](#)
- [set_logic_zero](#)

set_logic_zero

ポート/ピンを論理 0 に設定します。

構文

```
set_logic_zero [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ポートまたはピンを指定します。

カテゴリ

SDC、XDC

説明

指定の入力ポートおよび入力ピンを論理 0 に設定します。このコマンドは合成ではサポートされません。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 入力ポートまたはピンを 1 つまたは複数指定します。

例

次の例では、指定したポートを論理 0 に設定しています。

```
set_logic_zero [get_ports reset]
```

関連項目

- [all_inputs](#)

- [get_pins](#)
- [get_ports](#)
- [set_logic_one](#)
- [set_logic_unconnected](#)

set_max_delay

タイミング パスの最大遅延を指定します。

構文

```
set_max_delay [-rise] [-fall] [-reset_path] [-from <args>]
               [-rise_from <args>] [-fall_from <args>] [-to <args>] [-rise_to <args>]
               [-fall_to <args>] [-through <args>] [-rise_through <args>]
               [-fall_through <args>] [-datapath_only] [-quiet] [-verbose] <delay>
```

使用法

名前	説明
[-rise]	遅延値が立ち上がりパス遅延に適用されます。
[-fall]	遅延値が立ち下がりパス遅延に適用されます。
[-reset_path]	最大遅延を設定する前にこのパスをリセットします。
[-from]	パスの始点またはクロックを指定します。
[-rise_from]	始点またはクロックから立ち上がるパスに適用します。
[-fall_from]	始点またはクロックから立ち下がるパスに適用します。
[-to]	パスの終点またはクロックを指定します。
[-rise_to]	終点またはクロックで立ち上がるパスに適用します。
[-fall_to]	終点またはクロックで立ち下がるパスに適用します。
[-through]	通過ピン、セル、またはネットを指定します。
[-rise_through]	ピン、セル、またはネットを立ち上がりで通過するパスに適用します。
[-fall_through]	ピン、セル、またはネットを立ち下がりで通過するパスに適用します。
[-datapath_only]	遅延の算出からクロック スキューおよびジッターを除外します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<delay>	遅延値を指定します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

タイミング パスに許容される最大遅延をナノ秒 (ns) で指定します。-rise または -fall オプションを使用しない場合、指定の遅延値は指定のタイミング パスの立ち上がりエッジと立ち下がりエッジの両方に割り当てられます。

最大立ち上がりおよび立ち下がり遅延は、`set_min_delay` コマンドで定義されている同じパスの最小立ち上がりおよび立ち下がり遅延以上である必要があります。最大遅延が最小遅延より小さい場合、タイミング パスから最初に割り当てられていた遅延値が競合として削除され、削除された制約で指定されていた遅延値が 0 に設定されます。

遅延値は、`-from`、`-through`、または `-to` オプションの少なくとも 1 つを使用して定義されたタイミング パスに割り当てする必要があります。`-to` 終点などの汎用パス遅延は、`-from/-to` または `-from/-through/-to` パス定義などのような特定のパス定義により上書きされます。



重要: プライマリ入力または出力ポートに割り当てられる場合、システム レベルの遅延は入力または出力ポートを含むタイミング パス上の最大遅延の一部を占めます。つまり、`set_input_delay` または `set_output_delay` で指定された遅延は最大遅延の一部として考慮されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-rise` (オプション): 遅延値をタイミング パスの立ち上がりエッジに適用するよう指定します。

`-fall` (オプション): 遅延値をタイミング パスの立ち下がりエッジに適用するよう指定します。

注記: `-rise` または `-fall` のどちらも指定しない場合、遅延は立ち上がりエッジおよび立ち下がりエッジ両方のパス遅延に適用されます。

`-reset_path` (オプション): 新しいパス遅延を適用する前に既存の立ち上がりエッジまたは立ち下がりエッジ最大遅延を消去するよう指定します。`-to` のみを指定した場合、指定の終点に到達するすべてのパスで遅延が消去されます。`-from` のみを指定した場合、指定の始点からのすべてのパスで遅延が消去されます。`-from/-to` または `-from/-through/-to` を指定した場合、指定のパスがリセットされます。

`-from <value>` (オプション): パスの始点またはクロックを指定します。有効な始点は、プライマリ入力または入出力ポート、あるいは同期エレメントのクロック ピンです。クロックが指定されている場合は、そのクロックおよびそのクロックに接続されているレジスタのすべてのクロック ピンに関連するプライマリ入力および入出力ポートが始点として指定されます。

`-rise_from <element_name>` (オプション): 最大遅延を始点またはクロックから立ち上がるパスに適用します。

`-fall_from <element_name>` (オプション): 最大遅延を始点またはクロックから立ち下がるパスに適用します。

`-to <element_name>` (オプション): パスの終点またはクロックを指定します。有効な終点は、プライマリ出力または入出力ポート、あるいは同期エレメントのデータ ピンです。クロックが指定されている場合は、そのクロックおよびそのクロックに接続されているレジスタのすべてのデータ ピンに関連するプライマリ入力および入出力ポートが終点として指定されます。

`-rise_to <element_name>` (オプション): 最大遅延を終点またはクロックで立ち上がるパスに適用します。

`-fall_to <element_name>` (オプション): 最大遅延を終点またはクロックで立ち下がるパスに適用します。

`-through <element_name>` (オプション): 通過するピン、セル、またはネットを指定します。

`-rise_through <element_name>` (オプション): 最大遅延をピン、セル、またはネットを立ち上がりで通過するパスに適用します。

`-fall_through <element_name>` (オプション): 最大遅延をピン、セル、またはネットを立ち下がりで通過するパスに適用します。

`-datapath_only` (オプション): 指定のパスの遅延算出からクロック スキューとジッターを除外します。このオプションは、異なるクロックを持つ順次エレメント間の遅延を制約する際に遅延の算出にクロック スキューおよびジッターを考慮しない場合に使用します。最初のフリップフロップの clock-to-Q 遅延、フリップフロップ間のワイヤ遅延、および 2 番目のフリップフロップのセットアップ タイムのみが考慮されます。



重要: `-datapath_only` でデータパスを指定する場合、`-from` オプションを使用してパスの始点を定義する必要があります。指定のパスのホールド チェックは、`-datapath_only` オプションを使用したときにフォルス パスとして自動的に定義されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<delay> (必須): 最大遅延値をナノ秒 (ns) で指定します。<delay> は、正または負の浮動小数点値としてナノ秒 (ns) で指定します。デフォルト値は 0 です。

例

次の例では、すべての入力ポートと出力ポートの間の最大遅延を 60 ns に設定しています。

```
set_max_delay 60 -from [all_inputs] -to [all_outputs]
```

次の例では、既存の最大遅延を消去し、指定のクロックが供給される終点へのパスに新しい最大遅延を設定しています。

```
set_max_delay -reset_path 50 -to [get_clocks spi_clk]
```

`set_max_delay` コマンドは、単純なシンクロナイザーが使用されている場合に、クロック ドメインをまたぐタイミング制約を定義するのによく使用されます。次の例では、2 つのフリップフロップ (FF1 および FF2) に異なるクロックが供給されており、FF1/C が net1 を介して FF2/D に直接接続されています。この接続の遅延を 4.0 ns に制限するには、次のいずれかの制約を使用します。

```
set_max_delay -from FF1 -to FF2 -datapath_only 4.0
set_max_delay -from FF1/C -to FF2/D -datapath_only 4.0
```

関連項目

- [get_clocks](#)
- [get_nets](#)
- [get_ports](#)
- [report_timing](#)
- [set_input_delay](#)
- [set_min_delay](#)
- [set_output_delay](#)
- [set_units](#)

set_max_time_borrow

ラッチ用に借りることのできる最大時間を設定します。

構文

```
set_max_time_borrow [-quiet] [-verbose] <delay> <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><delay></code>	遅延値を指定します。有効な値は 0 以上です。
<code><objects></code>	クロック、セル、データ ピン、またはクロック ピンを指定します。

カテゴリ

SDC、XDC

説明

ラッチのタイミングを解析する際に、ネット間で借りることのできる最大時間をナノ秒で指定します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<delay>` (必須): 指定のオブジェクトに適用する遅延を指定します。`<delay>` は 0 以上の浮動小数点値としてナノ秒 (ns) で指定します。デフォルト値は 0 です。

`<objects>` (必須): 制限を適用するクロック、セル、データ ピン、またはクロック ピンを 1 つまたは複数指定します。

例

次の例では、すべてのクロックに接続されているラッチで借りることのできる時間を 0 に設定しています。デザイン全体で時間の借用がディスエーブルになります。

```
set_max_time_borrow 0.0 [all_clocks]
```

次の例では、最上位階層のネットで借りることのできる時間を 20 に設定しています。

```
set_max_time_borrow 20 {top/*}
```

関連項目

- [all_clocks](#)
- [get_clocks](#)
- [get_nets](#)

set_min_delay

タイミング パスの最小遅延を指定します。

構文

```
set_min_delay [-rise] [-fall] [-reset_path] [-from <args>]
              [-rise_from <args>] [-fall_from <args>] [-to <args>] [-rise_to <args>]
              [-fall_to <args>] [-through <args>] [-rise_through <args>]
              [-fall_through <args>] [-quiet] [-verbose] <delay>
```

使用法

名前	説明
[-rise]	遅延値が立ち上がりパス遅延に適用されます。
[-fall]	遅延値が立ち下がりパス遅延に適用されます。
[-reset_path]	最小遅延を設定する前にこのパスをリセットします。
[-from]	パスの始点またはクロックを指定します。
[-rise_from]	始点またはクロックから立ち上がるパスに適用します。
[-fall_from]	始点またはクロックから立ち下がるパスに適用します。
[-to]	パスの終点またはクロックを指定します。
[-rise_to]	終点またはクロックで立ち上がるパスに適用します。
[-fall_to]	終点またはクロックで立ち下がるパスに適用します。
[-through]	通過ピン、セル、またはネットを指定します。
[-rise_through]	ピン、セル、またはネットを立ち上がりで通過するパスに適用します。
[-fall_through]	ピン、セル、またはネットを立ち下がりで通過するパスに適用します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<delay>	遅延値を指定します。

カテゴリ

SDC、XDC

説明

タイミング パスに許容される最小遅延をナノ秒 (ns) で指定します。-rise または -fall オプションを使用しない場合、指定の遅延値は指定のタイミング パスの立ち上がりエッジと立ち下がりエッジの両方に割り当てられます。



重要: 最小立ち上がりおよび立ち下がり遅延は、同じパスの最大立ち上がりおよび立ち下がり遅延以下である必要があります。最大遅延が最小遅延より小さい場合、タイミング パスから最初に割り当てられていた遅延値が削除され、0 にリセットされます。

遅延値は、`-from`、`-through`、または `-to` オプションの少なくとも 1 つを使用して定義されたタイミング パスに割り当てる必要があります。`-to` 終点などの汎用パス遅延は、`-from/-to` または `-from/-through/-to` パス定義などのような特定のパス定義により上書きされます。

このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-rise` (オプション): 遅延値をタイミング パスの立ち上がりエッジに適用するよう指定します。

`-fall` (オプション): 遅延値をタイミング パスの立ち下がりエッジに適用するよう指定します。

`-reset_path` (オプション): 新しいパス遅延を適用する前に既存の立ち上がりエッジまたは立ち下がりエッジ最小遅延を消去します。`-to` のみを指定した場合、指定の終点に到達するすべてのパスで遅延が消去されます。`-from` のみを指定した場合、指定の始点からのすべてのパスで遅延が消去されます。`-from/-to` または `-from/-through/-to` を指定した場合、指定のパスがリセットされます。

`-from <objects>` (オプション): 指定の遅延を割り当てるタイミング パスの始点を指定します。有効な始点は、プライマリ入力または入出力ポート、あるいは同期エレメントのクロック ピンです。クロックが指定されている場合は、そのクロックおよびそのクロックに接続されているレジスタのすべてのクロック ピンに関連するプライマリ入力および入出力ポートが始点として指定されます。

`-rise_from <objects>` (オプション): 立ち上がりエッジに指定の遅延を割り当てるタイミング パスの始点を指定します。

`-fall_from <objects>` (オプション): 立ち下がりエッジに指定の遅延を割り当てるタイミング パスの始点を指定します。

`-to <objects>` (オプション): 最小遅延を割り当てるパスのデスティネーション オブジェクトを指定します。有効な終点は、プライマリ出力または入出力ポート、あるいは同期エレメントのデータ ピンです。クロックが指定されている場合は、そのクロックおよびそのクロックに接続されているレジスタのすべてのデータ ピンに関連するプライマリ入力および入出力ポートが終点として指定されます。

`-rise_to <objects>` (オプション): 最小遅延を割り当てる立ち上がりエッジ パスのデスティネーション オブジェクトを指定します。

`-fall_to <objects>` (オプション): 最小遅延を割り当てる立ち下がりエッジ パスのデスティネーション オブジェクトを指定します。

`-through <objects>` (オプション): 最小遅延を割り当てるパスが通過するピン、セル、またはネットを指定します。

`-rise_through <objects>` (オプション): 最小遅延を割り当てる立ち上がりパスが通過するピン、セル、またはネットを指定します。

`-fall_through <objects>` (オプション): 最小遅延を割り当てる立ち下がりパスが通過するピン、セル、またはネットを指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<delay> (必須): 最小遅延値をナノ秒 (ns) で指定します。<delay> は、正または負の浮動小数点値としてナノ秒 (ns) で指定します。デフォルト値は 0 です。

例

次の例では、プライマリ入力ポートと出力ポートの間 (組み合わせ/フィードスルー パス) の最小遅延を 20 ns に設定しています。

```
set_min_delay 20 -from [all_inputs] -to [all_outputs]
```

次の例では、すべてのプライマリ出力ポートが終点となるタイミング パスの最小遅延を 20 ns に設定しています。

```
set_min_delay 20 -to [get_ports -filter {DIRECTION == out}]
```

関連項目

- [get_clocks](#)
- [get_nets](#)
- [get_ports](#)
- [report_timing](#)
- [set_max_delay](#)

set_msg_config

メッセージ ID、文字列、または重要度で指定したメッセージの表示および制御方法を設定します。

構文

```
set_msg_config [-id <arg>] [-string <args>] [-severity <arg>]
               [-limit <arg>] [-new_severity <arg>] [-suppress] [-regex] [-quiet]
               [-verbose]
```

使用法

名前	説明
[-id]	選択した操作を指定のメッセージ ID のメッセージのみに適用します (例: -id {Common 17-35})。デフォルトでは、すべての ID が指定されます。
[-string]	選択した操作を指定の文字列を含むメッセージのみに適用します。デフォルト: none
[-severity]	選択した操作したを指定の重要度レベルのメッセージのみに適用します。「-severity INFO」のように指定します。デフォルトでは、すべての重要度に一致します。
[-limit]	指定のメッセージの最大表示回数を整数で指定します。-id または -severity と共に使用する必要があります。
[-new_severity]	現在のプロジェクトで、指定のメッセージの重要度を変更します。
[-suppress]	現在のプロジェクトで、指定のメッセージを非表示にします。
[-regex]	-string で指定した値が正規表現であることを示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

現在のプロジェクトで Vivado ツールにより返されるメッセージの制御ルールを設定します。メッセージの重要度を変更したり、メッセージが表示される回数を制限したり、メッセージを非表示にできます。1 つの `set_msg_config` コマンドで実行できるのは、いずれか 1 つの操作のみです。

- ツールで表示されるメッセージの重要度を特定のレベルに変更します。たとえば、指定したメッセージ ID の重要度を「WARNING」から「ERROR」に変更できます。



重要: Vivado Design Suite の ERROR メッセージの重要度を下げることができません。

- デザインセッション中または 1 回の起動中にツールでメッセージが表示される最大回数を設定します。特定のメッセージ ID の最大表示回数、または特定の重要度のメッセージの最大表示回数を設定できます。



ヒント: すべてのメッセージ ID のデフォルトの最大表示回数は 100 で、`messaging.defaultLimit` パラメーターで定義されています。この最大表示回数が、ツールで返される各メッセージに適用されます。このパラメーターの現在の値を取得するには `get_param` コマンドを使用し、変更するには `set_param` コマンドを使用します。

- 特定のメッセージ ID がツールでレポートされないようにします。非表示になっているメッセージが表示されるようにするには、`reset_msg_config` コマンドを使用します。
- 1 つの `set_msg_config` コマンドで複数の操作を実行しようとする、エラーが返されます。

`set_msg_config` コマンドで設定するメッセージは、`-string`、`-id`、および `-severity` で指定します。少なくとも 1 つを使用してコマンドを適用するメッセージを指定する必要があります。これらのすべてのオプションの指定に一致するメッセージのみに設定が適用されます。



ヒント: `set_msg_config` では、メッセージの指定にワイルドカードは使用できません。

メッセージ制御ルールはプロジェクト特定であり、プロジェクトを閉じて開き直しても継続されます。



重要: メッセージ制御ルールは現在のプロジェクトに適用され、合成 run やインプリメンテーション run などの下位プロセスに自動的に渡されます。`set_msg_config` は、`tcl.pre` および `tcl.post` スクリプトでは使用しないでください。

現在のプロジェクトで特定のメッセージに対して定義されている現在の制御ルールを確認するには、`get_msg_config` コマンドを使用します。デフォルト設定に戻すには、`reset_msg_config` コマンドを使用します。

`set_msg_config` コマンドは、メッセージ マネージャーを介してメッセージを生成しないので、`report_cdc` コマンドではサポートされません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-id <arg>` (オプション): 指定の引数に一致するメッセージ ID を検索するメッセージ ID のパターンを指定します。指定の `<arg>` は検索パターンとして使用されます。指定のパターンに一致するすべてのメッセージ ID に対して `set_msg_config` コマンドが実行されます。ツールで表示される各メッセージには、アプリケーション サブシステムのコードとメッセージ識別子を含む固有のメッセージ ID が付けられています。次に、メッセージ ID の例を示します。

```
[Common 17-54]
[Netlist 29-28]
[Synth 8-32]
[Synth 8-3295]
```



ヒント: 特定のメッセージ ID を指定するには、その ID に特定のパターンを指定します。たとえば次の例では、最初のコマンドは「Synth 8-32」および「Synth 8-3295」の両方に適用され、2 番目のコマンドは「Synth 8-32」にのみ適用されます。

```
set_msg_config -id "Synth 8-32" -new_severity "CRITICAL WARNING"
set_msg_config -id {[Synth 8-32]} -new_severity "CRITICAL WARNING"
```

`-string <args>` (オプション): 指定した文字列を含むメッセージのみに選択した操作を適用します。文字列は波かっこ ({}) で囲んで指定します。複数の文字列はスペースで区切って指定します。

```
{{Vivado}} {Synthesis}}
```

注記: 大文字/小文字が区別されます。

`-severity` (オプション): メッセージの重要度を指定します。次の 5 つの重要度があります。

- **ERROR:** デザインの結果が使用不可となったり、ユーザーの操作がないと回避できないような問題が発生していることを示すエラー メッセージです。
- **{CRITICAL WARNING}:** 入力や制約に適用されないものがあったり、FPGA ファミリには適していないものがあることを示すクリティカル警告メッセージです。修正することが強く推奨されます。

注記: これは 2 単語の値なので、`"` または `{ }` で囲む必要があります。

- **WARNING:** 制約または仕様が指定どおりに適用されず、デザインが最適な結果にならない可能性を示す警告メッセージです。修正するかどうかは、ユーザーが判断します。
- **INFO:** STATUS メッセージと同じですが、重要度とメッセージ ID タグが含まれる点が異なります。INFO メッセージには、必要に応じてアンサー データベースで確認できるようにメッセージ ID が含まれます。
- **STATUS:** デザインの処理に関するプロセスおよびフィードバックのステータスを示すステータス メッセージです。STATUS メッセージには、メッセージ ID は含まれません。

注記: STATUS メッセージにはメッセージ ID がないので、重要度を変更することはできません。

`-limit <arg>` (オプション): 選択したメッセージの最大表示回数を 1 以上の整数で指定します。`-1` に設定すると、`messaging.defaultLimit` で指定されたデフォルトの最大表示回数にリセットされます。

`-new_severity <arg>` (オプション): 指定したメッセージの新しい重要度を指定します。有効な値は、`-severity` オプションに示されています。



重要: `-new_severity` オプションを `-id` または `-string` オプションと共に使用すると、ERROR メッセージの重要度を下げることができるように思われるかもしれませんが、ERROR メッセージの重要度を下げることができません。次にエラーが発生したときに、これが Vivado ツールにより正しくレポートされます。下の例を参照してください。

`-suppress` (オプション): 指定したメッセージがレポートされないようにします。



注意: WARNING などの指定した重要度のすべてのメッセージを非表示にすると、デザインで発生する可能性のあるクロック乗せ換え (CDC) に関するメッセージが非表示になることがあります。

`-regexp` (オプション): `-string` オプションと共に使用し、検索パターンが正規表現で記述されていることを指定します。正規表現の検索文字列は、常に検索文字列の先頭にアンカーされます。検索文字列の先頭または末尾に「`*`」を追加して、部分文字列を含めるよう検索を拡張できます。正規表現構文の詳細は、<http://perldoc.perl.org/perlre.html> を参照してください。

注記: Tcl ビルトイン コマンド `regexp` はアンカーされておらず、標準 Tcl コマンドと同様に機能します。詳細は、<http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm> を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例は、Common 情報メッセージの重要度を CRITICAL WARNING に変更します。

```
set_msg_config -id {[Common 17-81]} -new_severity "CRITICAL WARNING"
```



重要: 次の例では、Common 17-69 メッセージは ERROR メッセージであり、重要度は変更できません。この例のコマンドを Tcl コンソールから実行しても、何も変更されません。

```
set_msg_config -id {[Common 17-69]} -new_severity WARNING
```

次に Common 17-69 メッセージが表示されるときに、エラーの重要度を下げることができないことを示す警告メッセージが表示され、メッセージは ERROR として表示されます。

```
WARNING: [Common 17-239] ERROR Messages are prohibited to be downgraded.
Message 'Common 17-69' is not downgraded.
ERROR: [Common 17-69] Command failed: report_design_analysis
-critical_paths can be run only after synthesis has successfully
completed.
```

次の例は、メッセージ ID 17-35 の警告メッセージで、文字列「clk」が含まれるメッセージをエラーに変更します。

```
set_msg_config -severity warning -string "clk" -id "17-35" \
-new_severity error
```

次の例は、指定したメッセージ ID のメッセージの重要度を変更し、現在のメッセージ制御ルールを取得して、特定のルールをリセットする 2 つのコマンドを示します。

```
set_msg_config -id "Common 17-361" -severity INFO -new_severity WARNING
get_msg_config -rules
-----
Message control rules currently in effect are:
Rule Name Rule Current
Message Count
1 set_msg_config -ruleid {1} -id {Common 17-361} -severity {INFO} -
new_severity {WARNING} 0
-----
reset_msg_config -id "Common 17-361" -default_severity
reset_msg_config -ruleid {1}
```



ヒント: 上記の例でメッセージをリセットするのに必要なのは、いずれか 1 つの `reset_msg_config` コマンドのみです。

次の例は、パラメーターを使用してメッセージのデフォルトの最大表示回数を変更し、指定したメッセージ ID の新しい最大表示回数を定義します。

```
get_param messaging.defaultLimit
100
set_param messaging.defaultLimit 1000
set_msg_config -id {[Common 17-81]} -limit 1500
```

関連項目

- [get_msg_config](#)

- [get_param](#)
- [reset_msg_config](#)
- [set_param](#)

set_multicycle_path

マルチサイクル パスを定義します。

構文

```
set_multicycle_path [-setup] [-hold] [-rise] [-fall] [-start] [-end]
                    [-reset_path] [-from <args>] [-rise_from <args>] [-fall_from <args>]
                    [-to <args>] [-rise_to <args>] [-fall_to <args>] [-through <args>]
                    [-rise_through <args>] [-fall_through <args>] [-quiet] [-verbose]
                    <path_multiplier>
```

使用法

名前	説明
[-setup]	セットアップのみの乗数を設定します。
[-hold]	ホールドのみの乗数を設定します。
[-rise]	パスの終点の立ち上がりエッジ遅延にパスの乗数を適用します。
[-fall]	パスの終点の立ち下がりエッジ遅延にパスの乗数を適用します。
[-start]	パスの乗数を適用するパスの始点を指定します。
[-end]	パスの乗数を適用するパスの終点を指定します。
[-reset_path]	マルチサイクルを適用する前に、指定のパスをリセットします。
[-from]	パスの始点またはクロックを指定します。
[-rise_from]	始点またはクロックから立ち上がるパスに適用します。
[-fall_from]	始点またはクロックから立ち下がるパスに適用します。
[-to]	パスの終点またはクロックを指定します。
[-rise_to]	終点またはクロックで立ち上がるパスに適用します。
[-fall_to]	終点またはクロックで立ち下がるパスに適用します。
[-through]	通過ピン、セル、またはネットを指定します。
[-rise_through]	ピン、セル、またはネットを立ち上がりで通過するパスに適用します。
[-fall_through]	ピン、セル、またはネットを立ち下がりで通過するパスに適用します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<path_multiplier>	サイクル数を指定します。

カテゴリ

[SDC](#)、[XDC](#)

説明

デフォルトでは、Vivado タイミング エンジンでシングル サイクル解析が実行され、セットアップ チェックがデスティネーション エッジのデスティネーションでソース クロックのエッジの 1 クロック サイクル後に実行されます。これは、タイミング パスによっては適切でない場合があります。たとえば、データが終点で安定するまでに複数クロック サイクル必要なロジック パスなどです。

`set_multicycle_path` コマンドは、パスの乗数 N を指定し、タイミング パスで送信クロック エッジから受信クロック エッジまでに N クロック サイクルかかることを示します。パスの乗数は、信号を始点から終点に伝搬するために複数クロック サイクル必要な場合に、信号の伝搬に必要なクロック サイクル数を定義します。このコマンドの使用法の詳細は、『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903) を参照してください。

`set_multicycle_path` コマンドは、セットアップ解析およびホールド解析用に、ソース クロックまたはデスティネーション クロックに対して、立ち上がりエッジまたは立ち下がりエッジにパスの乗数を指定します。このコマンドには、3 つの要素が含まれます。

- マルチサイクル パスが適用されるセットアップおよびホールド解析の指定。
- マルチサイクル パスが適用されるタイミング パスのデスティネーション。
- タイミング解析に適用するクロック サイクル数を定義するパスの乗数。

デフォルトでは、パスの乗数はセットアップ解析およびホールド解析の両方に適用されます。ホールド解析はセットアップ解析から導出されるので、セットアップ解析と共に移動されます。パスの乗数によりセットアップチェックが N クロック サイクル移動する場合は、ホールド チェックは $N-1$ クロック サイクル移動しますが、これにより、ホールド タイミング エラーが発生することがよくあります。

2 つ目の `set_multicycle_path` コマンドを `-hold` オプションを指定して使用し、ホールド解析を元の位置に戻すことができます。`-hold` オプションを指定すると、`<path_multiplier>` がホールド関係に適用され、ホールドチェックが元の位置に戻されます。次のコマンド例では、最初のコマンドでセットアップチェックを 3 クロック サイクル拡張し、2 つ目のコマンドでホールド チェックを 2 クロック サイクル ($N-1$) 拡張しています。2 つ目のコマンドでは、ホールド チェックを元の位置に戻しています。

```
set_multicycle_path 3 -from {usbEngine1/u4/csr_reg[26]/C} \
-to {usbEngine1/u1/u2/sizd_c_reg[12]/D}
set_multicycle_path 2 -from {usbEngine1/u4/csr_reg[26]/C} \
-to {usbEngine1/u1/u2/sizd_c_reg[12]/D} -hold
```

デフォルトでは、セットアップ パスの乗数はデスティネーション クロックに対して適用され、ホールド パスの乗数はソース クロックに対して適用されます。`-start` または `-end` オプションを使用すると、デフォルトのセットアップ解析またはホールド解析をソース クロックまたはデスティネーション クロックを基準にするよう変更できます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-setup` (オプション): パスの乗数をセットアップ チェックに適用します。これは、ホールド チェックにも影響します。`set_multicycle_path` コマンドで `-setup` または `-hold` オプションを指定しない場合、これがデフォルトです。

`-hold` (オプション): パスの乗数をホールド チェックに適用し、ホールド関係を指定のクロック サイクル数分変更します。

注記: `-setup` または `-hold` オプションのどちらも使用しない場合、`-setup` オプションのみを指定した場合は、`<path_multiplier>` はセットアップ チェックおよびホールド チェックの両方に適用されます。

`-rise` (オプション): パスの終点の立ち上がりエッジ遅延にパスの乗数を適用します。

`-fall` (オプション): パスの終点の立ち下がりエッジ遅延にパスの乗数を適用します。

注記: `-rise` または `-fall` のどちらも指定しない場合、乗数は立ち上がりエッジ遅延および立ち下がりエッジ遅延の両方に適用されます。

`-start` (オプション): デフォルトでは、セットアップ パスの乗数はデスティネーション クロック (`-end`) に対して定義されます。セットアップ要件をソース クロックに対して変更するには、`-start` オプションを使用する必要があります。

`-end` (オプション): デフォルトでは、ホールド パスの乗数はソース クロックに対して定義されます。ホールド要件をデスティネーション クロックに対して変更するには、`-end` オプションを使用する必要があります。

注記: `-start`/`-end` オプションは、同じクロック、同じ波形の 2 つのクロック、または位相シフトのないクロックが供給されるパスにマルチサイクル パス制約を適用する場合には影響がありません。

`-reset_path` (オプション): マルチサイクルの `path_multiplier` 値を適用する前に、指定のパスをリセットします。

`-from <args>` (オプション): パスの乗数を適用するパスの始点を指定します。

`-rise_from <args>` (オプション): マルチサイクル パスの乗数を適用する立ち上がりエッジ パスの始点を指定します。

`-fall_from <args>` (オプション): マルチサイクル パスの乗数を適用する立ち下がりエッジ パスの始点を指定します。

`-to <args>` (オプション): マルチサイクル パスの乗数を適用するパスの終点を指定します。

`-rise_to <args>` (オプション): マルチサイクル パスの乗数を適用する立ち上がりエッジ パスの終点を指定します。

`-fall_to <args>` (オプション): マルチサイクル パスの乗数を適用する立ち下がりエッジ パスの終点を指定します。

`-through <args>` (オプション): マルチサイクル パスの乗数を割り当てるパスが通過するピン、セル、またはネットを指定します。

`-rise_through <args>` (オプション): マルチサイクル パスの乗数を割り当てる立ち上がりパスが通過するピン、セル、またはネットを指定します。

`-fall_through <args>` (オプション): マルチサイクル パスの乗数を割り当てる立ち下がりパスが通過するピン、セル、またはネットを指定します。



重要: `-to`、`-through`、および `-from` (およびそのバリエーション) はオプションですが、`set_multicycle_path` 制約のタイミング パスを定義するため `-from`、`-to`、または `-through` のいずれかを指定する必要があります。指定しないと、エラーが返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<path_multiplier>` (必須): セットアップおよびホールド解析を移動するクロック サイクル数を指定します。

注記: `-hold` オプションを指定すると、`<path_multiplier>` がホールド関係に適用され、ホールド チェックが元の位置に戻されます。

例

次の例では、`-from/-to` オプションで定義したタイミング パスのセットアップ チェックに対して 3 クロック サイクルのパスの乗数を定義しています。パスの乗数 $N-1$ (この例では 2) が、同じタイミング パスのホールド チェックをデクリメントするに使用されます。

```
set_multicycle_path 3 -setup -from [get_pins data0_reg/C] \
    -to [get_pins data1_reg/D]
set_multicycle_path 2 -hold -from [get_pins data0_reg/C] \
    -to [get_pins data1_reg/D]
```

注記: セットアップ解析とホールド解析の関係については、『Vivado Design Suite ユーザー ガイド: 制約の使用』(UG903) を参照してください。

関連項目

- [report_timing](#)
- [report_timing_summary](#)
- [set_input_delay](#)
- [set_output_delay](#)

set_operating_conditions

消費電力見積もりの動作条件を設定します。

構文

```
set_operating_conditions [-voltage <args>] [-grade <arg>] [-process <arg>]
    [-junction_temp <arg>] [-ambient_temp <arg>] [-thetaja <arg>]
    [-thetasa <arg>] [-airflow <arg>] [-heatsink <arg>] [-thetajb <arg>]
    [-board <arg>] [-board_temp <arg>] [-board_layers <arg>]
    [-design_power_budget <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-voltage]	電圧ペアを指定します (例: {name value})。サポートされる電圧は、ファミリによって異なります。
[-grade]	電圧グレードを指定します。サポートされる値は、ファミリによって異なります。デフォルトは commercial です。
[-process]	プロセス データを指定します。有効な値は typical または maximum で、デフォルトは typical です。
[-junction_temp]	ジャンクション温度 (C) を指定します。有効な値は auto または実際の温度 (C) で、デフォルトは auto です。
[-ambient_temp]	周囲温度 (C) を指定します。有効な値は default または実際の温度 (C) で、デフォルトは default です。
[-thetaja]	ThetaJA (C/W) を指定します。有効な値は auto または実際の値 (C/W) で、デフォルトは auto です。
[-thetasa]	ThetaSA (C/W) を指定します。有効な値は auto または実際の値 (C/W) で、デフォルトは auto です。
[-airflow]	エアフロー (LFM) を指定します。有効な値は 0 ~ 750 で、デフォルトはファミリによって異なります。
[-heatsink]	ヒートシンクのサイズを指定します。有効な値は none、low、medium、high、custom で、デフォルトは medium です。
[-thetajb]	ThetaJB (C/W) を指定します。有効な値は auto または実際の値 (C/W) で、デフォルトは auto です。
[-board]	ボード タイプを指定します。有効な値は jedec、small、medium、large、custom で、デフォルトは medium です。
[-board_temp]	ボード温度 (C) を取得します。
[-board_layers]	ボードの層数を指定します。有効な値は 4to7、8to11、12to15、16+ で、デフォルトは 8to11 です。
[-design_power_budget]	デザインの消費電力バジェット (W) を指定します。デフォルトでは指定されません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC、Power (電力)

説明

デザインのパフォーマンスを解析する際に使用される実際の動作条件を設定します。デバイスの環境動作条件は、`report_power` コマンドによる消費電力解析で使用されます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

動作条件をデフォルト値に戻すには、`reset_operating_conditions` コマンドを使用します。

動作条件を確認するには、`report_operating_conditions` コマンドを使用します。

引数

`-voltage <arg>` (オプション): 電源の名前とその値のペアを指定します。サポートされる電源名およびその値は、デバイス ファミリによって異なります。たとえば、デバイス ファミリで `Vccint` という名前の電源がサポートされる場合、この電源の値を 0.8 に設定するには、「`-voltage {Vccint 0.8}`」と指定します。

注記: ターゲット デバイスの有効な動作範囲外の電圧を指定すると、`set_operating_conditions` コマンドにより指定した電圧に対応するようデバイスのスピード グレードが変更されることがあります。これは、タイミング解析に影響する可能性があります。UltraScale デバイスでは、`Vccint` 電圧を変更すると、指定された電圧レベルで示されているように、Vivado ツールにより自動的にデバイスが低電圧デバイスに、または低電圧デバイスから通常のデバイスに変更されます。

`-grade <arg>` (オプション): ターゲット デバイスの温度グレードを指定します。サポートされる値は、ファミリによって異なります。デフォルト値は `commercial` です。

`-process <arg>` (オプション): 製造プロセス特性を指定します。有効な値は、`typical` または `maximum` です。デフォルト値は `typical` です。

`-junction_temp <arg>` (オプション): モデリングに使用するデバイスのジャンクション温度を指定します。有効な値は `auto` または実際の温度 (摂氏) で、デフォルト値は `auto` です。

`-ambient_temp <arg>` (オプション): 環境周囲温度を摂氏で指定します。デフォルト値は `default` です。

`-thetaja <arg>` (オプション): モデリングに使用する Theta-JA 熱抵抗を C/W で指定します。デフォルト値は `auto` です。

`-theta_sa <arg>` (オプション): モデリングに使用する Theta-SA 熱抵抗を C/W で指定します。デフォルト値は `auto` です。

`-airflow <[0:750]>` (オプション): モデリングに使用するエアフローを LFM (リニア フィート/分) で指定します。デフォルト値はデバイス ファミリによって異なります。

`-heatsink <arg>` (オプション): モデリングに使用するヒートシンクのプロファイルを指定します。有効な値は `none`、`low`、`medium`、`high`、`custom` で、デフォルト値は `medium` です。

`-theta_jb <arg>` (オプション): モデリングに使用する Theta-JB 熱抵抗を C/W で指定します。デフォルト値は `auto` です。

`-board <arg>` (オプション): モデリングに使用するボードのサイズを指定します。有効な値は `jedec`、`small`、`medium`、`large`、`custom` です。デフォルト値は `medium` です。

`-board_temp <arg>` (オプション): モデリングに使用するボードの温度を摂氏で指定します。

`-board_layers <arg>` (オプション): モデリングに使用するボードの層数を指定します。有効な値は、4 ~ 7 層のボードでは `[4to7]`、8 ~ 11 層のボードでは `[8to11]`、12 ~ 15 層のボードでは `[12to15]`、16 層以上のボードでは `16+` です。デフォルト値は `12to15` です。

`-design_power_budget <arg>` (オプション): デザインの消費電力バジェットをワットで指定します。この値は、`report_power` コマンドで計算されたオンチップ消費電力とデザインの消費電力バジェットとの差をレポートするのに使用されます。このオプションを指定しない場合、差はレポートされません。デフォルトでは指定されません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、インダストリアル グレードのデバイスに対して周囲動作温度を 75°C に設定しています。

```
set_operating_conditions -grade industrial -ambient_temp 75
```

次の例では、電源 `Vccaux` の値を 1.9 に設定しています。

```
set_operating_conditions -voltage {Vccaux 1.89}
```

次の例では、製造プロセス コーナーを `maximum` に設定しています。

```
set_operating_conditions -process maximum
```

次の例では、製造プロセス コーナーを `maximum` に、電源 `Vccint` の値を 0.875 に設定しています。

```
set_operating_conditions -process maximum -voltage {Vccint 0.875}
```

関連項目

- [report_operating_conditions](#)
- [report_power](#)
- [reset_operating_conditions](#)

set_output_delay

ポートの出力遅延を設定します。

構文

```
set_output_delay [-clock <args>] [-reference_pin <args>] [-clock_fall]
                 [-rise] [-fall] [-max] [-min] [-add_delay] [-network_latency_included]
                 [-source_latency_included] [-quiet] [-verbose] <delay> <objects>
```

使用法

名前	説明
[-clock]	相対クロックを指定します。
[-reference_pin]	相対ピンまたはポートを指定します。
[-clock_fall]	遅延がクロックの立ち下がりエッジを基準にすることを示します。
[-rise]	立ち上がり遅延を指定します。
[-fall]	立ち下がり遅延を指定します。
[-max]	最大遅延を指定します。
[-min]	最小遅延を指定します。
[-add_delay]	既存の入力遅延を削除しません。
[-network_latency_included]	クロックのネットワーク レイテンシが既に含まれていることを示します。
[-source_latency_included]	クロックのソース レイテンシが既に含まれていることを示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<delay>	遅延値を指定します。
<objects>	ポートを指定します。

カテゴリ

SDC、XDC

説明



ヒント: Vivado IDE の [XDC] → [Timing Constraints] 言語テンプレートおよび Timing Constraints ウィザードでは、特定のタイミング制約の定義に関するタイミング図と詳細が提供されます。これらを参照して追加情報を得ることができます。

デザインのインターフェイスのクロック エッジに対するプライマリ出力ポートの外部システム レベル遅延を指定します。出力遅延値はナノ秒 (ns) で指定し、FPGA デバイス外のクロックおよびデータの相対位相によって、正または負の値を指定できます。

ザイリンクス FPGA デザインのシステム レベル タイミングを正確にモデリングするには、FPGA 外部のオブジェクトのタイミング遅延をデザインのプライマリ入力または出力ポートに割り当てする必要があります。これらの遅延は、`set_input_delay` および `set_output_delay` コマンドを使用して定義します。



重要: 出力ポートに `set_max_delay` 制約も設定されている場合は、指定した出力遅延値は `max_delay` の算出の一部として考慮されます。つまり、出力遅延は出力ポートを含むタイミング パス上の最大遅延の一部を占めるということです。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-clock <arg>` (オプション): 遅延が指定のクロックの立ち上がりエッジを基準にすることを指定します。

`-reference_pin <arg>` (オプション): 遅延がクロックではなく指定のピンを基準にすることを指定します。

`-clock_fall` (オプション): 遅延が指定のクロックの立ち上がりエッジではなく立ち下がりエッジを基準にすることを示します。

`-rise` (オプション): 遅延が立ち上がりエッジ用であることを示します。

`-fall` (オプション): 遅延が立ち下がりエッジ用であることを示します。

`-max` (オプション): 指定の遅延を最大しきい値として処理します。

`-min` (オプション): 指定の遅延を最小しきい値として処理します。

`-add_delay` (オプション): ポートに指定の遅延制約を追加し、ポートに既に定義されているほかの `set_output_delay` 制約と共存させます。デフォルトでは、既存の遅延が置き換えられます。

`-network_latency_included` (オプション): 遅延値に基準クロックのクロック ネットワーク レイテンシが含まれることを示します。指定の入力または出力遅延値にソース レイテンシまたはネットワーク レイテンシが含まれていない場合、Vivado タイミング エンジンでは、クロック エッジがクロック レイテンシの後にキャプチャ フリップフロップに到達すると考慮されます。

`-source_latency_included` (オプション): 指定の遅延値に基準クロックのソース レイテンシが含まれることを示します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<delay>` (オプション): 指定のポートに適用する遅延をナノ秒 (ns) で指定します。有効な値は浮動小数点値で、デフォルト値は 0 です。

`<objects>` (必須): 遅延値を適用するポートを 1 つまたは複数指定します。

例

次の例では、指定のクロックに対してポートの出力遅延を設定しています。

```
set_output_delay 5.0 -clock [get_clocks cpuClk] [get_ports]
```

次の例は上記の例とほぼ同じですが、ネットワーク レイテンシが含まれています。

```
set_output_delay 5.0 -clock [get_clocks cpuClk] \  
-network_latency_included [get_ports]
```

次の例では、clk_dds という名前のクロックを作成し、デバイス外にあるクロックの立ち上がりエッジおよび立ち下がりがエッジの両方でデータが送信されてから、クロックの立ち上がりエッジおよび立ち下がりがエッジの両方で動作する内部フリップフロップのデータ出力までの出力遅延制約を作成しています。

```
create_clock -name clk_dds -period 6 [get_ports DDR_CLK_IN]  
set_output_delay -clock clk_dds -max 2.1 [get_ports DDR_OUT]  
set_output_delay -clock clk_dds -max 1.9 [get_ports DDR_OUT] -clock_fall -  
add_delay  
set_output_delay -clock clk_dds -min 0.9 [get_ports DDR_OUT]  
set_output_delay -clock clk_dds -min 1.1 [get_ports DDR_OUT] -clock_fall -  
add_delay
```

注記: -add_delay オプションが使用されているので、新しい最小および最大遅延制約は同じポートの最初の遅延と共存します。

関連項目

- [all_clocks](#)
- [create_clock](#)
- [get_ports](#)
- [set_max_delay](#)
- [set_input_delay](#)

set_package_pin_val

1 つまたは複数のパッケージ ピンのユーザー列を設定します。

構文

```
set_package_pin_val [-quiet] [-verbose] <column> <value> <package_pins>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><column></code>	ユーザー列名を指定します。
<code><value></code>	設定する値を指定します。
<code><package_pins></code>	パッケージ ピン名を指定します。

カテゴリ

[XDC、PinPlanning \(ピン プランニング\)](#)

説明

ユーザー定義のパッケージ ピン属性を作成し、値をパッケージの特定のピンに割り当てます。

ユーザー定義のピン属性は CSV ファイルで定義でき、`read_csv` コマンドを使用して I/O ピン プランニング プロジェクトにインポートするか、このコマンドを使用してプロジェクトで変更できます。

注記: パッケージ ピンのツールで定義されたプロパティを設定するには、`set_property` コマンドを使用します。

引数

`<column>` (必須): ユーザー定義の列名を指定します。列名では、大文字/小文字が区別されます。指定した列が存在しない場合は、パッケージ ピンに新しい属性が作成されます。指定したユーザー定義列名が既に存在する場合は、指定した値が指定したピンに割り当てられます。

注記: 列は、ツールの GUI の [Package Pins] ウィンドウに表示される属性の列です。指定したパッケージ ピンの属性は、`write_csv` コマンドでエクスポートするなどできます。

`<value>` (必須): 指定の列に割り当てる値を指定します。同じ列に対して異なるピンに異なる値を割り当てる場合は、`set_package_pin_val` コマンドを繰り返し使用します。

`<package_pins>` (必須): 値を割り当てるパッケージ ピンを指定します。パッケージ ピンの指定には `get_package_pins` コマンドを使用できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、[Package Pins] ウィンドウに新しいユーザー定義列を追加し、指定のピンに値 `true` を割り当てています。

```
set_package_pin_val -column track1 -value true -package_pins AK27
```

次の例では、`Test` というユーザー定義列を作成し、すべての AK パッケージ ピンに値 `RED` を割り当て、指定した 3 つのピンの値を `GREEN` に変更しています。

```
set_package_pin_val -column Test -value RED \
    -package_pins [get_package_pins AK*]
set_package_pin_val -column Test -value GREEN \
    -package_pins {AK1 AK2 AK3}
```

関連項目

- [get_package_pins](#)
- [set_property](#)
- [write_csv](#)

set_param

パラメーター値を設定します。

構文

```
set_param [-quiet] [-verbose] <name> <value>
```

戻り値

新しく設定されたパラメーター値

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><name></code>	パラメーター名を指定します。
<code><value></code>	パラメーター値

カテゴリ

[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

ユーザーが定義可能なパラメーターの値を設定します。これらのパラメーターを使用すると、ツールのさまざまな動作を設定および制御できます。現在定義されているパラメーターの説明は、`report_param` コマンドを参照してください。

たとえば、Vivado Design Suite の `general.maxThreads` パラメーターを定義できます。マルチプロセッサ システムでは、DRC レポート、スタティック タイミング解析、配置、配線などのプロセスを高速化するため、マルチスレッドが使用されます。すべてのタスクに適用されるデフォルトの制限があり、Windows システムでは 2、Linux システムでは 8 です。この制限を変更するには、次のコマンドを使用します。

```
set_param general.maxThreads <value>
```

`<value>` には 1 ～ 8 の整数値を指定します。

同時に使用可能な最大スレッド数は、実行するタスクによっても異なります。これらのプロセスを実行する前に `maxThreads` パラメーターを変更できます。特定の Tcl コマンドの最大スレッド数は、次のとおりです。

- `phys_opt_design`: 8
- `place_design`: 8
- `report_drc`: 8
- `report_timing` および `report_timing_summary`: 8
- `route_design`: 8

- synth_design: 4

reset_param コマンドを使用すると、変更されたパラメーターの値をデフォルト設定に戻すことができます。

注記: 指定したパラメーターの値を -1 に設定すると、その機能をオフにできます。

引数

<name> (必須): 値を設定するパラメーターの名前を指定します。このコマンドで値を設定できるのは、一度に 1 つのパラメーターのみです。

<value> (必須): 指定したパラメーターに設定する値を指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、配置、配線、タイミング解析で、マルチスレッド プロセスで実行するスレッド数を定義するパラメーターを設定しています。

```
set_param general.maxThreads 4
```

注記: Vivado ツールでは、1 ～ 8 のスレッドがサポートされます。現在の設定を確認するには、get_param コマンドを使用します。

次の例では、メッセージの最大表示回数の新しいデフォルト値を設定しています。

```
set_param messaging.defaultLimit 1000
```

関連項目

- [get_param](#)
- [list_param](#)
- [report_param](#)
- [reset_param](#)

set_part

現在のプロジェクトのパーツを設定します。プロジェクトが開いていない場合は、ディスクに保存されていないプロジェクトが作成されます。

構文

```
set_part [-quiet] [-verbose] <part>
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<part>	現在のプロジェクトに設定するパーツを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)

説明

この後実行するエラボレーション、合成、インプリメンテーション、および解析に使用する現在のプロジェクトのパーツを変更します。



ヒント: パーツは現在のプロジェクトに対してのみ変更され、メモリ内のデザインでは変更されません。タイミング解析用にメモリ内のデザインに対してデバイスのスピード グレードを変更するには、`set_speed_grade` コマンドを使用します。既存のデザイン チェックポイントを開く際にパーツを変更するには、`open_checkpoint` または `read_checkpoint` コマンドで `-part` オプションを使用します。

このコマンドは、非プロジェクト ベース デザインでメモリ内のプロジェクトのパーツを変更するためのもので、プロジェクト ベース デザインでは使用できません。プロジェクト モードのデザインでは、次のようにプロジェクトの PART プロパティを設定します。

```
set_property PART xc7vx485tffg1158-2 [current_project]
```

使用可能なパーツのリストを取得するには、`get_parts` コマンドを使用します。

`set_part` コマンドは、非プロジェクト ベース デザインのインメモリ プロジェクトを作成するか、既存のインメモリ プロジェクトのパーツを設定します。

注記: プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892)を参照してください。

このコマンドを実行すると、インメモリ プロジェクトで使用するよう設定されたパーツが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<part> (必須): 現在のプロジェクトまたはインメモリ デザインで使用されるパーツを指定します。

例

次の例では、現在のメモリ内のプロジェクトのパーツを変更しています。

```
set_part xc7vx485tffg1158-2
```

関連項目

- [create_project](#)
- [get_parts](#)
- [open_checkpoint](#)
- [read_checkpoint](#)
- [set_property](#)
- [set_speed_grade](#)

set_power_opt

消費電力の最適化用に制約を設定します。

構文

```
set_power_opt [-include_cells <args>] [-exclude_cells <args>]  
              [-clocks <args>] [-cell_types <args>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-include_cells]</code>	クロック ゲーティングに含めるインスタンスを指定します。デフォルトは all です。
<code>[-exclude_cells]</code>	クロック ゲーティングから除外するインスタンスを指定します。デフォルト: none
<code>[-clocks]</code>	指定したクロックが供給されるインスタンスにクロック ゲーティングを適用します。デフォルトでは、すべてのクロックが指定されます。
<code>[-cell_types]</code>	クロック ゲーティングを適用するセル タイプを指定します。all または none、あるいは bram、reg、srl の 1 つ以上を指定します。デフォルトは all です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

電力、XDC

説明

消費電力最適化に含めるまたは除外するセル インスタンスを指定します。指定したセルは、power_opt_design コマンドで最適化されます。



ヒント: ブロック RAM の最適化は、opt_design コマンドでデフォルトで実行されます。set_power_opt コマンドを使用して、一部またはすべての BRAM セルを opt_design の最適化から除外することもできます。

set_power_opt コマンドを複数回実行した場合、設定は累積的であり、まず最適化するセル タイプを幅広く設定し、特定の階層セルを含めて、その階層内のセルを除外するなど、消費電力最適化の対象を調整できます。

実行された消費電力最適化は、report_power_opt コマンドを使用してレポートできます。

引数

`-include_cells <args>` (オプション): 指定したインスタンスのみをクロック ゲーティングに含めます。このオプションは、power_opt_design を使用して最適化する特定のセルまたはブロックをリストするために使用します。デフォルトでは、すべてのセルで消費電力最適化が実行されます。

`-exclude_cells <args>` (オプション): 指定したインスタンスをクロック ゲーティングから除外します。デフォルトでは、消費電力最適化から除外されるセルはありません。`-exclude_cells` オプションは、現在含まれているセルから除外します。デフォルトではすべてのセルが含まれますが、`-include_cells` が指定されている場合は、`-exclude_cells` は現在含まれているセルのみに適用されます。

`-clocks <args>` (オプション): 指定のクロックが供給されるインスタンスのみに消費電力最適化を実行します。デフォルトでは、デザインのすべてのクロックが含まれます。

注記: `-clocks` と `-include_cells` の両方を使用して、指定のクロックが供給されないセルのリストを生成できますが、消費電力最適化は実行されません。

`-cell_types [all | bram | reg | srl | none]` (オプション): 消費電力最適化を指定したセル タイプでのみ実行します。デフォルトでは、すべてのタイプのセルに対して消費電力最適化が実行されます。`all` または `none` を使用して、以前の `set_power_opt` コマンドの設定をリセットまたはクリアできます。`bram`、`srl`、または `reg` タイプのセルを 1 つ以上を指定することも可能です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、BRAM セルのみに対して消費電力最適化を設定し、消費電力最適化を実行しています。

```
set_power_opt -cell_types bram
power_opt_design
```

次の例では、BRAM および REG セルに対して消費電力最適化を設定し、SRL を追加して、消費電力化を実行しています。その後セルをクリアし、SRL のみを含めて消費電力化を再実行しています。

```
set_power_opt -cell_types { bram reg}
set_power_opt -cell_types { srl}
power_opt_design
set_power_opt -cell_types { none}
set_power_opt -cell_types { srl}
power_opt_design
```

次の例では、BRAM セルにのみに対して消費電力最適化を設定し、cpuEngine ブロックを最適化から除外し、cpuEngine/cpu_dbg_dat_i ブロックを追加して消費電力化を実行しています。

```
set_power_opt -cell_types bram
set_power_opt -exclude_cells cpuEngine
set_power_opt -include_cells cpuEngine/cpu_dbg_dat_i
power_opt_design
```

関連項目

- [opt_design](#)
- [power_opt_design](#)

- [report_power_opt](#)

set_propagated_clock

伝搬されるクロック レイテンシを指定します。

構文

```
set_propagated_clock [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	クロック、ポート、またはピンを指定します。

カテゴリ

SDC、XDC

説明

クロック レイテンシをクロック ネットワーク全体に伝搬します。これにより、クロック ネットワーク全体でより正確なスキューおよびタイミング結果が得られます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): レイテンシを伝搬するクロック オブジェクトを指定します。

例

次の例では、最上位のプライマリ システム クロックを伝搬するよう指定しています。

```
set_propagated_clock [get_clocks top/clk]
```

次の例では、sublevel1 からのすべてのクロックを伝搬するよう指定しています。

```
set_propagated_clock [get_clocks sublevel1/*]
```

関連項目

- [get_clocks](#)
- [create_clock](#)

set_property

オブジェクトのプロパティを設定します。

構文

```
set_property [-dict <args>] [-quiet] [-verbose] <name> <value> <objects>...
```

使用法

名前	説明
[-dict]	設定するプロパティの名前と値のペアを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<name>	設定するプロパティの名前を指定します。-dict オプションと共に使用することはできません。
<value>	設定するプロパティの値を指定します。-dict オプションと共に使用することはできません。
<objects>	プロパティを設定するオブジェクトの名前を指定します。

カテゴリ

[Object \(オブジェクト\)](#)、[PropertyAndParameter \(プロパティおよびパラメーター\)](#)、[XDC](#)

説明

指定のオブジェクトにプロパティと値を設定します。

このコマンドは、デザインのオブジェクトに対してプロパティを定義するために使用できます。各オブジェクトにはあらかじめ定義されたプロパティがあり、指定可能な値または値の範囲があります。これらのプロパティの値を指定するには、set_property コマンドを使用します。オブジェクトのプロパティに設定されている値を確認するには、report_property、list_property、または list_property_values コマンドを使用します。

オブジェクトには、固有の <name> と <value> のペアを指定することにより、カスタム プロパティも設定できます。オブジェクトにカスタム プロパティがある場合、report_property または list_property コマンドのレポートに含まれます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。



ヒント: get_property コマンドを使用すると、オブジェクトに設定されたプロパティを確認できます。

引数

-dict (オプション): 1 つの set_property コマンドで、1 つのオブジェクトに対して複数のプロパティ (<name> と <value> のペア) のディクショナリを指定する際に使用します。<name> と <value> のペアは、ダブルクォーテーション (") または波かっこ ({}) で囲んで指定します。

```
-dict "name1 value1 name2 value2 ... nameN valueN"
```



重要: `save_constraints`、`save_constraints_as`、または `write_xdc` コマンドを使用してデザインの制約を記述する際、`-dict` オプションを使用して指定されたプロパティは、各名前/値ペアが個別の `set_property` コマンドとして記述されます。XDC 制約がこのように拡大しないようにするには、非プロジェクト デザインで Tcl スクリプト ドリブの方法を使用するか、Tcl スクリプトを制約セットのデザイン ソースとして使用します。非プロジェクト モードのデザインの詳細は『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892)、制約セットでの Tcl スクリプトの使用については『Vivado Design Suite ユーザー ガイド: 制約の使用』(UG903)を参照してください。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<name>` (必須): オブジェクトに設定するプロパティの名前を指定します。`<name>` では大文字/小文字が区別されるので、適切に指定してください。

`<value>` (必須): 指定のオブジェクトの `<name>` プロパティに設定する値を指定します。プロパティ タイプに対して値が有効であるかどうかチェックされます。値が指定のプロパティに有効でない場合は、エラーが返されます。



重要: プロパティの値にダッシュ (-) などの特殊文字が含まれる場合があり、これによってダッシュ以降の値がコマンドの別のオプションと解釈されてしまう場合があります。この場合、ここに説明するように、位置引数 (`name`、`value`、`objects`) ではなく明示的なオプション (`-name`、`-value`、`-objects`) を使用してください。これを下の例に示します。

`<objects>` (必須): プロパティを割り当てる 1 つまたは複数のオブジェクトを指定します。

例

次の例は、セル オブジェクトに `TRUTH` というユーザー定義のブール型プロパティを作成し、セルに設定します。

```
create_property -type bool truth cell
set_property truth false [lindex [get_cells] 1]
```

次の例は、`-dict` オプションを使用して、現在のデザインに複数のプロパティを一度に設定します。

```
set_property -dict "POST_CRC enable POST_CRC_ACTION correct_and_continue"
\
[current_design]
```

次の例は、現在のファイルセットの TOP プロパティを設定し、プロジェクトの最上位モジュールを定義します。

```
set_property top fftTop [current_fileset]
set_property top_file {C:/Data/sources/fftTop.v} [current_fileset]
```

注記: 最上位モジュールを定義するには、現在のプロジェクトのソース ファイルセットに含まれる階層ブロックに TOP プロパティを設定する必要があります。上記の例では、TOP がプロパティ名、`fftTop` が値、`current_fileset` がオブジェクトです。さらに、`TOP_FILE` プロパティを最上位モジュールのデータ ソースに指定する必要があります。

この例は、ダッシュ (-) を含むプロパティ値の設定方法を示します。ダッシュのために、ダッシュ以降の値が値の一部ではなく新しいコマンド オプションと解釈され、エラーが発生することがあります。この場合、set_property コマンドの位置引数の代わりに明示的なオプションを使用する必要があります。

```
set_property {XELAB.MORE_OPTIONS} {-pulse_e_style ondetect} \
[get_filesets sim_1]
ERROR: [Common 17-170] Unknown option '-pulse_e_style ondetect',
please type 'set_property -help' for usage info.
set_property -name {XELAB.MORE_OPTIONS} -value {-pulse_e_style ondetect} \
-objects [get_filesets sim_1]
```

次の例では、指定の I/O バンクに内部 VREF プロパティ値を設定しています。

```
set_property internal_vref {0.75} [get_iobanks 0]
```

次の例は、指定の I/O バンクの DCI_CASCADE プロパティを設定することにより、DCI カスケードを定義します。

```
set_property DCI_CASCADE {14} [get_iobanks 0 ]
```

次の例は、synth_1 run を設定し、Vivado Synthesis 2013 オプションを設定して、その合成 run を実行します。

```
set_property flow {Vivado Synthesis 2016} \
[get_runs synth_1]
set_property STEPS.SYNTH_DESIGN.ARGS.FANOUT_LIMIT 500 \
[get_runs synth_1]
set_property STEPS.SYNTH_DESIGN.ARGS.GATED_CLOCK_CONVERSION on \
[get_runs synth_1]
set_property STEPS.SYNTH_DESIGN.ARGS.FSM_EXTRACTION one_hot \
[get_runs synth_1]
launch_runs synth_1
```

次の例は上記の例と同様ですが、-dict オプションを使用して 1 つの set_property コマンドで合成 run のすべてのプロパティを設定しています。

```
set_property -dict [ list flow {Vivado Synthesis 2016} \
STEPS.SYNTH_DESIGN.ARGS.FANOUT_LIMIT 500 \
STEPS.SYNTH_DESIGN.ARGS.GATED_CLOCK_CONVERSION on \
STEPS.SYNTH_DESIGN.ARGS.FSM_EXTRACTION \
one_hot ] [get_runs synth_1]
launch_runs synth_1
```

関連項目

- [current_fileset](#)
- [create_property](#)
- [create_run](#)
- [get_cells](#)
- [get_property](#)
- [get_runs](#)
- [launch_runs](#)
- [list_property](#)
- [list_property_value](#)
- [report_property](#)
- [reset_property](#)

set_speed_grade

タイミング スピード グレードと温度グレードを設定します。

構文

```
set_speed_grade [-temperature <arg>] [-quiet] [-verbose] [<value>]
```

戻り値

結果文字列

使用法

名前	説明
[-temperature]	タイミング解析に使用する温度グレードを指定します。7 シリーズ以前のデバイスでは使用できません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<value>]	タイミング解析に使用するスピード グレードを指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

注記: デザインに対して `set_speed_grade` コマンドを使用した後、タイミング解析には使用できますが、インプリメンテーションは実行されません。デザインのインプリメンテーションを実行する場合は、デザイン チェックポイントを保存し、`read_checkpoint -part` コマンドを使用してデザインを新しいスピード グレードでインプリメントしてください。

現在のデザインのターゲット デバイスに対してタイミング解析で使用するスピード グレードを設定します。

このコマンドは、ターゲット デバイスのスピード グレードをタイミング解析用にのみ変更するためのもので、デザインのほかのプロセスには影響しません。開いている合成済みデザインまたはインプリメント済みデザインで実行する必要があります。

`report_timing_summary`、`report_timing`、またはその他のタイミング コマンドを実行する前に、`set_speed_grade` コマンドを使用して解析用のスピード グレードを変更します。タイミングが有効であれば、`set_property` または `set_part` コマンドを使用してデザインのターゲット パーツを変更し、デザインを再合成および再インプリメントできます。



ヒント: UltraScale デバイスでは、`-temperature` を指定するか、`<value>` でパーツのスピード グレードを指定します。7 シリーズ デバイスでは、`<value>` のみを指定可能です。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスと設定されたスピード グレードが返され、正常に実行されなかった場合はエラーが返されます。

引数

-temperature (オプション): タイミング解析に使用する UltraScale デバイスの温度グレードを指定します。

注記: このオプションは、7 シリーズ以前のデバイスでは使用できません。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<value> (オプション): ターゲット デバイスのスピード グレードを指定します。典型的な値は、-1、-1L、-2、-3 です。-999 などの無効な値を指定すると、現在のパーツに有効な値をリストするエラー メッセージが表示されます。

例

次の例では、現在のデザインのデバイスのスピード グレードを -1 に設定しています。

```
set_speed_grade -1
```

関連項目

- [report_drc](#)
- [report_timing](#)
- [report_timing_summary](#)
- [set_part](#)
- [set_property](#)

set_switching_activity

指定したオブジェクトまたはデフォルトのタイプのスイッチング アクティビティを設定します。

構文

```
set_switching_activity [-toggle_rate <arg>] [-default_toggle_rate <arg>]
    [-type <args>] [-all] [-static_probability <arg>]
    [-default_static_probability <arg>] [-signal_rate <arg>] [-hier]
    [-deassert_resets] [-quiet] [-verbose] [<objects>...]
```

使用法

名前	説明
[-toggle_rate]	トグル レートは、同期ロジック エLEMENTの出力がその入力に対して切り替わるレート (%) を示し、0 ~ 200% の範囲で表されます。トグル レートが 100% の場合、各クロック サイクルで出力が平均で 1 回トグルし、立ち上がりエッジまたは立ち下がりエッジのいずれかで変化することを示し、有効出力信号周波数はクロック周波数の 1/2 になります。Default: 0.0
[-default_toggle_rate]	消費電力解析でデザインのプライマリ入力に使用するデフォルトのトグル レートを指定します。デフォルトのトグル レートは、スイッチ アクティビティがユーザーにより 設定されないプライマリ入力ネット、シミュレーション データ、またはデザインの制約に設定されます。有効な値は 0 <= value < 200 で、デフォルト値は 12.5 です。Default: 12.5
[-type]	指定のカテゴリのノードを指定します。有効なタイプは、io_output、io_bidir_enable、register、lut_ram、lut、dsp、bram_enable、bram_wr_enable、gt_txdata、gt_rxdata です。
[-all]	-type オプションと共に使用し、-type オプションで指定したインスタンス内のネットのスイッチング アクティビティを設定します。
[-static_probability]	スタティック確率値を指定します。有効な値は 0 <= value <= 1 で、デフォルトは 0.5 です。
[-default_static_probability]	デザインの消費電力解析で使用するデフォルトのスタティック確率を指定します。デフォルトのスタティック確率は、ユーザー、デザインのシミュレーション データまたは制約でスイッチング アクティビティが指定されていないプライマリ入力に使用されます。有効な値は 0 <= value <= 1 で、デフォルト値は 0.5 です。Default: 0.5
[-signal_rate]	ELEMENTのステートが 1 秒間に变化する回数を指定します。サイリンクス ツールでは、秒ごとの遷移数が百万単位で示されます (Mtr/s)。Default: 0.0
[-hier]	<objects> で指定した階層インスタンスのすべてのレベルでスイッチング アクティビティを設定します。このオプションは、<objects> 引数と共に使用する必要があります。
[-deassert_resets]	極性に競合のないすべてのセット、リセット、プリセット、およびクリア信号をすべてディassertします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	スイッチング アクティビティを設定するオブジェクトを指定します。

カテゴリ

XDC、Power (電力)

説明

現在の合成済みデザインまたはインプリメント済みデザインで消費電力見積もりを実行する際に使用する信号レートおよびスイッチング確率を設定します。レポートされるスイッチング アクティビティには、ネット、ポート、ピンの単純な信号レートと単純なスタティック確率、およびセルのステートに依存するスタティック確率が含まれます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

デザインのスイッチング アクティビティは、静的およびダイナミック消費電力の両方に影響します。スタティック消費電力はロジック ステートの遷移によって異なり、ダイナミック消費電力はトグル レートに正比例します。

`set_switching_activity` コマンドは、デザイン全体のデフォルト アクティビティ レートを指定するか、デザインに含まれる 1 つ以上の信号または指定のモジュールのアクティビティを定義するのに使用できます。

現在のスイッチング アクティビティ 属性は、`report_switching_activity` コマンドを使用して確認できます。スイッチング アクティビティをデフォルト 値にリセットするには、`reset_switching_activity` コマンドを使用します。

注記: `reset_switching_activity` コマンドは、指定したオブジェクトのスイッチング アクティビティをリセットするために使用します。これらの値を変更またはリセットするには、`set_switching_activity -default_toggle_rate` または `-default_static_probability` コマンドを使用します。

引数

`-toggle_rate <value>` (オプション): 同期ロジック エLEMENTの出力の指定したクロック入力に対するスイッチング レートとしてトグル レート (%) を指定します。トグル レートは、0 ~ 100% の範囲で指定します。クロックおよび DDR 信号では、200% までのトグル レートを指定できます。トグル レートが 100% の場合、各クロック サイクルで出力が平均で 1 回トグルし、立ち上がりエッジまたは立ち下がりエッジのいずれかで変化することを示し、有効出力信号周波数はクロック周波数の 1/2 になります。<value> のデフォルト値は、`-default_toggle_rate` オプションで指定されているクロック周波数の 12.5% です。

注記: このオプションは、`-static_probability` オプションと共に指定する必要があります。

`-default_toggle_rate <rate>` (オプション): 消費電力解析で現在のデザインのプライマリ入力に使用されるデフォルトのトグル レートを設定します。デフォルトのトグル レートは、ユーザー、デザインのシミュレーション データまたは制約でスイッチング アクティビティが指定されていないプライマリ入力ネットに使用されます。非同期入力では、トグル レートはデザインの最速のクロックに対して設定されます。有効な値は $0 \leq \text{value} < 200$ で、デフォルト値は 12.5 です。

注記: このオプションはデザイン全体のデフォルト トグル レートを指定するので、<objects> と共に使用することはできません。

`-type <arg>` (オプション): `set_switching_activity` コマンドを適用するロジック エンティティのタイプを指定します。デフォルトでは、現在のデザインの最上位または指定の <objects> に適用されます。`-type` オプションは、現在のデザインの最上位にあるロジック オブジェクトの指定のタイプにコマンドを適用します。`-all` または `-hier` オプションを使用して、コマンドを適用するオブジェクトの範囲を変更できます。有効なロジック タイプは次のとおりです。

- `io_output`: プライマリ出力。
- `io_bidir_enable`: 双方向ポートのイネーブル ピン。

- register: 指定のデザイン/階層のすべてのレジスタ 出力。
- lut: 指定のデザイン/階層のすべての LUT 出力。
- lut_ram: 指定のデザイン/階層のすべての分散 RAM 出力。
- dsp: 指定のデザイン/階層のすべての DSP 出力。
- bram_enable: BRAM のイネーブル ピン (ENARDEN/ENBWREN)。
- bram_wr_enable: BRAM のライト イネーブル (WEA/WEBWE)。
- gt_txdata: すべての GT の出力 TX データ ピン。
- gt_rxdata: すべての GT の出力 RX データ ピン。

注記: `-type` オプションは、`-toggle_rate` または `-signal_rate` オプションを設定する場合にのみ使用可能で、ほかの設定はサポートされません。

`-all` (オプション): `-type` と共に使用し、指定したタイプのロジック オブジェクト インスタンスすべてのスイッチング アクティビティを設定します。デフォルトでは、`set_switching_activity` コマンドは現在のデザインの最上位または現在のインスタンス (`current_instance`)、あるいはそのレベルにある指定のタイプのオブジェクトに適用されます。

`-static_probability <value>` (オプション): 消費電力解析で使用するスタティック スwitching 確率を指定の `<objects>` に設定します。有効な値は $0 \leq \text{<value>} \leq 1$ で、デフォルト値は 0.5 です。

`-default_static_probability <value>` (オプション): 現在のデザインの消費電力解析で使用されるデフォルトのスタティック 確率を設定します。デフォルトのスタティック 確率は、ユーザー、デザインのシミュレーション データまたは制約でスイッチング アクティビティが指定されていないプライマリ入力に使用されます。有効な値は $0 \leq \text{value} \leq 1$ で、デフォルト値は 0.5 です。

注記: このオプションはデザイン全体のデフォルトを定義するので、`<objects>` と共に使用することはできません。特定のオブジェクトの値 (`<value>`) を定義するには、`-static_probability` オプションを使用してください。

`-signal_rate <value>` (オプション): 解析で使用する信号周波数を、秒ごとの遷移数 (百万単位) で指定します。エレメントがステートを変更する 1 秒ごとの回数を指定します。Low から High への遷移と High から Low への遷移の両方を含みます。デフォルト値は 0 です。

注記: このオプションは、`-static_probability` オプションと共に使用する必要があります。

`-hier` (オプション): スwitching アクティビティを指定の階層オブジェクト (`<objects>`) のすべてのレベルの信号に適用します。`-hier` を使用しない場合、指定の `<objects>` または現在の階層レベルにある `-type` オプションで指定されたオブジェクトにスイッチング アクティビティが適用されます。

`-deassert_resets` (オプション): 信号のスタティック 確率を 0 に設定することにより、すべてのリセット タイプの制御信号 (セット、プリセット、リセット、クリア) をすべて自動的にディアサートするよう指定します。極性が競合していない制御信号のみがディアサートされます。ネットがアクティブ Low およびアクティブ High のリセット ピンに接続されている場合、この信号はディアサートされません。また、ネットがアクティブ High のリセット、およびアクティブ High のセットまたはイネーブル ピンに接続されている場合も、この信号はディアサートされません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (オプション): スイッチング アクティビティを適用するポート、ピン、およびネット オブジェクトを1つまたは複数指定します。-type オプションを使用してロジック オブジェクトを定義した場合は、セルを指定します。

例

次の例では、すべてのポートに信号レートとスイッチング確率を指定し、それらのポートのスイッチング属性をレポートしています。

```
set_switching_activity -signal_rate 55 -static_probability .33 [get_ports]
report_switching_activity [get_ports]
```

次の例では、現在のデザインのデフォルト スイッチング確率を指定しています。

```
set_switching_activity -default_static_probability .75
```

次の例では、指定のトグル レートおよびスタティック確率を階層 CPU/MEM のすべてのレジスタに設定します。

```
set_switching_activity -type register -toggle_rate 0.4 \
    -static_probability 0.5 [get_cells CPU/MEM]
```

次の例では、指定のトグル レートおよびスタティック確率を階層 CPU/ およびその下位階層のすべてのレジスタに設定します。

```
set_switching_activity -type register -toggle_rate 0.4
    -static_probability 0.5 -hier [get_cells CPU]
```

関連項目

- [get_clocks](#)
- [get_nets](#)
- [get_ports](#)
- [power_opt_design](#)
- [report_power](#)
- [report_switching_activity](#)
- [reset_switching_activity](#)

set_system_jitter

システム ジッターを設定します。

構文

```
set_system_jitter [-quiet] [-verbose] <system_jitter>
```

戻り値

システム ジッター

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><system_jitter></code>	システム ジッター値を指定します。有効な値は 0 以上です。

カテゴリ

[XDC](#)

説明

デザインのプライマリ クロックおよび生成クロックを含むすべてのクロックに対し、システム ジッターをナノ秒 (ns) で設定します。システム ジッターは、電源ノイズやボード ノイズなど、FPGA 内のすべてのクロックに影響する過剰なノイズを考慮するために使用します。デフォルトのシステム ジッターはテクノロジーによって異なり、サポートされるすべての動作条件下で複数の電源を使用したデバイス特定評価に基づいて、各ザイリンクス FPGA ファミリーに対してあらかじめ定義されています。

システム ジッターは、パスのクロックのばらつきの算出に使用されるトータル システム ジッター (T_{sj}) の要素です。内部ノードの同時スイッチ、クロストーク、およびその他デザインのパスのタイミングに影響する現象が原因で V_{ccint} レール上に発生する最大ノイズによります。



重要: ザイリンクスが算出したジッターには、クロック リソースによるばらつき、入力ジッター、およびシステムジッターが考慮されています。set_system_jitter コマンドを使用すると、ザイリンクスが算出したデフォルトのシステム ジッターは無効になるのでお勧めしません。

システム ジッターおよび入力ジッターは、通常ガウス分布に従うランダム ジッターであり、ワースト ケースの組み合わせを表すために二次方程式で追加されます。入力ジッターがヌルの場合、内部レジスタ間のパスのトータル システム ジッター (T_{sj}) は次の式で算出されます。

- $$T_{sj} = \sqrt{(\text{SourceClockSystemJitter}^2 + \text{DestinationClockSystemJitter}^2)}$$

たとえばデフォルトのシステム ジッター 50 ps を使用した場合、次のようになります。

- $$T_{sj} = \sqrt{(0.050^2 + 0.050^2)} = 0.071\text{ns} = 71\text{ps}$$

`set_system_jitter` コマンドは、デザインのすべてのクロックに適用されます。特定のプライマリ クロックに追加のジッターを指定するには、`set_input_jitter` コマンドを使用します。



ヒント: `SYSTEM_JITTER` はクロックのプロパティとしてレポートされますが、デザインに含まれるすべてのクロックに適用されます。`INPUT_JITTER` もプライマリ クロックのプロパティです。これらのプロパティは、`get_property` または `report_property` コマンドを使用して返すことができます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<system_jitter> (必須): システム全体に適用するシステム ジッターをナノ秒 (ns) で指定します。

`set_system_jitter` コマンドでジッターを指定すると、デフォルト値の代わりに指定した値が使用されます。

例

次の例では、プライマリ クロック `sysClk` を定義し、システム全体のジッターを 0.1 ns に設定しています。

```
create_clock -period 10 -name sysClk [get_ports sysClk]
set_system_jitter 0.1
```

次の例では、プライマリ クロック `sysClk` を定義し、このプライマリ クロックを 2 で分周したクロック `sysClkDiv2` を生成した後、デザインのすべてのクロックに対してシステム ジッター 0.2 ns を設定しています。プライマリ クロックにのみ、追加の入力ジッター 0.09 ns を指定しています。

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_generated_clock -name sysClkDiv2 -source [get_ports sysClk] \
    -divide_by 2 [get_pins clkgen/sysClkDiv/Q]
set_system_jitter 0.2
set_input_jitter sysClk 0.09
```

次の例では、`sysClk` および `procClk` というプライマリ クロックを定義し、システムのすべてのクロックに対してシステム ジッター 0.2 ns を設定しています。クロック `procClk` にのみ、追加の入力ジッター 0.05 ns を指定しています。

```
create_clock -period 10 -name sysClk [get_ports sysClk]
create_clock -period 25 -name procClk [get_ports procClk]
set_system_jitter 0.2
set_input_jitter procClk 0.05
```

関連項目

- [report_timing](#)
- [set_clock_uncertainty](#)
- [set_input_delay](#)

- [set_input_jitter](#)

set_units

チェックに使用する単位を設定します。

構文

```
set_units [-capacitance <arg>] [-current <arg>] [-voltage <arg>]
          [-power <arg>] [-resistance <arg>] [-altitude <arg>] [-quiet]
          [-verbose]
```

使用法

名前	説明
[-capacitance]	キャパシタンスの単位 (ファラッド) を指定します。有効な値は kF ~ fF で、デフォルトは pF です。
[-current]	電流の単位 (アンペア) を指定します。有効な値は kA ~ fA で、デフォルトは mA です。
[-voltage]	電圧の単位 (ボルト) を指定します。有効な値は kV ~ fV で、デフォルトは V です。
[-power]	電力の単位 (ワット) を指定します。有効な値は kW ~ fW で、デフォルトは mW です。
[-resistance]	抵抗の単位 (オーム) を指定します。有効な値は kOhm ~ fOhm で、デフォルトは ohm です。
[-altitude]	標高の単位を指定します。有効な値は meters または feet で、デフォルトは meters です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

SDC、XDC

説明

デザインの解析で使用するデフォルトの単位を指定します。report_power コマンドで返される値は、-current、-voltage、-power、および -resistance オプションの設定により変更されます。

set_units コマンドを複数回使用して、単位を定義および再定義できます。set_units に以前に設定した単位値が含まれている場合は、その単位が再定義されます。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

-capacitance <arg> (オプション): キャパシタンスの単位 (ファラッド) を指定します。有効な値の範囲はキロファラド (kF) からフェムトファラド (fF) までで、デフォルトの容量の単位はピコファラド (pF) です。

-current <arg> (オプション): 電流の単位 (アンペア) を指定します。有効な値の範囲はキロアンペア (kA) からフェムトアンペア (fA) までで、デフォルトの電流の単位はミリアンペア (mA) です。

`-voltage <arg>` (オプション): 電圧の単位 (ボルト) を指定します。有効な値の範囲はキロボルト (kV) からフェムトボルト (fV) までで、デフォルトの電圧の単位はボルト (V) です。

`-power <arg>` (オプション): 電力の単位 (ワット) を指定します。有効な値の範囲はキロワット (kW) からフェムトワット (fW) までで、デフォルトの電力の単位はミリワット (mW) です。

`-resistance <arg>` (オプション): 抵抗の単位 (オーム) を指定します。有効な値の範囲はキロオーム (kOhm) からフェムトオーム (fOhm) までで、デフォルトの抵抗の単位はオーム (Ohm) です。

`-altitude [meters | feet]` (オプション): 海拔の単位をメートル (meters) またはフット (feet) に指定します。デフォルトの単位は meters です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、電圧の単位をミリボルトに設定し、すべての値に 3 桁を使用するよう設定しています。

```
set_units -voltage mV
```

次の例では、電流のデフォルト単位をアンペアに変更しています。

```
set_units -voltage kV -current A
```

注記: 2 番目の例の `set_units` は、最初の例の電圧単位を再定義し、電流の単位を定義しています。

関連項目

- [report_power](#)
- [set_operating_conditions](#)

set_value

HDL オブジェクト (変数、信号、ワイヤ、またはレジスタ) の現在の値を指定の値に設定します。

構文

```
set_value [-radix <arg>] [-quiet] [-verbose] <hdl_object> <value>
```

使用法

名前	説明
[-radix]	値の解釈に使用する基数を指定します。有効な値は、default、dec、bin、oct、hex、unsigned、ascii、smag です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<hdl_object>	値を設定する HDL オブジェクトを指定します。
<value>	指定のオブジェクトに割り当てる値を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーション時間における 1 つの HDL オブジェクトの値を設定します。

HDL オブジェクトには、Verilog または VHDL テストベンチおよびソース ファイルで定義されている HDL 信号、変数、または定数が含まれます。HDL 信号には、Verilog の wire または reg エンティティ、および VHDL 信号が含まれます。HDL 変数には、Verilog の real、realtime、time、event などがあります。

HDL 定数には、Verilog のパラメーターおよび localparam、VHDL ジェネリックおよび定数が含まれます。HDL スコープは、Verilog のモジュール、関数、タスク、プロセス、begin-end ブロックなど、HDL コードの宣言部分で定義されます。VHDL スコープには、エンティティ/アーキテクチャ定義、関数、プロシージャ、およびプロセス ブロックが含まれます。

引数

-radix <arg> (オプション): 指定したオブジェクトの値を表示するのに使用する基数を指定します。有効な値は default、dec、bin、oct、hex、unsigned、ascii、および smag です。

注記: dec は、符号付き 10 進数を示します。符号なしデータの場合は、unsigned を指定してください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hdl_object> (必須): 値を取得する 1 つの HDL オブジェクトの名前を指定します。オブジェクトは、名前または `get_objects` コマンドで指定します。

<value> (必須): 指定したオブジェクトに設定する値を指定します。指定の値 (<value>) は、<hdl_object> のタイプによって異なります。HDL オブジェクトタイプには、logic、浮動小数点、VHDL 列挙型、VHDL 整数があります。logic 以外では、`-radix` オプションは無視されます。

- logic は、実際の HDL オブジェクトタイプを示すのではなく、次のような VHDL std_logic と同様の値を持つオブジェクトを示します。
 - Verilog 暗示 4 値ビット型。
 - VHDL ビットおよび std_logic 既定義型。
 - std_logic のサブセットである VHDL 列挙型 (文字リテラル 0 および 1 を含む)。
- logic 型では、値は基数によって異なります。
 - 指定の値のビット数が logic 型に適切なビット数より少ない場合、値は適切な長さになるよう 0 で拡張されます (符号拡張されません)。
 - 指定の値のビット数が logic 型に適切なビット数より多い場合、MSB 側の余分なビットはすべて 0 である必要があります、そうでない場合は Vivado シミュレータでサイズ不一致エラーが返されます。
- 浮動小数点オブジェクトの値は浮動小数点値です。
- logic でない VHDL 列挙型の値は、値の列挙セットからのスカラー値 (文字の場合はシングル クォーテーションなし) です。
- VHDL 整数型の値は、その型で許容される範囲の符号付き 10 進整数です。

例

次の例では、sysClk 信号の値を設定しています。

```
set_value sysClk Z
```

次の例では、bin、dec、および unsigned 基数を使用して、指定のバスに同じ値を設定しています。

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 10100101
set_value -radix unsigned /test/bench_VStatus_pad_0_i[7:0] 165
set_value -radix dec /test/bench_VStatus_pad_0_i[7:0] -91
```

次の例では、指定の値がロジックタイプに適切なビット数よりも少ない場合にビット拡張が実行されることを示しています。

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 101
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
00000101
```

次の例では、指定の値がロジック タイプに適切なビット数よりも多い場合にビット切り捨てが実行されることを示しています。

```
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 0010100101
get_value -radix bin /test/bench_VStatus_pad_0_i[7:0]
10100101
set_value -radix bin /test/bench_VStatus_pad_0_i[7:0] 1110100101
ERROR: [#UNDEF] Object size 8 does not match size of given value 1110100101
```

注記: 2 番目の `set_value` コマンドでは、余分なビットは 0 ではないので、エラーが返されます。

関連項目

- [current_time](#)
- [get_objects](#)
- [get_value](#)
- [report_values](#)

setup_ip_static_library

現在のプロジェクトまたはリポジトリから IP スタティック ファイルを抽出し、compile_simlib 用に準備します。

構文

```
setup_ip_static_library [-directory <arg>] [-ip_repo_path <arg>]  
  [-ips <arg>] [-library <arg>] [-project] [-install]  
  [-no_update_catalog] [-force] [-quiet] [-verbose]
```

戻り値

なし

使用法

名前	説明
[-directory]	指定したディレクトリのスタティック ファイルを抽出します。
[-ip_repo_path]	指定した IP リポジトリ パスからスタティック ファイルを抽出します。
[-ips]	指定した IP のみのスタティック ファイルを抽出します。
[-library]	指定した IP ライブラリのスタティック ファイルを抽出します。
[-project]	現在のプロジェクトのスタティック ファイルを抽出します。
[-install]	IP カタログのスタティック ファイルを抽出します。
[-no_update_catalog]	IP カタログをアップデートしません。
[-force]	既存のスタティック ファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)、[xilinx_tclstore \(ザイリンクス Tcl Store\)](#)

説明

現在のプロジェクトで使用されている IP コアまたはザイリンクス IP カタログからの IP コアのスタティック シミュレーション ファイルを取得し、compile_simlib で指定のシミュレータ用に IP をコンパイルする際に使用するソース ライブラリを作成します。

引数

-directory <arg> (オプション): スタティック ライブラリへのディレクトリ パスを指定します。デフォルト (このオプションなし) では、ライブラリは static_compiled_lib というディレクトリに保存されます。

-ip_repo_path args <arg> (オプション): 指定の IP リポジトリ パスからスタティック ファイルを抽出します。

-ips <args> (オプション): 指定した IP オブジェクトのみのスタティック ファイルを抽出します。IP は get_ips コマンドを使用して指定できます。

`-library <args>` (オプション): 指定した IP ライブラリのスタティック ファイルを抽出します。IP ライブラリは名前で指定できます。

`-project` (オプション): 現在のプロジェクトのスタティック ソース ライブラリを抽出して準備します。

`-install <arg>` (オプション): IP カタログのスタティック ソース ライブラリを抽出して準備します。

`-no_update_catalog` (オプション): IP カタログをアップデートしません。

`-force` (オプション): 既存のファイルを上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のプロジェクトに含まれるすべての IP のスタティック ライブラリを `./static_compiled_lib` に作成しています。

```
setup_ip_static_library -project
```

次の例では、現在のプロジェクトのスタティック ライブラリを `/work/simlib` に作成しています。指定したディレクトリが存在しない場合は、作成されます。

```
setup_ip_static_library -directory /work/simlib -project
```

関連項目

- [compile_simlib](#)
- [get_files](#)
- [get_ips](#)

setup_pr_configurations

パーティション インスタンスおよび RM の組み合わせに基づいて、最小限の PR コンフィギュレーションおよび子インプリメンテーション run を自動的に作成します。

構文

```
setup_pr_configurations [-partitions <args>] [-use_netlist] [-force]
                        [-run <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
[-partitions]	パーティション インスタンスとリコンフィギャラブル モジュールのペアを指定します。
[-use_netlist]	ネットリストを使用してパーティション定義のインスタンスを取得し、PR コンフィギュレーションを作成します。
[-force]	アクティブな親インプリメンテーション run の PR コンフィギュレーションおよび子 run と PR コンフィギュレーションを削除し、新しい PR コンフィギュレーションおよび run を作成します。
[-run]	子インプリメンテーション run および PR コンフィギュレーションを作成する親インプリメンテーション run を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Partition \(パーティション\)](#)

説明

パーティション インスタンスおよびリコンフィギャラブル モジュールの組み合わせに基づいて、最小限の PR コンフィギュレーションおよび子インプリメンテーション run を自動的に作成します。

パーシャル リコンフィギュレーション (PR) デザイン フローでは、PR コンフィギュレーションでパーティション定義 (partitionDef) の特定のインスタンスに割り当てるリコンフィギャラブル モジュール (RM) を指定できます。このフローでは、スタティック デザインと 1 つ以上の RM の組み合わせに基づいて、独自のデザイン コンフィギュレーションを作成できます。PR デザイン フローでは、PR コンフィギュレーションをそれぞれインプリメンテーションする必要があります。この結果、RM のパーシャル ビットストリームは作成されますが、統合された各コンフィギュレーションのビットストリーム全体は作成されません。詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

このコマンドは、それらのコンフィギュレーションに必要な PR コンフィギュレーションおよびインプリメンテーション run を自動的に作成します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-partitions <arg>` (オプション): PR コンフィギュレーションに割り当てるパーティション インスタンスおよびリコンフィギュラブル モジュールのペアのリストを指定します。引数は `<partitionInstance>:<RM>` という形式で指定する必要があります。この形式により、デザインに含まれる 1 つの `partitionDef` の複数のインスタンスに異なる RM を割り当てることができます。

`-force` (オプション): 現在の PR コンフィギュレーションおよびコンフィギュレーション `run` を削除し、新しい PR コンフィギュレーションおよび `run` を作成します。

`-run <arg>` (オプション): PR コンフィギュレーションのインプリメンテーション `run` に使用する親 `run` を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、プロジェクトのパーティション定義 (`partitionDef`) および RM をサポートするのに必要な PR コンフィギュレーションおよび PR コンフィギュレーション インプリメンテーション `run` を自動的に作成しています。

```
setup_pr_configurations
```

関連項目

- [create_partition_def](#)
- [create_pr_configuration](#)
- [create_reconfig_module](#)
- [current_pr_configuration](#)
- [delete_pr_configurations](#)
- [get_pr_configurations](#)

show_objects

[Find Results] ウィンドウにオブジェクトを表示します。

構文

```
show_objects [-name <arg>] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
[-name]	タブのタイトルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	[Find Results] ウィンドウに表示するオブジェクトを指定します。

カテゴリ

GUIControl (GUI 制御)

説明

指定のオブジェクトを Vivado IDE の [Find Results] ウィンドウに表示します。

注記: このオプションは、Vivado IDE で実行する場合にのみ有益です。Tcl またはバッチ モードで実行しても、エラーやコメントは返されません。

引数

-name <arg>: [Find Results] ウィンドウで開くレポート タブの名前を指定します。名前を指定しない場合、デフォルト名 `find_1` が使用されます。



ヒント: `show_objects` コマンドを `-name` オプションを使用せずに複数回実行すると、前の結果が上書きされます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects>: [Find Results] ウィンドウに表示するデザイン オブジェクトを指定します。オブジェクトは、名前で指定するのではなく、デザイン オブジェクトとして指定する必要があります。オブジェクトは、`get_cells`、`get_pins`、`get_nets` などのコマンド、または `all_inputs` および `all_rams` などのコマンドで指定できます。

例

次の例では、現在のデザインの DSP オブジェクトすべてを [Find Results] ウィンドウに表示しています。

```
show_objects -name All_DSPs [all_dsps]
```

次の例では、デザイン階層に含まれるすべてのセルで、PRIMITIVE_TYPE がクロックまたは DSP のものを表示しています。

```
show_objects -name find_1 [get_cells -hierarchical \  
-filter { PRIMITIVE_TYPE =~ CLK.*.* || PRIMITIVE_TYPE =~ MULT.dsp.* } ]
```

関連項目

- [all_inputs](#)
- [all_rams](#)
- [get_cells](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)

show_schematic

ネットリスト アイテムを回路図で表示します。

構文

```
show_schematic [-add] [-remove] [-regenerate] [-pin_pairs] [-name <arg>]
               [-quiet] [-verbose] <objects>
```

使用法

名前	説明
[-add]	既存の回路図に追加します。
[-remove]	既存の回路図から削除します。
[-regenerate]	回路図のレイアウトを再生成します。
[-pin_pairs]	オブジェクトを接続されたピンのペアとして処理します。これは、パスを表示する際に有益な場合があります。
[-name]	[Schematic] ウィンドウのタイトルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	回路図で表示するネットリスト アイテムを指定します。

カテゴリ

GUIControl (GUI 制御)

説明

ツールを GUI モードで起動している場合に、指定のデザイン オブジェクトを含む回路図を作成します。

表示される回路図の範囲は、指定したオブジェクトによって異なります。セルから作成された回路図には、指定したセルとセル間の接続が表示されます。ピンから作成された回路図には、ピン オブジェクトが表示され、-pin_pairs オプションを指定した場合はペア ピンの接続が表示されます。ネットから作成された回路図には、指定したネットと、ネットに接続されているセルおよびポートが表示されます。

複数の階層レベルの回路図を表示するには、current_instance コマンドを使用して最上位階層または必要な階層レベルを設定し、get_* コマンドでデザイン オブジェクトを指定する際に -hierarchical オプションを使用します。

注記: このオプションは、Vivado IDE で実行する場合にのみ有益です。Tcl またはバッチ モードで実行しても、エラーやコメントは返されません。

引数

-add (オプション): 指定のオブジェクトを回路図に追加します。

-remove (オプション): 指定のオブジェクトを回路図から削除します。

-regenerate (オプション): 回路図を再生成します。

`-pin_pairs` (オプション): 接続されたピン オブジェクトのペアを指定した場合に、回路図にピンとその間のワイヤを表示します。`-pin_pairs` オプションを指定しない場合、または接続されていないピンを指定した場合は、ワイヤは表示されません。

`-name <arg>` (オプション): GUI に表示される [Schematic] ウィンドウの名前を指定します。回路図にオブジェクトを追加する場合、回路図からオブジェクトを削除する場合、回路図を再生成する場合は、この名前を使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): 回路図に表示するネットリスト オブジェクトを指定します。

例

次の例では、デザインの最上位の回路図を作成し、ネットおよびネットに接続されているポートおよびセルを表示しています。

```
show_schematic [get_nets]
```

次の例では、階層レベルを指定し、そのレベルから下の階層レベルを含む回路図を作成しています。

```
current_instance A
show_schematic [get_nets -hier]
```

次の例では、指定のピンと、それらを接続するワイヤを表示する回路図を作成しています。

```
show_schematic -pin_pairs [get_pins {data0_i/O data_reg/D}]
```

関連項目

- [current_instance](#)
- [get_cells](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)

split_diff_pair_ports

差動ペアの 2 つのポートをシングルエンド ポートに分割します。

構文

```
split_diff_pair_ports [-quiet] [-verbose] <ports>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><ports></code>	分割するポートを指定します。

カテゴリ

[PinPlanning \(ピン プランニング\)](#)

説明

既存の差動ペア ポートを 2 つのシングルエンド ポートに分割します。

注記: このコマンドを実行しても、その動作に関するメッセージや戻り値は返されません。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<ports>` (必須): シングル ポートに分割する差動ペアの 2 つのポート名を指定します。

例

次の例では、指定の差動ペア ポートを 2 つのシングルエンド ポートに分離しています。

```
split_diff_pair_ports PORT_N PORT_P
```

関連項目

- [make_diff_pair_ports](#)

- [create_port](#)
- [create_interface](#)

start_gui

GUI を起動します。

構文

```
start_gui [-verbose]
```

使用法

名前	説明
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

ツールを Vivado Design Suite Tcl シェルで実行している場合に GUI を起動します。GUI は、現在のプロジェクト、デザイン、run の情報を読み込んで起動します。

引数

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ツールを Tcl モードで実行しているときに、GUI を起動しています。

```
Vivado% start_gui
```

関連項目

- [stop_gui](#)

start_vcd

VCD 出力の記述を開始します (\$dumpopen Verilog システム タスクと同等)。open_vcd で開始した VCD の生成を stop_vcd Tcl コマンドで停止した後に使用できます。

構文

```
start_vcd [-quiet] [-verbose]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

start_vcd コマンドは、VCD (Value Change Dump) 情報の指定した VCD オブジェクトへの記述を開始します。この Tcl コマンドは、Verilog の \$dumpopen システム タスクと同様の操作を実行します。



重要: start_vcd コマンドを実行する前に、open_vcd コマンドを実行する必要があります。

戻り値はありません。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、HDL 信号の現在の VCD ファイルへの記述を開始しています。

```
start_vcd
```

関連項目

- [close_vcd](#)
- [open_vcd](#)
- [stop_vcd](#)

startgroup

グループ単位で実行を取り消し/やり直しできるコマンド シーケンスを開始します。

構文

```
startgroup [-try] [-quiet] [-verbose]
```

戻り値

int

使用法

名前	説明
<code>[-try]</code>	既にグループが開始されている場合は、開始しません。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

グループ単位で実行を取り消し/やり直しできるコマンド シーケンスを開始します。コマンド シーケンスを終了するには、`endgroup` コマンドを使用します。

`undo` または `redo` で一度に取り消し/やり直しできるコマンド グループは複数作成できますが、コマンド グループを入れ子にすることはできません。`startgroup` を使用して新しいコマンド シーケンスを作成する前に、`endgroup` を使用してコマンド シーケンスを終了する必要があります。



ヒント: `startgroup/endgroup` コマンドは、`undo` コマンドで一度に取り消しできる、または `redo` コマンドを使用して一度にやり直しできる関連コマンドのシーケンスをサポートするためのものですが、一部のコマンドは `endgroup` を意図せずにトリガーすることがあり、コマンドによっては `undo` または `redo` コマンドはサポートされません。この制限は完全に定義されていません。

`startgroup` コマンドを実行したとき、グループが既に開始している場合は整数値 0 が返され、新しいグループが開始された場合は整数値 1 が返されます。

引数

`-try` (オプション): 新しいグループが開始された場合に 1 を返します。グループが既に開始している場合は 0 が返され、新しいグループは開始できません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、まず `startgroup` を実行し、関連するコマンドのシーケンスを実行して、`endgroup` を実行します。このコマンド シーケンスは、グループ単位で実行を取り消したりやり直したりすることができます。

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEngnSRM
add_cells_to_pblock pblock_wbArbEngine \
    [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEngnSRM \
    [get_cells [list usbEngine1/usbEngineSRAM]] -clear_locs
endgroup
```

関連項目

- [endgroup](#)
- [redo](#)
- [undo](#)

step

シミュレーションを次の文に進めます。

構文

```
step [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在のシミュレーションを HDL ソース ファイルの次の実行文に進めます。

この機能を使用すると、シミュレータでソース コードを 1 行ずつ実行できます。HDL ソースの各行または各機能がシミュレーション結果にどのように影響するかを調べる場合に有益です。

このコマンドを実行すると、HDL ソース ファイルからの次の実行行に関する情報が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、HDL ソース コードの現在の実行行を実行し、次の実行行でその行に関する情報を返して停止します。

```
step
Stopped at time : 0 fs : File "C:/Data/ug937/sim/testbench.v" Line 17
```

関連項目

- [run](#)
- [stop](#)

stop

条件内で使用し、条件が満たされた場合にシミュレーションを一時停止するよう指定します。

構文

```
stop [-quiet] [-verbose]
```

戻り値

シミュレーションでの停止は終了ではなく一時停止

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

`stop` コマンドは、現在のシミュレーションを一時停止します。このコマンドを `add_condition` コマンドで定義された条件内で使用し、条件が満たされた場合にシミュレーションを一時停止できます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`resetLow` という条件 (リセット信号が Low) を定義し、この条件が真になったときに、標準出力にメッセージを表示し、現在のシミュレーションを停止します。

```
add_condition -name resetLow {/testbench/reset == 0 } {  
  puts "Condition Reset was encountered at [current_time]. \  
    Stopping simulation."  
  stop }  
}
```

関連項目

- [add_condition](#)
- [report_conditions](#)
- [restart](#)
- [run](#)
- [step](#)

stop_gui

GUI を閉じます。

構文

```
stop_gui [-verbose]
```

使用法

名前	説明
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

GUI モードを停止し、ツールを Tcl モードにして、Vivado Design Suite Tcl シェルを実行します。Tcl モードでは、すべてのコマンドは Tcl コマンドまたは Tcl スクリプトで入力する必要があります。

引数

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ツールの GUI モードを終了し、Tcl モードにしています。

```
stop_gui
```

関連項目

- [start_gui](#)

stop_hw_hbm_amon

指定したハードウェア HBM に対してアクティビティ モニターの実行をディスエーブルにします。

構文

```
stop_hw_hbm_amon [-quiet] [-verbose] <hw_objects>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_objects></code>	ハードウェア オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

`stop_hw_hbm_amon` コマンドは、Vivado ハードウェア マネージャーで `run_hw_hbm_amon` コマンドを使用して開始した HBM アクティビティ モニターの実行を停止します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_objects>` (必須): 停止する HBM アクティビティ モニター オブジェクト (`hw_hbm`) を指定します。

例

次の例では、指定した HBM コアに関連付けられている HBM アクティビティ モニターを停止しています。

```
stop_hw_hbm_amon [get_hw_hbms *HBM_2]
```

関連項目

- [add_hw_hbm_pc](#)

- [commit_hw_hbm](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_hbms](#)
- [refresh_hw_hbm](#)
- [remove_hw_hbm_pc](#)
- [resume_hw_hbm_amon](#)
- [run_hw_hbm_amon](#)

stop_hw_sio_scan

ハードウェア SIO スキャンを停止します。

構文

```
stop_hw_sio_scan [-quiet] [-verbose] <hw_sio_scans>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_scans></code>	ハードウェア SIO スキャン

カテゴリ

Hardware (ハードウェア)

説明

Vivado シリアル I/O 解析機能で実行中の指定したスキャンを停止します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用のアイ スキャン ハードウェアを使用してリンクのスキャンを実行すると有益です。Vivado シリアル I/O 解析機能では、リンク スキャンを作成、実行、および保存できます。

このコマンドを使用すると、`run_hw_sio_scan` コマンドを使用して開始した進行中のスキャンを停止できます。

作成したスキャン オブジェクトを削除するには、`remove_hw_sio_scan` コマンドを使用します。

このコマンドが正常に実行された場合はメッセージが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_scans>` (必須): 停止するハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトを 1 つ以上指定します。`hw_sio_scan` は、`get_hw_sio_scans` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、run_hw_sio_scan コマンドを使用してスキャンを開始し、停止しています。

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_3}] 0]  
stop_hw_sio_scan [get_hw_sio_scans SCAN_3]
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [run_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)
- [write_hw_sio_scan](#)

stop_hw_sio_sweep

ハードウェア SIO スイープを停止します。

構文

```
stop_hw_sio_sweep [-quiet] [-verbose] <hw_sio_sweeps>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_sweeps></code>	ハードウェア SIO スイープ

カテゴリ

Hardware (ハードウェア)

説明

指定したスイープ スキャンを停止します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用の機能を使用してリンクのスキャンを実行すると有益です。また、リンクに対して GT の異なる設定を使用して複数のスキャンを実行するのも有益です。デザインにどの設定が最適かを判断するのに役立ちます。Vivado シリアル I/O 解析機能では、リンク スイープ (ある値の範囲で実行するリンク スキャンのグループ) を定義、実行、および保存できます。

このコマンドを使用すると、`run_hw_sio_sweep` コマンドを使用して開始した進行中のスイープ スキャンを停止できます。

作成したスイープ スキャン オブジェクトを削除するには、`remove_hw_sio_sweep` コマンドを使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_sweeps>` (必須): 停止する `hw_sio_sweep` オブジェクトを 1 つ以上指定します。`hw_sio_sweep` は、`get_hw_sio_sweeps` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定した実行中のスイープ スキャンを停止しています。

```
stop_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

関連項目

- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

stop_vcd

VCD 出力の記述を停止します (\$dumpoff Verilog システム タスクと同等)。VCD 出力の記述を再開するには、start_vcd Tcl コマンドを使用します。

構文

```
stop_vcd [-quiet] [-verbose]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

現在の VCD (Value Change Dump) ファイルへのシミュレーション値の記述を停止します。シミュレーション情報のファイルへの出力が、start_vcd コマンドで再開されるまで停止します。

この Tcl コマンドは、Verilog の \$dumpoff システム タスクと同様の操作を実行します。



重要: stop_vcd コマンドを実行する前に、open_vcd コマンドを実行する必要があります。

戻り値はありません。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例では、シミュレーション値の現在の VCD ファイルへの記述を停止しています。

```
stop_vcd
```

関連項目

- [close_vcd](#)
- [open_vcd](#)
- [start_vcd](#)

swap_locs

2 つの位置を入れ替えます。

構文

```
swap_locs [-quiet] [-verbose] <aloc> <bloc>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><aloc></code>	1 つ目の位置 (ポート、セル、サイト) を指定します。bloc と同じタイプである必要があります。
<code><bloc></code>	2 つ目の位置 (ポート、セル、サイト) を指定します。aloc と同じタイプである必要があります。

カテゴリ

[Floorplan \(フロアプラン\)](#)

説明

2 つの類似したロジック エLEMENT に設定されている LOC 制約を入れ替えます。ロジック エLEMENT とは、FPGA のデバイス リソースに配置可能なELEMENT です。

swap_locs コマンドを実行すると、選択された 2 つのELEMENT を新しいロケーションに配置できるかどうかを確認するため、一部の DRC チェックが実行されます。いずれかのELEMENT のロケーションが何らかの理由で無効である場合、swap_locs コマンドでエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

`<aloc>` (必須): 入れ替える 1 つ目のロジックのロケーションを指定します。ポート、セル、またはデバイス サイトとして指定できます。

`<bloc>` (必須): 入れ替える 2 つ目のロジックのロケーションを指定します。ポート、セル、またはデバイス サイトとして指定できます。<aloc> で指定したタイプと一致している必要があります。

例

次の例では、指定した 2 つのデバイス サイトに割り当てられているインスタンスを入れ替えています。

```
swap_locs [get_sites {OLOGIC_X2Y1}] [get_sites {OLOGIC_X2Y0}]
```

関連項目

- [get_cells](#)
- [get_ports](#)
- [get_sites](#)

synth_design

Vivado 合成を使用してデザインを合成し、デザインを開きます。

構文

```
synth_design [-name <arg>] [-part <arg>] [-constrset <arg>] [-top <arg>]
  [-include_dirs <args>] [-generic <args>] [-verilog_define <args>]
  [-flatten_hierarchy <arg>] [-gated_clock_conversion <arg>]
  [-directive <arg>] [-rtl] [-bufg <arg>] [-no_lc] [-fanout_limit <arg>]
  [-shreg_min_size <arg>] [-mode <arg>] [-fsm_extraction <arg>]
  [-rtl_skip_ip] [-rtl_skip_constraints] [-keep_equivalent_registers]
  [-resource_sharing <arg>] [-cascade_dsp <arg>]
  [-control_set_opt_threshold <arg>] [-incremental <arg>]
  [-max_bram <arg>] [-max_uram <arg>] [-max_dsp <arg>]
  [-max_bram_cascade_height <arg>] [-max_uram_cascade_height <arg>]
  [-retiming] [-no_srlextract] [-assert] [-no_timing_driven] [-sfcu]
  [-quiet] [-verbose]
```

戻り値

design object

使用法

名前	説明
[-name]	デザイン名
[-part]	ターゲット パーツを指定します。
[-constrset]	使用する制約ファイルセットを指定します。
[-top]	最上位モジュール名を指定します。
[-include_dirs]	Verilog 検索ディレクトリを指定します。
[-generic]	ジェネリック パラメーターを指定します。構文は、-generic <name>=<value> -generic <name>=<value> ... です。
[-verilog_define]	Verilog の `define 文の値を指定します。構文は、-verilog_define <macro_name>[=<macro_text>] -verilog_define <macro_name>[=<macro_text>] ... です。
[-flatten_hierarchy]	LUT マップ中に階層をフラット化します。有効な値は full、auto、none、rebuilt で、デフォルトは rebuilt です。
[-gated_clock_conversion]	クロック ゲーティング ロジックをフリップフロップ イネーブルに変換します。有効な値は off、on、auto で、デフォルトは off です。
[-directive]	合成の目標を指定します。有効な値は default、RuntimeOptimized、AreaOptimized_high、AreaOptimized_medium、AlternateRoutability、AreaMapLargeShiftRegToBRAM、AreaMultThresholdDSP、FewerCarryChains、PerformanceOptimized で、デフォルトは default です。
[-rtl]	エラボレーションを実行し、RTL デザインを開きます。
[-bufg]	合成で使用するグローバル クロック バッファの最大数を指定します。デフォルトは 12 です。

名前	説明
<code>[-no_lc]</code>	LUT の組み合わせをディスエーブルにします。LUT ペアを 1 つのデュアル出力 LUT に組み合わせません。
<code>[-fanout_limit]</code>	ファンアウトの最大数を指定します。このオプションは制御信号 (セット、リセット、クロック イネーブルなど) には適用されません。制御信号を複製する必要がある場合は、MAX_FANOUT を使用してください。デフォルトは 10000 です。
<code>[-shreg_min_size]</code>	SRL にマップされるレジスタ チェーンの最小長を指定します。デフォルトは 3 です。
<code>[-mode]</code>	デザイン モードを指定します。有効な値は default、out_of_context で、デフォルトは default です。
<code>[-fsm_extraction]</code>	FSM 抽出エンコードを指定します。有効な値は off、one_hot、sequential、johnson、gray、user_encoding、auto で、デフォルトは auto です。
<code>[-rtl_skip_ip]</code>	デザインの RTL エラボレーションでサブデザイン チェックポイントを除外します。-rtl オプションを使用している場合にのみ有効です。
<code>[-rtl_skip_constraints]</code>	エラボレート済みデザインに対して制約を読み込んで検証しません。-rtl オプションを使用している場合にのみ有効です。
<code>[-keep_equivalent_registers]</code>	同じロジックがソースとなるレジスタが統合されないようにします。合成の KEEP 属性を使用してもレジスタの統合を回避できます。
<code>[-resource_sharing]</code>	算術演算子を共有します。有効な値は auto、on、off で、デフォルトは auto です。
<code>[-cascade_dsp]</code>	DSP ブロック出力を合計する加算器のインプリメント方法を制御します。有効な値は auto、tree、force で、デフォルトは auto です。
<code>[-control_set_opt_threshold]</code>	同期制御セット最適化のしきい値の制御セット数を削減します。有効な値は、auto および負でない整数です。値が大きいくほど、実行される制御セットの最適化が多くなり、制御セットの数が少なくなります。制御セットの最適化を完全にディスエーブルにするには、0 に設定します。Default: auto
<code>[-incremental]</code>	インクリメンタル フロー用の DCP ファイルの名前を指定します。
<code>[-max_bram]</code>	デザインで使用可能なブロック RAM の最大数を指定します。-1 に指定すると、選択しているパーツで使用可能な最大数が選択されます。デフォルトは -1 です。
<code>[-max_uram]</code>	デザインで使用可能な UltraRAM ブロックの最大数を指定します。-1 に指定すると、選択しているパーツで使用可能な最大数が選択されます。デフォルトは -1 です。
<code>[-max_dsp]</code>	デザインで使用可能なブロック DSP の最大数を指定します。-1 に指定すると、選択しているパーツで使用可能な最大数が選択されます。デフォルトは -1 です。
<code>[-max_bram_cascade_height]</code>	ツールでカスケード接続可能なブロック RAM の最大数を指定します。-1 に指定すると、選択しているパーツで使用可能な最大数が選択されます。デフォルトは -1 です。
<code>[-max_uram_cascade_height]</code>	ツールでカスケード接続可能な URAM の最大数を指定します。-1 に指定すると、選択しているパーツで使用可能な最大数が選択されます。デフォルトは -1 です。
<code>[-retiming]</code>	組み合わせゲートまたは LUT の反対側にレジスタを自動的に移動することにより (レジスタのバランス調整)、回路のパフォーマンスを向上します。元の動作および回路のレイテンシは保持され、RTL ソースを変更する必要はありません。
<code>[-no_srlextract]</code>	シフト レジスタを抽出せず、単純なレジスタとしてインプリメントされるよう指定します。

名前	説明
<code>[-assert]</code>	VHDL アサート文の評価をイネーブルにします。エラーの場合は、合成フローが停止してエラー メッセージが表示されます。
<code>[-no_timing_driven]</code>	タイミングドリブン モードで実行しません。
<code>[-sfcu]</code>	合成をシングル ファイル コンパイル ユニット モードで実行します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Tools (ツール)

説明

Vivado 合成エンジンを直接起動し、プロジェクト モードまたは非プロジェクト モードのデザインをコンパイルおよび合成します。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892) を参照してください。

非プロジェクト モードでは、`synth_design` コマンドで Vivado 合成を直接実行できます。



ヒント: `synth_design` コマンドをマルチスレッドで実行し、プロセスを高速化できます。
`general.maxThreads` パラメーターの設定に関する詳細は、`set_param` コマンドを参照してください。

プロジェクト モードでは、`create_run` コマンドを使用して作成した合成 run から合成を実行します。
`launch_runs` コマンドを使用して run を実行すると、Vivado 合成に対しては `synth_design` が呼び出されます。

`synth_design` コマンドは、RTL ソース ファイルをエラボレートし、エラボレート済みデザインを開くのにも使用できます。

```
synth_design -rtl -name rtl_1
```

このコマンドを実行すると、合成プロセスが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-name <arg>` (オプション): 合成完了後に Vivado ツールで開いたときに割り当てられる合成済みデザインの名前を指定します。この名前は参照用であり、デザインの最上位またはデザインに含まれるロジックとは関係ありません。

`-part <arg>` (オプション): デザインに使用するターゲット ザイリンクス デバイスを指定します。このオプションを指定しない場合、プロジェクトで設定されたデフォルトのパーツが使用されます。

`-constrset <arg>` (オプション): デザインを合成する際に使用する XDC 制約の名前を指定します。Vivado 合成では XDC を使用する必要があり、UCF はサポートされません。`-constrset` オプションには、既存の制約ファイルセットを指定する必要があります。新しいファイルセットを作成することはできません。新しいファイルセットを作成する場合は、`create_fileset` コマンドを使用します。

`-top <arg>` (オプション): デザイン階層の最上位モジュールを指定します。



重要: `find_top` コマンドを使用して `-top` オプションを指定する場合は、`find_top` で複数の候補が返された場合に 1 つのみが指定されるようにしてください。例を参照してください。

`-include_dirs <args>` (オプション): Verilog ``include` ファイルを検索するディレクトリを指定します。複数のディレクトリを指定するには、次のように記述します。

```
-include_dirs {C:/data/include1 C:/data/include2}
```

`-generic <name>=<value>` (オプション): VHDL ジェネリック エンティティまたは Verilog パラメーターの値を定義します。`-generic` オプションは、RTL デザイン ソースで割り当てられているパラメーター値を変更する場合に使用できますが、変更できるのはデザイン最上位のパラメーターのみです。下位モジュールのパラメーターは、インスタネーション時にのみ変更可能で、`-generic` オプションでは変更できません。`-generic` オプションの構文は `<name>=<value>` で、ジェネリックまたはパラメーターの名前とその値を指定します。`synth_design` コマンドで、各ジェネリックまたはパラメーター値に対してそれぞれ `-generic` オプションを指定します。

```
synth_design -generic width=32 -generic depth=512 ...
```



重要: ブール型または `std_logic` VHDL ジェネリック型に 2 進数値を指定する場合は、標準 VHDL フォーマットではなく Verilog ビット フォーマットを使用して値を指定する必要があります。

```
0 = 1'b0
01010000 = 8'b01010000
```

`-verilog_define <name>=<text>` (オプション): Verilog の ``define` 文および ``ifdef` 文の値を指定します。`-verilog_define` オプションの構文は `<name>=<text>` で、`define` 文の名前とその値を指定します。この引数は、1 つの `synth_design` コマンドで複数回使用できます。

```
synth_design -verilog_define <name>=<value> -verilog_define
<name>=<value> ...
```

`-flatten_hierarchy <arg>` (オプション): LUT マップ中にデザインの階層をフラット化します。有効な値は次のとおりです。

- `rebuilt`: 合成が完了した後に RTL デザインの元の階層を再構築するよう試みます。これがデフォルト設定です。
- `full` (オプション): デザインの階層をフラットにします。
- `none` (オプション): デザインの階層をフラットにしません。これによりデザインの階層が保持され増すが、合成ツールで実行できるデザインの最適化が制限されます。

`-gated_clock_conversion <arg>` (オプション): クロック ゲーティング ロジックを変換して、フリップフロップ イネーブル ピンを使用します。この最適化によりロジックが削除され、ネットリストが簡略化されます。詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) の `GATED_CLOCK` プロパティに関する説明を参照してください。有効な値は次のとおりです。

- `off`: RTL デザインで `GATED_CLOCK` プロパティが使用されているかどうかにかかわらず、合成中のクロック ゲーティング ロジックの変換をディスエーブルにします。
- `on`: RTL デザインで `GATED_CLOCK` プロパティが使用されている場合にクロック ゲーティング ロジックを変換します。
- `auto`: RTL に `GATED_CLOCK` プロパティがある場合、または有効なクロック制約が設定されたゲートが検出された場合に、ゲーテッド クロック変換が自動的に実行されるようにします。

`-directive <arg>` (オプション): 特定のデザイン目標を達成するよう合成します。1 つの `synth_design` コマンドに対して 1 つのモードのみを指定できます。値では大文字/小文字が区別されます。有効な値は次のとおりです。

- `default`: デフォルトの合成プロセスを実行します。

- `runtimeoptimized`: 実行する最適化を少なくし、いくつかの RTL 最適化を削除して、合成のランタイムを短縮します。
- `AreaOptimized_high`: `AreaMapLargeShiftRegToBRAM`、`AreaThresholdUseDSP` を含む一般的なエリア最適化を実行します。
- `AreaOptimized_medium`: 三項加算インプリメンテーションの強制、コンパレータのキャリー チェーンの使用に対する新しいしきい値の適用、エリアで最適化されたマルチプレクサーのインプリメントを含む、全般的なエリア最適化を実行します。
- `AlternateRoutability`: 配線性を向上するアルゴリズムを実行します。MUXF および CARRY の使用数が削減します。
- `AreaMapLargeShiftRegToBRAM`: 大型のシフト レジスタを検出し、専用ブロック RAM を使用してインプリメントします。
- `AreaMultThresholdDSP`: 専用 DSP ブロック推論のしきい値を低くします。
- `FewerCarryChains`: オペランド サイズのしきい値を高くし、キャリー チェーンの代わりに LUT を使用します。
- `PerformanceOptimized`: ロジック段数の削減を含む全般的なタイミング最適化を実行します。エリアが増加する可能性があります。

`-rtl` (オプション): HDL ソース ファイルをエラボレートし、RTL デザインを開きます。IP カタログからの IP などの独立階層 (OOC) モジュールを使用するデザインでは、その OOC モジュールの合成済みデザイン チェックポイント (DCP) がデザインにインポートされ、関連の制約ファイル (XDC) がエラボレート済みデザインにインポートされます。`-rtl_skip_ip` および `-rtl_skip_constraints` オプションを使用すると、このデフォルト動作をディスエーブルにできます。

`-rtl_skip_ip` (オプション): このオプションを使用するには、`-rtl` オプションも使用する必要があります。RTL デザインをエラボレートする場合に、デザインの OOC モジュールの DCP ファイルを読み込まず、スタブ ファイルを読み込んで OOC モジュールをブラック ボックスとして処理します。このオプションを使用すると、エラボレーション速度が大幅に向上します。



ヒント: OOC 合成 run は、エラボレーション中に読み込む DCP ファイルを生成する場合、またはブラック ボックスに必要なスタブ ファイルを生成する場合の両方で必要です。

`-rtl_skip_constraints` (オプション): このオプションを使用するには、`-rtl` オプションも使用する必要があります。RTL デザインをエラボレートする場合に、デザイン制約 (XDC) ファイルをエラボレート済みデザインに読み込みません。このオプションを使用すると、エラボレーション速度が大幅に向上します。

`-bufg <arg>` (オプション): 合成でクロック ネットに使用するグローバル クロック バッファの最大数を指定します。1 からターゲット デバイスに含まれる BUFG の数までの値を指定します。デフォルト値は 12 です。



ヒント: Vivado 合成では、RTL ソースにインスタンス化されている BUFG も含め、指定した数までの BUFG が推論されます。たとえば、`-bufg 12` を設定し、RTL に 3 つの BUFG がインスタンス化されているとすると、ツールで 9 個までの BUFG を推論可能です。

`-no_lc` (オプション): デフォルトの LUT の組み合わせ機能をディスエーブルにします。

`-fanout_limit <arg>` (オプション): 合成中に適用される最大ネット ファンアウトのターゲット制限を指定します。デフォルト値は 10,000 です。このオプションは、ロードの数が内部制限を越える場合に Vivado 合成により適用されますが、ガイドラインとしてのみ使用され、厳密な要件ではありません。特定の信号のファンアウトを厳密に制御する必要がある場合は、MAX_FANOUT プロパティを使用してください。



重要: `-fanout_limit` オプションは制御信号 (セット、リセット、クロック イネーブルなど) には適用されません。これらの信号を複製する必要がある場合は、MAX_FANOUT プロパティを使用してください。

`-shreg_min_size <arg>` (オプション): SRL にマップされるレジスタ チェーンの最小長を整数で指定します。デフォルト値は 3 です。

`-mode [default | out_of_context]` (オプション): `out_of_context` モードを指定すると、アウト オブ コンテキスト (OOC) デザイン フローで使用する IP モジュールまたはブロック モジュールが合成されます。このモードでは、モジュールに I/O バッファは挿入されず、モジュールが OOC とマークされます。ブロックは、解析目的でインプリメントすることもできます。詳細は、『Vivado Design Suite ユーザー ガイド: IP を使用した設計』 (UG896) または『Vivado Design Suite ユーザー ガイド: 階層デザイン』 (UG905) を参照してください。

`-fsm_extraction <arg>` (オプション): Vivado 合成では、デフォルトでは有限状態 マシン (FSM) エンコードは自動 (`auto`) に設定されています。このオプションは状態 マシンの識別を有効にし、適用するエンコードを指定します。有効な値は `off`、`one_hot`、`sequential`、`johnson`、`gray`、`auto` です。自動エンコード (`auto`) に設定すると、各状態 マシンに最適なエンコードが選択されます。この場合、同じデザインの異なる FSM に異なるエンコードが使用されることがあります。

注記: Vivado 合成で有限状態 マシンの抽出をディスエーブルにするには、`-fsm_extraction off` を使用します。これにより、`FSM_ENCODING` プロパティは無効になります。

`-keep_equivalent_registers` (オプション): KEEP プロパティのように機能し、最適化中にレジスタが統合されないようにします。

`-resource_sharing <arg>` (オプション): 加算器や減算器などの算術演算子を異なる信号間で共有します。このオプションをオンにすると、エリア使用率が向上する場合があります。有効な値は `auto`、`on`、`off` で、デフォルトは `auto` です。

`-cascade_dsp [auto | tree | force]` (オプション): DSP ブロックの出力を加算する加算器のインプリメント方法を指定します。有効な値は `auto`、`tree`、`force` です。デフォルトは `auto` です。

`-control_set_opt_threshold <arg>` (オプション): 同期制御セット最適化のしきい値の制御セット数を削減します。制御セットとして使用し始める前に、制御セットのファンアウトの大きさを指定します。たとえば、`-control_set_opt_threshold` オプションを 10 に設定すると、5 つのレジスタにのみファンアウトする同期リセットは、レジスタのリセット ラインを使用するのではなく、D 入力ロジックに移動されます。`-control_set_opt_threshold` オプションを 4 に設定すると、リセット ラインが使用されます。有効な値は、`auto` または 0 ~ 16 の整数です。デフォルト設定は `auto` で、選択したデバイス アーキテクチャによってことなるしきい値が使用されます。

`-incremental` (オプション): インクリメンタル コンパイル フローに使用する DCP ファイルを指定します。インクリメンタル合成フローでは、インクリメンタル DCP からのネットリストが現在のデザインのデザイン オブジェクトに適用され、可能な場合は既存の合成結果が再利用されます。

`-max_bram <arg>` (オプション): 合成中にデザインに追加するブロック RAM の最大数を指定します。1 からターゲット デバイスに含まれる BRAM の数までの値を指定します。`-1` に設定すると、デバイスに含まれるブロック RAM の数を超えることはありません。デフォルト値は `-1` です。

注記: 0 に設定すると、デザインにブロック RAM は推論されませんが、この設定は推奨されません。

`-max_uram <arg>` (オプション): 合成中にデザインに追加する UltraRAM ブロック (URAM) の最大数を指定します。1 からターゲット デバイスに含まれる URAM の数までの値を指定します。`-1` に設定すると、デバイスに含まれる URAM ブロックの数を超えることはありません。デフォルト値は `-1` です。

注記: 0 に設定すると、デザインに URAM は推論されませんが、この設定は推奨されません。

`-max_dsp <arg>` (オプション): 合成中にデザインに追加する DSP の最大数を指定します。1 からターゲット デバイスに含まれる DSP の数までの値を指定します。`-1` に設定すると、デバイスに含まれる DSP の数を超えることはありません。デフォルト値は `-1` です。

注記: 0 に設定すると、デザインに DSP は推論されませんが、この設定は推奨されません。

`-max_bram_cascade_height <arg>` (オプション): ツールでカスケード接続可能なブロック RAM の最大数を指定します。デフォルト値は -1 で、Vivado 合成によりターゲット パーツで使用可能な最大数が判断されます。デフォルト値は -1 です。

`-max_uram_cascade_height <arg>` (オプション): ツールでカスケード接続可能な URAM の最大数を指定します。デフォルト値は -1 で、Vivado 合成によりターゲット パーツで使用可能な最大数が判断されます。デフォルト値は -1 です。

`-retiming` (オプション): 組み合わせゲートまたは LUT の反対側にレジスタを自動的に移動することにより (レジスタのバランス調整)、回路のパフォーマンスを向上します。元の動作および回路のレイテンシは保持され、RTL ソースを変更する必要はありません。

`-no_srlextract` (オプション): シフト レジスタを抽出せず、単純なレジスタとしてインプリメントされるよう指定します。

`-assert` (オプション): VHDL アサート文の評価をイネーブルにします。エラーの場合は、合成フローが停止してエラー メッセージが表示されます。

`-no_timing_driven` (オプション): デフォルトのタイミング ドリブン合成アルゴリズムをディスエーブルにします。これにより合成の実行時間が短縮されますが、合成に対するタイミングの影響は無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、`set_property` コマンドを使用してアクティブ プロジェクトのターゲット パーツを指定し、ソース ファイルをエラボレートして、RTL デザインを開いています。

```
set_property part xc7vx485tffg1158-1 [current_project]
synth_design -rtl -name rtl_1
```

注記: この例では、デフォルトのソース セット、制約セット、パーツが使用されます。

次の例では、`find_top` コマンドを使用して合成用に現在のデザインの最上位モジュールを定義しています。

```
synth_design -top [lindex [find_top] 0]
```

注記: `find_top` では複数の候補が返されますが、インデックス 0 を指定すると合成に最適な候補が選択されます。

次の例では、最上位モジュールとターゲット デバイスを定義し、階層をフラット化しないように指定して、現在のデザインを合成します。合成 run の結果がネットリスト デザインとして開きます。

```
synth_design -top top -part xc7k70tfbg676-2 -flatten_hierarchy none
open_run synth_1 -name netlist_1
```

関連項目

- [create_ip_run](#)
- [create_run](#)
- [current_design](#)
- [current_project](#)
- [find_top](#)
- [open_run](#)
- [opt_design](#)
- [set_property](#)

synth_ip

IP の合成ネットリストを生成します。

構文

```
synth_ip [-force] [-quiet] [-verbose] <objects>
```

使用法

名前	説明
<code>[-force]</code>	ネットリストを強制的に再生成します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	ネットリストを生成する必要があるすべてのオブジェクトを指定します。

カテゴリ

Project (プロジェクト)、IPFlow (IP フロー)

説明

このコマンドは、非プロジェクト フローでアウト オブ コンテキスト (OOC) IP フローをサポートするために合成済み デザイン チェックポイント ファイル (DCP) を生成するか、または OOC 階層デザイン フローで IP モジュールを合成 およびインプリメントするために使用します。 `get_ips` コマンドを使用して IP オブジェクトを指定するか、 `get_files` コマンドを使用して IP コア ファイル (XCI) を指定します。



重要: この機能をイネーブルにするには、XCI ファイルに対して `set_property` コマンドを使用して `GENERATE_SYNTH_CHECKPOINT` プロパティを 1 に設定することにより、IP コアを OOC 生成用にマークする必要があります。

プロジェクト ベースのデザインでは、`create_ip_run` および `launch_runs` コマンドを使用します。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』(UG892) を参照してください。

`synth_ip` コマンドは、IP コアを合成する前に必要なターゲット ファイルを自動的に生成します。IP を合成するのに必要なソース ファイルは、IP run ディレクトリにコピーされます。完了すると、新しく生成された OOC ターゲット ファイル (DCP、スタブ ファイル、論理シミュレーション ネットリストなど) は IP コアに登録されます。

引数

`-force` (オプション): 指定の IP に対する出力ファイルが生成されており、すべて最新であっても、指定の IP オブジェクトを強制的に再合成します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (必須): 合成する IP オブジェクトを 1 つ以上指定します。`get_ips` コマンドを使用して IP コアを指定するか、`get_files` コマンドを使用して XCI ファイルを指定します。

例

次の例では、指定の IP オブジェクトを合成し、合成済みコアが最新である場合でもネットリストを再生成しています。

```
synth_ip [get_ips char_fifo] -force
```

関連項目

- [create_ip_run](#)
- [get_files](#)
- [get_ips](#)
- [launch_runs](#)
- [synth_design](#)
- [read_checkpoint](#)

tie_unused_pins

未使用のセル ピンを接続します。

構文

```
tie_unused_pins [-of_objects <args>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-of_objects]</code>	指定したセルの未使用ピンを接続します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Netlist \(ネットリスト\)](#)

説明

開いている合成済みデザインまたはインプリメント済みデザインで、セルの未接続のピンを接続します。このコマンドでは、内部プロセスを使用して、ピンを High に接続するか Low に接続するかを特定します。

このコマンドは、`create_cell` コマンドでネットリストに追加されたセルの未接続ピンを接続するために使用します。

引数

`-of_objects <args>` (オプション): 指定したセル オブジェクトの未使用のピンを接続します。

注記: `-of_objects` オプションでは、オブジェクトを名前で指定するのではなく、`get_*` コマンド (`get_cells`、`get_pins` など) を使用して指定する必要があります。また、`-of_objects` を検索パターン (`<pattern>`) と共に使用することはできません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、セルの未使用のピンを、その使用状況によって High または Low に接続しています。

```
tie_unused_pins -of_objects [get_cells cpuEngine]
```

関連項目

- [create_cell](#)
- [create_pin](#)
- [get_cells](#)
- [get_pins](#)

undo

前のコマンドの実行を取り消します。

構文

```
undo [-list] [-quiet] [-verbose]
```

戻り値

取り消すことができるタスクのリスト (-list を使用した場合)

使用法

名前	説明
[-list]	取り消すことができるタスクのリストを表示します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明



重要: undo および redo コマンドは Vivado IDE で使用するためのもので、Tcl スクリプトでデザインの以前のステートを復元するために使用することはお勧めしません。デザインを特定の状態に戻すには、write_checkpoint コマンドを使用してデザイン チェックポイントを保存し、read_checkpoint コマンドを使用して復元できるようにします。

前に実行したコマンドを取り消します。このコマンドを繰り返して使用し、一連のコマンドを取り消すことができます。

startgroup および endgroup コマンドを使用してコマンド グループを作成した場合、undo コマンドでコマンドグループがシーケンスとして取り消されます。undo コマンドは endgroup コマンドから開始し、startgroup コマンドに到達するまで繰り返し替えされます。

あるコマンドに対して undo を実行し、その後そのコマンドを実行することにした場合は、redo コマンドを使用できます。

引数

-list (オプション): 取り消すことができるコマンドのリストを返します。undo コマンドを使用すると、コマンドのリストを順にさかのぼってコマンドが取り消されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、取り消すことが可能なコマンドのリストが返されます。

```
undo -list
```

関連項目

- [redo](#)
- [startgroup](#)
- [endgroup](#)

ungroup_bd_cells

階層セル内のセルのグループを親セルに移動し、階層セルを削除します。セル間の接続は保持されます。これらのセルとその他のセルの接続は、階層セルをまたぐことにより保持されます。

構文

```
ungroup_bd_cells [-prefix <arg>] [-quiet] [-verbose] [<cells>...]
```

戻り値

正しく実行された場合は 0。

使用法

名前	説明
[-prefix]	セルに追加する接頭辞を指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<cells>]	セル名を検索するパターンを指定します。デフォルトは * です。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

group_bd_cells または move_bd_cells コマンドで階層モジュールにグループ化された IP インテグレーター セルのグループを解除します。選択した階層モジュール内のセルを親セルのレベルに移動し、階層モジュールを削除します。

選択したセル間の接続は保持されます。これらのセルとほかのセルの間の接続は、不必要なサブシステムのポートおよびピンを削除することにより自動的に保持されます。

このコマンドが正常に実行された場合は 0 が返され、正常に実行されなかった場合はエラーが返されます。

引数

-prefix <arg> (オプション): 階層を 1 レベル上に移動するセルに適用する接頭辞を指定します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<cells> (必須): `group_bd_cells` または `move_bd_cells` コマンドで定義された階層モジュールを指定します。このコマンドで一度に指定できるのは 1 つの階層モジュールのみです。

例

次に例を示します。

```
ungroup_bd_cells -prefix up2_ [get_bd_cells myMod2]
```

関連項目

- [get_bd_cells](#)
- [group_bd_cells](#)
- [move_bd_cells](#)

unhighlight_objects

現在ハイライトされているオブジェクトのハイライトを解除します。

構文

```
unhighlight_objects [-color_index <arg>] [-rgb <args>] [-color <arg>]
                    [-leaf_cells] [-quiet] [-verbose] [<objects>]
```

使用法

名前	説明
[-color_index]	色を色インデックスで指定します。
[-rgb]	色を RGB で指定します。
[-color]	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
[-leaf_cells]	最下位セルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	ハイライトを解除するオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

このコマンドは GUI モードで使用します。highlight_objects コマンドでハイライトしたオブジェクトのハイライトを解除します。

このコマンドでは、下に説明する色オプションがサポートされています。指定の色でハイライトされているオブジェクトすべてのハイライトを解除するために使用できます。例を参照してください。

引数

-color_index <arg> (オプション): 有効な値は 1 ～ 20 の整数で、ハイライトを解除する色を色インデックスで指定します。色インデックスは、[Tools]→[Settings] をクリックし、[Settings] ダイアログ ボックスの [Colors]→[Highlight] ページで定義します。テーマの設定方法の詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) を参照してください。

-rgb <args> (オプション): ハイライトを解除する色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定します。

-color <arg> (オプション): ハイライトを解除する色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

-leaf_cells <arg> (オプション): ハイライトを解除するセルの名前またはセル オブジェクトを 1 つまたは複数指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): ハイライトを解除するオブジェクトを指定します。オブジェクトを指定しない場合は、指定した色でハイライトされているオブジェクトすべてのハイライトが解除されます。色を指定しない場合は、すべてのオブジェクトのハイライトが解除されます。

例

次の例は、指定のセル オブジェクトのハイライトを解除します。

```
unhighlight_objects -leaf_cells [get_cells cpuEngine/*]
```

次の例は、現在黄色でハイライトされているオブジェクトのハイライトを解除します。

```
unhighlight_objects -color yellow
```

関連項目

- [get_selected_objects](#)
- [highlight_objects](#)

unmark_objects

現在マークされているアイテムのマークを解除します。

構文

```
unmark_objects [-rgb <args>] [-color <arg>] [-quiet] [-verbose] [<objects>]
```

使用法

名前	説明
<code>[-rgb]</code>	色を RGB で指定します。
<code>[-color]</code>	有効な値は、red、green、blue、magenta、yellow、cyan、および orange です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<objects>]</code>	マークを解除するオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

`mark_objects` コマンドでマークしたオブジェクトのマークを解除します。このコマンドは GUI モードで使います。

このコマンドでは、下に説明する色オプションがサポートされています。指定の色でマークされているオブジェクトすべてのマークを解除するために使用できます。ただし、指定のオブジェクトのマークを解除するのにこれらのオプションを指定する必要はありません。例を参照してください。

引数

`-rgb <args>` (オプション): マークを解除する色を、RGB コードを使用して {R G B} の形式で指定します。たとえば、{255 255 0} は黄色を指定します。

`-color <arg>` (オプション): マークを解除する色を色の名前で指定します。指定可能な値は、red、green、blue、magenta、yellow、cyan、および orange です。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects> (オプション): マークを解除する 1 つまたは複数のオブジェクトを指定します。オブジェクトを指定しない場合は、指定した色でマークされているオブジェクトすべてのマークが解除されます。色を指定しない場合は、すべてのオブジェクトのマークが解除されます。

例

次の例では、選択したオブジェクトのマークが解除されます。

```
unmark_objects [get_selected_objects]
```

次の例では、現在黄色でマークされているオブジェクトのマークが解除されます。

```
unmark_objects -color yellow
```

関連項目

- [get_selected_objects](#)
- [mark_objects](#)

unplace_cell

1 つまたは複数のインスタンスの配置を解除します。

構文

```
unplace_cell [-quiet] [-verbose] <cell_list>...
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><cell_list></code>	配置を解除するセルを指定します。

カテゴリ

Floorplan (フロアプラン)

説明

指定したセルを現在配置されている配置サイトから解除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<cell_list>` (必須): デバイスから配置を解除するセルを 1 つまたは複数指定します。

例

次の例では、指定したセルの配置を解除しています。

```
unplace_cell {fftEngine/fftInst/ingressLoop[6].ingressFifo/buffer_fifo/i_4773_12897}
```

次の例では、複数のセルの配置を解除しています。

```
unplace_cell {div_cntr_reg_inferredi_4810_15889 div_cntr_reg[0] div_cntr_reg[1]}
```

関連項目

- [create_cell](#)
- [place_cell](#)
- [remove_cell](#)

unregister_proc

登録されている Tcl プロシージャを登録解除します。

構文

```
unregister_proc [-quiet] [-verbose] <tasknm>
```

戻り値

なし

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><tasknm></code>	登録解除する Tcl タスクの名前を指定します。タスクはプロシージャのラッパーである必要があります。

カテゴリ

[Tools \(ツール\)](#)

説明

Vivado Design Suite インタープリターから Tcl コマンド (<tasknm>) を登録解除します。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<tasknm>` (必須): 登録解除する Tcl タスクの名前を指定します。タスクは登録済み Tcl プロシージャのラッパーである必要があります。

例

次の例では、findCmd Tcl タスクを登録解除しています。

```
unregister_proc findCmd
```

関連項目

- [register_proc](#)

unselect_objects

現在選択されているアイテムの選択を解除します。

構文

```
unselect_objects [-quiet] [-verbose] [<objects>]
```

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	選択を解除するオブジェクトを指定します。

カテゴリ

[GUIControl \(GUI 制御\)](#)

説明

`select_objects` コマンドで選択したオブジェクトの選択を解除します。

プライマリ オブジェクトとセカンダリ オブジェクト両方の選択が解除されます。セカンダリ オブジェクトは、[Tools]→[Settings] をクリックし、[Settings] ダイアログ ボックスの [Selection Rules] ページで定義します。選択規則の設定方法は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』 (UG893) を参照してください。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (オプション): 選択を解除する 1 つまたは複数のオブジェクトを指定します。オブジェクトを指定しない場合は、すべてのオブジェクトの選択が解除されます。

例

次の例では、デバイス上の指定したサイトの選択が解除されます。

```
unselect_objects [get_sites SLICE_X56Y214]
```

次の例では、現在選択されているオブジェクトすべての選択が解除されます。

```
unselect_objects
```

関連項目

- [get_selected_objects](#)
- [select_objects](#)

update_clock_routing

グローバル クロックのクロック配線が配置後に変更された場合にアップデートします。

構文

```
update_clock_routing [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Tools \(ツール\)](#)

説明

`update_clock_routing` コマンドは、UltraScale アーキテクチャをターゲットとするデザインのグローバル クロックすべての配線構造を手動でアップデートするアドバンス コマンドです。個別のクロックではなく、デザインに含まれるすべてのグローバル クロックに対して実行されます。

UltraScale および UltraScale+ デザインのクロッキング アーキテクチャは柔軟性が高いため、グローバル クロックの配線には 2 段階のプロセスが必要です。まず Vivado 配置により、クロック ソース領域からデスティネーション クロック領域にグローバル クロックを配線するのに必要な配線リソースが割り当てられます (CLOCK_ROOT または USER_CLOCK_ROOT)。次に、Vivado 配線によりクロック ネットの配線ギャップが埋められます。これらの 2 つの段階の間の構造は、ギャップ ツリーと呼ばれます。この構造では、各グローバル クロック ネットにベース配線リソースは割り当てられていますが、配線リソースが割り当てられていない配線ギャップがあります。

ギャップ ツリーが構築された後、配線によりすべての最下位プリミティブまでの残りのクロック ネットワークが配線され、配線ギャップが埋められます。インプリメンテーション run では、グローバル クロック配線は自動的に処理されます。ただし、たとえばクロック ネットの USER_CLOCK_ROOT プロパティを変更するなどしてインプリメンテーション後にクロック ツリーが変更された場合、Vivado ツールでギャップ ツリーを適切に再構築して配線ギャップを埋めるために、`update_clock_routing` コマンドが必要な場合があります。

次に例を示します。

- グローバル クロックのクロック ルートを移動する。
- グローバル クロックが配置されていないクロック領域にグローバル クロックのロードを追加または移動し、アップデートされたデザインでタイミング解析を実行する。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、指定したクロック ネットの `USER_CLOCK_ROOT` プロパティをアップデートし、ネットの配線を解除して、クロック配線をアップデートしています。

```
set_property USER_CLOCK_ROOT X1Y0 [get_nets {clk1 clk2}]
route_design -unroute -nets [get_nets {clk1 clk2}]
update_clock_routing
```



重要: クロック配線をアップデートする前に、クロック ネットの既存のクロック配線をクリーンアップするため、`unroute` コマンドが必要です。

関連項目

- [route_design](#)
- [set_property](#)

update_compile_order

デザイン グラフに基づいて、ファイルセットのコンパイル順および場合によっては最上位をアップデートします。

構文

```
update_compile_order [-force_gui] [-fileset <arg>] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-force_gui]</code>	GUI からインタラクティブに実行している場合でも、このコマンドを実行します。
<code>[-fileset]</code>	デザイン グラフに基づいてアップデートするファイルセットを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

現在のプロジェクトまたは指定のファイルセットでデザイン ソースのコンパイル順をアップデートします。

引数

`-force_gui`: Vivado IDE の GUI モードのコンパイル順をアップデートします。

`-fileset <arg>`: 指定のファイルセットのコンパイル順をアップデートします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、シミュレーション ファイルセットのソース ファイルのコンパイル順をアップデートしています。

```
update_compile_order -fileset sim_1
```

関連項目

- [add_files](#)
- [import_files](#)

update_design

現在のデザインのネットリストをアップデートします。

構文

```
update_design -cells <args> [-strict] [-from_file <arg>]  
              [-from_design <arg>] [-from_cell <arg>] [-black_box] [-buffer_ports]  
              [-quiet] [-verbose]
```

使用法

名前	説明
-cells	新しいサブネットリストでアップデートするセルを指定します。
[-strict]	セルを置き換えるのに完全に一致したポートを必要とします。このオプションを設定しない場合、余分なポートが許可されます。
[-from_file]	新しいサブネットリストを含むファイルの名前を指定します。
[-from_design]	新しいサブネットリストを含む、開いているネットリスト デザインの名前を指定します。
[-from_cell]	新しいサブネットリストを定義する from_design のセルの名前を指定します。
[-black_box]	セルをブラック ボックスに変換します。
[-buffer_ports]	ブラック ボックスのすべてのポートにバッファを挿入します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

メモリ内のデザインをアップデートし、指定のセルの現在のネットリストを指定のファイルまたは開いている別のデザインからのネットリストに置き換えるか、またはセルをブラック ボックス セルに変換します。

update_design コマンドでは、1 つのインスタンスまたはマスター セルのすべてのインスタンスをアップデートできます。

メモリ内のデザインのみが新しいネットリストに変更されます。write_checkpoint コマンドを使用してデザインを保存してください。そうしないと、プロジェクトを閉じたとき、またはツールを終了したときにアップデートは失われます。

引数

-cells <arg> (必須): 指定のネットリストでアップデートするの名前のリストまたはセル オブジェクト指定します。

`-strict` (オプション): 新しいネットリストがインポート先のセルと同じポートを持つことを必須にします。新しいネットリストに指定のセルに必要なポートがすべてあるかどうかチェックされます。`-strict` オプションを使用しない場合、追加ポートは許容されます。

`-from_file` (オプション): 新しいネットリストを含むファイルの名前を指定します。構造 Verilog ネットリスト (.v) または EDIF ネットリスト (.edf) ファイルを指定できます。

注記: `-from_file` と `-from_design` オプションを同時に使用することはできません。

`-from_design` (オプション): 現在のプロジェクトの開いている別のデザインからネットリストをインポートします。デザインを別のプロセスではなく現在のツール セッションで開いている必要があります。

`-from_cell` (オプション): `-from_design` オプションで指定したデザインのセルの名前を指定します。現在のデザインのセルが指定したセルからのネットリストでアップデートされます。デフォルトでは、`-from_design` で指定したデザインの最上位セルが使用されます。

注記: このオプションは、`-from_design` オプションを使用している場合にのみ使用可能です。

`-black_box` (オプション): 指定のセルをブラック ボックス セルに変換します。

`-buffer_ports` (オプション): ブラック ボックス セルのすべてのポートにバッファを挿入します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ブラック ボックス セルを指定のファイルからのネットリストに置き換えます。

```
update_design -from_file C:/Data/cell_contents.v -cell black_box_cell
```

次の例では、`arnd4` セルを指定した Verilog ネットリストでアップデートしています。

```
update_design -cell arnd4 -from_file C:/Data/round_4.v
```

次の例では、現在のデザインに含まれる `arnd4` セルを、指定したデザインの同じセルからのネットリストでアップデートしています。

```
update_design -cell arnd4 -from_design netlist_2 -from_cell arnd4
```

関連項目

- [get_cells](#)
- [write_checkpoint](#)

update_files

プロジェクトのファイルを、指定のファイルまたはディレクトリに基づいてアップデートします。

構文

```
update_files [-from_files <args>] [-norecurse] [-to_files <args>]  
             [-filesets <args>] [-force] [-report_only] [-quiet] [-verbose]
```

戻り値

アップデートされたファイルのリスト

使用法

名前	説明
[-from_files]	アップデートに使用するファイルおよびディレクトリを指定します。
[-norecurse]	下位ディレクトリでは検索を実行しないよう指定します。
[-to_files]	アップデートする既存のプロジェクトファイルおよびディレクトリを指定します。
[-filesets]	ファイルセット名を指定します。
[-force]	プロジェクトのインポートされたファイルを、読み取り専用であっても可能な限り上書きします。
[-report_only]	実際のファイル アップデートは実行せず、実行されるはずのアップデートをレポートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

指定したリモート ファイルの内容で指定のファイルをアップデートします。このコマンドは、元のリモート ファイルの内容でローカル ファイルをアップデートしたり、別のリモート ファイルの内容に置き換えるために使用します。

このコマンドを実行すると、アップデートされたファイルのリストが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-from_files <args> (オプション): -to_files オプションで指定したファイルをアップデートする際に使用するファイルまたはディレクトリの順序付きリストを指定します。

-norecurse (オプション): 下位ディレクトリではコマンドを実行しません。

`-to_files <args>` (オプション): `-from_files` オプションで指定したファイルでアップデートするファイルのパスと名前を指定します。

`-filesets <args>` (オプション): 指定のファイルセットのファイルを `-from_files` オプションで指定したファイルで上書きします。

`-force` (オプション): 書き込みが制限されている場合でも、指定のファイルを強制的に上書きします。

`-report_only` (オプション): アップデートされるファイルに関するレポートを生成しますが、実際のファイルのアップデートは実行しません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、さまざまなプロジェクト ソース IP コア ファイルを `-from_files` オプションで指定したファイルでアップデートした場合の結果レポートを生成し、実際のアップデートは実行しません。

```
update_files -from_files C:/Data/IP/*.xci \
  -to_file [get_files *.xci] -report_only
```

注記: 上書きされるローカル ファイルの方が新しくても、警告メッセージは表示されません。

関連項目

- [reimport_files](#)

update_hw_firmware

SmartLynq ファームウェア イメージをアップデートします。

構文

```
update_hw_firmware [-file_path <arg>] [-config_path <arg>] [-skip_update]
                    [-reset] [-format] [-flash] [-quiet] [-verbose] [<hw_server>]
```

使用法

名前	説明
[-file_path]	BOOT.BIN ファイルへのオプションのパスを指定します。デフォルトの BOOT.BIN を使用します。
[-config_path]	config.ini ファイルへのオプションのパスを指定します。config.ini はアップデートされません。
[-skip_update]	BOOT.BIN を SmartLynq に書き込みません。
[-reset]	ほかの操作の後に SmartLynq ケーブルをリセットし、アップデートを完了して、hw_server への接続を解除します。
[-format]	ほかの操作の前に SmartLynq ケーブル EMMC をフォーマットします。SmartLynq ケーブル上のファイルはすべて失われます。
[-flash]	SmartLynq ケーブル QSPI に書き込みます。これにより、プライマリ FSBL およびセーフ モード イメージがアップデートされます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_server>]	ハードウェア サーバーを指定します。デフォルトは、現在のハードウェア サーバーです。

カテゴリ

[Hardware \(ハードウェア\)](#)

update_hw_gpio

SmartLynq GPIO PMOD ピンの値をアップデートします。

構文

```
update_hw_gpio [-quiet] [-verbose] [<output_enable_mask>]
               [<output_pin_values>] [<hw_server>]
```

戻り値

すべての GPIO PMOD ピン値

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<output_enable_mask>]	GPIO の出力ピンをイネーブルにする 8 ビットの 16 進数マスクを指定します。デフォルトでは、すべての出力ピンがディスエーブルになります。
[<output_pin_values>]	出力ピンの 8 ビットの 16 進数値を指定します。デフォルトでは、すべての出力ピンが Low で駆動されます。
[<hw_server>]	ハードウェア サーバーを指定します。デフォルトは、現在のハードウェア サーバーです。

カテゴリ

[Hardware \(ハードウェア\)](#)

update_ip_catalog

IP カタログをアップデートします。このコマンドを実行する前に、オプションで「set_property ip_repo_paths <repo_path_list> [current_fileset]」を使用してリポジトリを設定できます。

構文

```
update_ip_catalog [-rebuild] [-add_ip <arg>] [-delete_ip <arg>]
                  [-delete_mult_ip <args>] [-disable_ip <arg>] [-enable_ip <arg>]
                  [-add_interface <arg>] [-create_index] [-repo_path <arg>]
                  [-update_module_ref] [-quiet] [-verbose]
```

戻り値

コマンドが正常に実行された場合は True

使用法

名前	説明
[-rebuild]	指定したリポジトリのインデックス ファイルを再構築します。リポジトリが指定されていない場合は、すべてのリポジトリを再構築します。
[-add_ip]	指定した IP を指定したリポジトリに追加します。IP の component.xml へのパスまたは IP を含む ZIP ファイルへのパスを指定します。
[-delete_ip]	指定した IP を指定したリポジトリから削除します。IP の component.xml へのパスまたは VLNV を指定します。
[-delete_mult_ip]	指定した IP を指定したリポジトリから削除します。IP のリストを component.xml ファイルへのパスまたは VLNV で指定します。
[-disable_ip]	指定したリポジトリの指定した IP をディスエーブルにします。IP の component.xml へのパスまたは VLNV を指定します。
[-enable_ip]	指定したリポジトリでディスエーブルになっている指定した IP をイネーブルにします。IP の component.xml へのパスまたは VLNV を指定します。
[-add_interface]	指定したインターフェイスを指定したリポジトリに追加します。インターフェイスの XML ファイルへのパスを指定します。
[-create_index]	指定のリポジトリのデータをディスクにキャッシュし、読み込み時間を短縮します。
[-repo_path]	-rebuild、-add_ip、-delete_ip、-delete_mult_ip、-disable_ip、または -create_index オプションと共に使用し、操作を実行するリポジトリへのパスを指定します。
[-update_module_ref]	モジュール参照をそのソース (HDL ファイルなど) からアップデートします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

IPFlow (IP フロー)

説明

現在のデザインに関連付けられている IP カタログをアップデートします。

ザイリンクス® IP カタログ (リポジトリ) は、使用している Vivado Design Suite のバージョンのインストール ディレクトリにあります。また、下の例に示すように、`set_property` コマンドを使用してソース ファイルセットの `IP_REPO_PATHS` プロパティをカスタム IP の場所に設定して、カスタム IP をリポジトリに追加できます。

`update_ip_catalog` コマンドを使用すると、カタログに含まれる IP コアを個別に追加、削除、ディスエーブル、またはイネーブルにできます。個別のコアは、`component.xml` ファイルへのパスまたは IP の VLNV プロパティで参照できます。



ヒント: VLNV プロパティは `Vendor:Library:Name:Version` 形式の文字列で、カタログ内の IP を識別します。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-rebuild` (オプション): IP カタログ インデックス全体を再構築するか、`-repo_path` で指定された IP リポジトリのインデックスのみを再構築します。

`-add_ip <arg>` (オプション): IP コアを指定した IP リポジトリに追加します。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。IP は、IP の `component.xml` へのパスまたは IP を含む ZIP ファイルへのパスで指定します。

`-delete_ip <arg>` (オプション): IP コアを指定した IP リポジトリから削除します。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。IP は、IP の `component.xml` へのパスまたは IP の VLNV プロパティで指定します。

`-delete_mult_ip <arg>` (オプション): 指定した IP コアを IP リポジトリから削除します。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。IP は、`component.xml` ファイルへのパスまたは IP の VLNV プロパティで指定します。

`-disable_ip <arg>` (オプション): 指定の IP リポジトリの IP コアをディスエーブルにします。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。IP は、IP の `component.xml` へのパスまたは IP の VLNV プロパティで指定します。

`-enable_ip <arg>` (オプション): 指定のリポジトリでディスエーブルになっている IP コアをイネーブルにします。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。IP は、IP の `component.xml` へのパスまたは IP の VLNV プロパティで指定します。

`-add_interface <arg>` (オプション): IP リポジトリに追加するユーザー定義 AXI インターフェイスの XML ファイルへのパスを指定します。

`-create_index` (オプション): 指定のリポジトリのデータをディスクにキャッシュし、読み込み時間を短縮します。このオプションを使用する場合は、`-repo_path` オプションも指定する必要があります。

`-repo_path <arg>` (オプション): `-rebuild`、`-add_ip`、`-delete_ip`、`-delete_mult_ip`、または `-create_index` オプションと共に使用し、操作を実行するリポジトリのディレクトリ名を指定します。



重要: IP リポジトリは、`set_property` コマンドを使用して `IP_REPO_PATH` を設定することにより、現在のソース ファイルセットに追加しておく必要があります。例を参照してください。

`-update_module_ref` (オプション): ソース ファイルからモジュール定義を読み込み直すことにより、RTL ソース ファイルからモジュール定義を参照するブロック デザイン セルを更新します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在のソース ファイルセットの `IP_REPO_PATHS` プロパティを設定し、IP リポジトリを追加して、IP カタログ全体のインデックスを再構築しています。

```
set_property IP_REPO_PATHS C:/Data/IP_LIB [current_fileset]
update_ip_catalog -rebuild
```

次の例では、VLNV プロパティで指定した IP コアを指定した IP リポジトリからディスエーブルにしています。

```
update_ip_catalog -disable_ip {myCo.com:ip:custom_decoder:1.0} \
-repo_path C:/Data/ip
```

次の例では、`component.xml` ファイルへのパスで指定した IP コアを IP リポジトリからディスエーブルにしています。

```
update_ip_catalog -disable_ip C:/Data/ip/custom_encoder_1/component.xml \
-repo_path C:/Xilinx/Vivado/data/ip
```

関連項目

- [create_ip](#)
- [import_ip](#)
- [generate_target](#)
- [update_module_reference](#)
- [validate_ip](#)

update_macro

マクロをアップデートします。

構文

```
update_macro [-absolute_grid] [-quiet] [-verbose] <macro> <rlocs>
```

使用法

名前	説明
<code>[-absolute_grid]</code>	相対ロケーションに絶対座標を使用します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><macro></code>	アップデートするマクロを指定します。
<code><rlocs></code>	インスタンスとサイトの名前を交互に指定します。

カテゴリ

XDC

説明

作成済みのマクロに最下位セルおよび相対配置を指定します。

マクロは、プリミティブまたは最下位ロジック セル、その接続、配置座標での位置で構成されます。指定の相対ロケーション (`<rlocs>`) は、相対座標または絶対座標 (RPM_GRID) に基づいて指定できます。絶対および相対配置座標の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。

1 つのセルは、1 つのマクロにのみ割り当てることができます。最下位セルを複数のマクロに割り当てようとすると、エラーが返されます。プリミティブでないセルをマクロに割り当てようとすると、エラーが返されます。

既存のマクロの内容を変更するには、`delete_macro` コマンドを使用してマクロを削除し、`create_macro` コマンドで再作成し、新しい内容でアップデートする必要があります。既存のマクロを単に上書きしたり変更したりすることはできません。

引数

`-absolute_grid` (オプション): `<rlocs>` が絶対 RPM_GRID で指定されていることを示します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<macro> (必須): アップデートするマクロの名前を指定します。

<rlocs> (必須): マクロに含める最下位セルとその相対ロケーション (RLOC) を指定します。-absolute_grid オプションを指定している場合、ロケーションは相対ロケーションではなく実際のデバイス座標に基づきます。セルおよび RLOC は名前と値のペアで指定し、複数の最下位セルと RLOC を 1 つのマクロに割り当てるには次のように指定します。

```
{cell0 XmYn cell1 XmYn ... cellN XmYn}
```

説明:

- m: その前にリストされているセルの相対または絶対 X 座標を示す整数。
- n: その前にリストされているセルの相対または絶対 Y 座標を示す整数。
- -absolute_grid オプションを指定していない場合、セルの座標はお互いのセルに相対します。
- 相対座標は、それぞれに相対して配置されたスライスの仮想配列に基づきます。
- 絶対座標は、ターゲット デバイスの実際のスライスの RPM_X および RPM_Y プロパティにより決定されます。これらのプロパティは、選択したサイトに対して report_property コマンドを使用して確認できます。

例

次の例では、usbMacro0 という名前のマクロを作成し、現在のインスタンスを usbEngine0/u0 モジュールに設定して、3 つのセルをマクロに割り当て、相対配置で 2 つのセルを同じスライスに、3 つ目のセルを縦に隣接するスライスに配置しています。

```
create_macro usbMacro0
current_instance usbEngine0/u0
update_macro usbMacro0 {rx_active_reg X0Y0 \
    rx_err_reg X0Y0 rx_valid_reg X0Y1}
```

次の例では、usbMacro1 という名前のマクロを作成し、最下位セルに階層パスを使用して 3 つのセルをマクロに割り当て、絶対座標を使用してセルをマクロ内に配置しています。

```
create_macro usbMacro1
set Site1 [get_sites SLICE_X8Y77]
set Site2 [get_sites SLICE_X9Y77]
set Site3 [get_sites SLICE_X8Y78]
set RPM1 X[get_property RPM_X $Site1]Y[get_property RPM_Y $Site1]
set RPM2 X[get_property RPM_X $Site2]Y[get_property RPM_Y $Site2]
set RPM3 X[get_property RPM_X $Site3]Y[get_property RPM_Y $Site3]
update_macro usbMacro1 -absolute_grid "usbEngine1/u0/rx_active_reg $RPM1 \
usbEngine1/u0/rx_err_reg $RPM2 usbEngine1/u0/rx_valid_reg $RPM3"
```

注記: 上記の例では、Tcl 変数を使用してサイトをキャプチャし、それらのサイトの RPM_X および RPM_Y プロパティの値を update_macro コマンドで使用しています。update_macro コマンドで波かっこ ({}) の代わりにダブルクォーテーション ("") を使用しており、これによって Tcl シェルでコマンドの変数置換を実行できます。変数および変数置換の詳細は、『Vivado Design Suite ユーザー ガイド: Tcl スクリプト機能の使用』 (UG894) を参照してください。

次の例では、usbMacro1 マクロのプロパティをレポートし、セルに割り当てられている絶対座標を確認しています。

```
report_property -all [get_macros usbMacro1]
```

関連項目

- [create_macro](#)
- [delete_macros](#)
- [get_macros](#)
- [get_property](#)
- [get_sites](#)
- [place_design](#)
- [report_property](#)

update_module_reference

モジュール参照定義およびインスタンスを更新します。

構文

```
update_module_reference [-quiet] [-verbose] [<ips>...]
```

戻り値

正常に完了したか、エラーが発生したかを示す戻りコード。

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<ips>]	アップグレードするモジュール参照を指定します。デザイン内の IP インスタンスを指定します。

カテゴリ

[IPFlow \(IP フロー\)](#)

説明

ソース ファイルからモジュール定義を読み込み直すことにより、RTL ソース ファイルからモジュール定義を参照するブロック デザインセルを更新します。

注記: このコマンドでは、ソース ファイルは再読み込みされません。ソース ファイルが変更されている場合は、個別にアップデートする必要があります。

このコマンドを実行すると、アップデート プロセスとデザインの変更に関する警告が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<ips> (オプション): 現在のデザインでアップデートするモジュール参照を指定します。モジュールは、名前で指定するか、または get_ips コマンドを使用してオブジェクトとして指定します。

例

次の例では、現在のデザイン指定したモジュール参照をアップデートしています。

```
update_module_reference {rtlRam_0 uart_0}
```

関連項目

- [create_bd_cell](#)
- [create_bd_design](#)

update_noc_qos

NoC ソリューションをアップデートします。

構文

```
update_noc_qos [-force] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-force]</code>	既存のソリューションが無効であっても強制的にアップデートします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Tools \(ツール\)](#)

update_sw_parameters

DCP に生成されたソフトウェア パラメーター情報ファイルを追加します。

構文

```
update_sw_parameters [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

デザイン チェックポイント (DCP) を最新のハードウェア定義でアップデートします。このハードウェア定義には、アップデートされたソフトウェア パラメーター情報が含まれます。

IP のソフトウェア パラメーターを変更した場合、run はアップデート必要にはなりませんが、生成済みのデザイン チェックポイントは変更したパラメーター情報でアップデートする必要があります。この情報は hwdef ファイルに含まれます。update_sw_parameters コマンドを実行すると、hwdef ファイルが生成済みのデザイン チェックポイントに追加されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

例

次の例は、CIPS IP のソフトウェア パラメーターを変更し、hw_handoff ターゲットを生成した後、変更した値で DCP をアップデートします。

```
set_property -dict [list CONFIG.PS_I2C0_PERIPHERAL_ENABLE {1}]
[get_bd_cells versal_cips_0]
generate_target hw_handoff [get_files top.bd]
update_sw_parameters
```

関連項目

- [refresh_design](#)
- [close_design](#)

update_timing

タイミングをアップデートします。

構文

```
update_timing [-full] [-skip_delay_calc] [-quiet] [-verbose]
```

使用法

名前	説明
<code>[-full]</code>	インクリメンタルなタイミング アップデートではなく、完全なタイミング アップデートを実行します。
<code>[-skip_delay_calc]</code>	遅延の算出をスキップします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

Timing (タイミング)

説明

現在のデザインのタイミングをアップデートします。

タイミング データをアップデートすると、タイミング エンジンが最後に実行されてからデザインに追加されたタイミング制約が反映されます。このコマンドを使用すると、完全なタイミング解析を実行せずに、タイミング データベースのインメモリ ビューをアップデートできます。

タイミングは、`report_timing` コマンドなど、タイミングを変更するコマンドまたはタイミング情報のアップデートが必要なコマンドにより自動的にアップデートされます。`update_timing` コマンドは、タイミング エンジンに最新の制約が適用されていることを確実にするため、タイミングを手動でアップデートする際に使用できます。

`update_timing` コマンドは、デフォルトではインクリメンタル解析を使用し、更新が必要な情報のみをアップデートして解析時間を短縮します。デザインのタイミング データを包括的に確認するため完全なアップデートを指定することもできますが、タイミング解析の実行時間が長くなるのを回避するため、`-full` オプションは必要な場合にのみ使用してください。

引数

`-full` (オプション): デフォルトのタイミング データのインクリメンタル アップデートではなく、完全なタイミング解析を実行します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、インメモリ タイミング データの完全なアップデートを実行しています。

```
update_timing -full
```

関連項目

- [report_timing](#)
- [report_timing_summary](#)
- [reset_timing](#)

upgrade_bd_cells

コンフィギャラブル IP インテグレーター セルを最新バージョンにアップグレードします。

構文

```
upgrade_bd_cells [-latest <arg>] [-quiet] [-verbose] <objects>...
```

戻り値

アップグレードされた IP インテグレーター セルの名前、エラーが発生した場合は ""。

使用法

名前	説明
<code>[-latest]</code>	IP インテグレーター ブロックを最新バージョンにアップグレードします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><objects></code>	アップグレードする IP インテグレーター セルを指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

IP インテグレーター セルを IP インテグレーター カタログで使用可能な最新バージョンにアップグレードします。

このコマンドを使用すると、IP インテグレーター サブシステム デザインで IP コアを以前のリリースから最新のカタログの IP コアにアップグレードできます。

このコマンドが正常に実行された場合はアップグレードされた IP インテグレーター セルのリストが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-latest` (オプション): 指定したセルを IP カタログで使用可能な最新バージョンにアップグレードします。これがデフォルトの動作であり、指定する必要はありません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<objects>: 最新バージョンにアップグレードする IP インテグレーター セルを指定します。オブジェクトは、`get_bd_cells` コマンドを使用して指定する必要があります。名前で指定することはできません。

例

次の例では、指定したセルを IP インテグレーター カタログで使用可能な最新バージョンにアップグレードしています。

```
upgrade_bd_cells [get_bd_cells {vidOut1 cmpy_1 newMod1}]
INFO: [BD 41-1162] The cell '/vidOut1' is already at its latest version.
INFO: [BD 41-1162] The cell '/cmpy_1' is already at its latest version.
WARNING: [BD 41-1082] Hierarchy block (/newMod1) cannot be directly
upgraded.
Please dive into the hierarchy and select individual cells to upgrade.
```

注記: ユーザー定義の階層モジュールに対しては、警告メッセージが返されます。

関連項目

- [get_bd_cells](#)

upgrade_ip

コンフィギュラブル IP を最新バージョンにアップグレードします。

構文

```
upgrade_ip [-srcset <arg>] [-vlnv <arg>] [-log <arg>] [-quiet] [-verbose]
           <objects>...
```

戻り値

正常に完了したか、エラーが発生したかを示す戻りコード。

使用法

名前	説明
[-srcset]	アップグレードする IP を含むソース ファイル セットを指定します。デフォルトは、現在のソース ファイルセットです。
[-vlnv]	アップグレードする IP の IP カタログの VLNv 文字列を指定します。VLNV 文字列は IP コアの IPDEF プロパティにマップされています。厳密な比較が実行され、指定した IP がカタログに存在しない場合はエラーが返されます。デフォルトは現在の IP の最新バージョンです。 <vendor>:<library>:<name>:<version> という形式の文字列で指定します。
[-log]	IP のアップグレード レポートを追加するログ ファイルを指定します。既存の書き込み可能なファイルまたは書き込み可能なディレクトリの存在しないファイル指定します。デフォルトは空の文字列であり、ログ ファイルは記述されません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<objects>	アップグレードする IP を指定します。デザイン内の IP インスタンスを get_ips <instance name> または get_bd_cells <cell name> コマンドを使用して指定します。

カテゴリ

IPFlow (IP フロー)

説明

指定した IP コアを IP カタログの最新バージョンにアップグレードします。

アップグレードできるのは、アップグレードが明示的にサポートされている IP のみです。ipdef オブジェクトの UPGRADE_VERSIONS プロパティを確認すると、IP コアのアップグレード バージョンがあるかどうかわかります。



ヒント: upgrade_ip コマンドでは、ブロック デザインのセル IP をブロック デザイン セル オブジェクト (bd_cell) として指定できます。このコマンドはブロック デザイン内のブロック デザインのセル オブジェクトをアップグレードし、ダイアグラムを Vivado IP インテグレーターで開く必要はありません。

引数

`-srcset <arg>` (オプション): IP ファイルをアップグレードするソース ファイルセットを指定します。指定しない場合、デフォルトのソース ファイルセットは `sources_1` です。

`-vlnv <arg>` (オプション): IP カタログからアップグレードする IP の Vendor:Library:Name:Version 属性を指定します。Vlnv 属性は、IP カタログ内のオブジェクトを識別します。

`-log <arg>` (オプション): IP アップグレード情報を追加するファイルの名前を指定します。デフォルトでは、ログ ファイルは生成されません。

注記: パスをファイル名の一部として指定しない場合は、現在のプロジェクト ディレクトリにログ ファイルが作成されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<objects>` (必須): アップグレードするブロック デザイン セルの IP コアを指定します。IP は、`get_ips` コマンドを使用してオブジェクトとして指定する必要があります。



重要: `get_ips -all` は、再帰問題が発生することがあるので、使用しないでください。

例

次の例では、現在のプロジェクトに含まれるすべての IP を最新バージョンにアップグレードしています。

```
upgrade_ip [get_ips]
```

関連項目

- [create_ip](#)
- [get_bd_cells](#)
- [get_ips](#)
- [import_ip](#)
- [open_bd_design](#)

upload_hw_ila_data

キャプチャを停止し、キャプチャされたハードウェア ILA データをアップロードします。

構文

```
upload_hw_ila_data [-quiet] [-verbose] [<hw_ilas>...]
```

戻り値

ハードウェア ILA データ オブジェクト

使用法

名前	説明
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_ilas>]	ハードウェア ILA オブジェクトを指定します。デフォルトは、現在のハードウェア ILA です。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

ザイリンクス FPGA ハードウェア デバイス上の指定した ILA デバッグ コアのメモリ バッファからキャプチャされたデータをアップロードし、Vivado ロジック解析のハードウェア ILA データ (hw_ila_data) オブジェクトに移動します。

run_hw_ila コマンドを使用してキャプチャ プロセスを開始した後、いつでも ILA デバッグ コアからキャプチャされたデータをアップロードできますが、Tcl スクリプトで wait_on_hw_ila コマンドを使用して、データ サンプル バッファにキャプチャされたデータが完全に挿入されるまで待つことをお勧めします。その前に upload_hw_ila_data コマンドを使用すると、次のようなメッセージが表示されることがあります。

```
INFO: [Labtools 27-1965] The ILA core 'hw_ila_1' trigger was stopped by
user \
at 2014-Mar-06 08:59:30
INFO: [Labtools 27-2212] The ILA core 'hw_ila_1' captured '6' windows with
\
'64' samples each, and a last partial window with '0' samples.
```

アップロード プロセスでは、FPGA デバイス (hw_device) 上の ILA デバッグ コア (hw_ila) からキャプチャされたデータを移動する際に hw_ila_data オブジェクトが作成されます。hw_ila_data オブジェクトの名前は、データのアップロード元である hw_ila コアに基づいて付けられます。



ヒント: 各 hw_ila オブジェクトには、1 つの hw_ila_data オブジェクトのみが関連付けられています。特定の hw_ila コアに対して upload_hw_ila_data コマンドを実行するたびに、既存の hw_ila_data オブジェクトは上書きされます。

hw_ila_data オブジェクトは、display_hw_ila_data コマンドを使用して Vivado ロジック解析機能の波形ウィンドウに表示でき、write_hw_ila_data コマンドを使用してディスクに保存できます。

このコマンドを実行すると、hw_ila_data オブジェクトが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<hw_ilas> (オプション): データのアップロード元である hw_ila オブジェクトを 1 つ以上指定します。hw_ila は、get_hw_ilas または current_hw_ila コマンドを使用してオブジェクトとして指定するか、名前で指定できます。hw_ila オブジェクトを指定しない場合、現在のハードウェア current_hw_ila からデータがアップロードされます。

例

次の例では、ターゲット hw_device 上の現在のハードウェア ILA デバッグ コアをトリガー待機状態にし、トリガー イベントまたはキャプチャ条件が発生したときにプローブのサンプル データをキャプチャしています。Tcl スクリプトはサンプル データがキャプチャされる間一時停止されます。その後、hw_device 上の hw_ila からデータを hw_ila_data オブジェクトにアップロードしています。

```
run_hw_ila -trigger_now [current_hw_ila]
wait_on_hw_ila [current_hw_ila]
upload_hw_ila_data [current_hw_ila]
```

関連項目

- [current_hw_device](#)
- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [display_hw_ila_data](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [run_hw_ila](#)
- [wait_on_hw_ila](#)
- [write_hw_ila_data](#)

validate_bd_design

指定のデザインまたは指定のセルに対してパラメーターの伝搬を実行します。

構文

```
validate_bd_design [-force] [-design <arg>] [-include_pfm] [-quiet]  
                  [-verbose]
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
<code>[-force]</code>	デザインの検証を強制的に再実行します。
<code>[-design]</code>	デザイン名を指定します。指定しない場合、パラメーターの伝搬は現在のデザインに対して実行されます。
<code>[-include_pfm]</code>	デザインの PFM 属性の検証を含めます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

IP インテグレーター サブシステム デザイン、IP セル、または階層モジュールを検証します。

引数

`-force` (オプション): ブロック デザインの検証を強制的に実行します。

`-design <arg>` (オプション): 検証する IP インテグレーター サブシステム デザインを指定します。指定しない場合、現在の IP インテグレーター サブシステム デザインが検証されます。

`-include_pfm` (オプション): SDx プラットフォームのハードウェア部分として作成されたブロック デザインに必要な DSA および PFM 属性を検証します。プラットフォーム プロパティが適切に定義されていない場合は、エラーが返されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、現在の IP インテグレーター サブシステム デザインを検証し、必要に応じて検証を強制的に実行します。

```
validate_bd_design -force
```

関連項目

- [create_bd_design](#)
- [open_bd_design](#)
- [save_bd_design](#)

validate_hw_platform

指定したハードウェア プラットフォームを検証します。

構文

```
validate_hw_platform [-verbose] [-quiet] [<file>]
```

戻り値

シェル ファイル名

使用法

名前	説明
<code>[-verbose]</code>	詳細情報を表示します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[<file>]</code>	ザイリンクス シェル アーカイブ ファイルへのパスを指定します。

カテゴリ

[Platform \(プラットフォーム\)](#)、[Vitis](#)

説明

ザイリンクス サポート アーカイブ (XSA) ファイルを検証し、プラットフォームに必要な内容が含まれているかどうかを確認します。

このコマンドを実行すると、XSA の検証に関する情報が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-verbose` (オプション): XSA および検証プロセスに関する詳細情報を返します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`<file>` (オプション): XSA ファイルのパスとファイル名を指定します。

注記: パスをファイル名の一部として指定しない場合は、まず現在の作業ディレクトリ、次にツールを起動したディレクトリで指定ファイルが検索されます。

例

次の例は、指定した XSA を検証し、プラットフォームに関する詳細情報を表示します。

```
validate_hw_platform C:/Data/zc706.xsa -verbose
```

関連項目

- [open_hw_platform](#)
- [write_hw_platform](#)

validate_ip

IP が有効であることを確認します。

構文

```
validate_ip [-save_ip] [-quiet] [-verbose] [<ips>]
```

使用法

名前	説明
<code>[-save_ip]</code>	IP ファイルをディスクに書き込みます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<ips>]</code>	確認する IP を指定します。

カテゴリ

IPFlow (IP フロー)

説明

IP に対して DRC チェックを実行し、正しく構築されているかどうかを確認します。すべての IP が正しく構築されている場合は 1、問題がある場合は 0 が返されます。

引数

`-save_ip` (オプション): 検証した後に既存の IP ファイルをアップデートします。新しいファイルは作成されず、コアの XCI および BOM ファイルがアップデートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンドラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<ips>` (オプション): 検証する IP コアを指定します。

例

次の例では、現在のプロジェクトに含まれる IP を検証し、IP のファイルをアップデートしています。

```
validate_ip -save_ip [get_ips]
```

関連項目

- [create_ip](#)
- [generate_target](#)
- [upgrade_ip](#)
- [update_ip_catalog](#)
- [import_ip](#)

verify_hw_devices

ハードウェア デバイスを検証します。

構文

```
verify_hw_devices [-key <arg>] [-user_efuse <arg>] [-control_efuse <arg>]  
                  [-security_efuse <arg>] [-verbose] [-quiet] [<hw_device>...]
```

戻り値

ハードウェア デバイス

使用法

名前	説明
[-key]	暗号キーを検証します。有効な値は efuse です。
[-user_efuse]	検証用の 16 進ユーザー eFUSE 値を指定します。
[-control_efuse]	検証用の 16 進制御 eFUSE 値を指定します。
[-security_efuse]	検証用の 16 進セキュリティ eFUSE 値を指定します。
[-verbose]	検証中の eFUSE レジスタ値を表示します。
[-quiet]	コマンド エラーを表示しません。
[<hw_device>]	ハードウェア デバイスを指定します。デフォルトは現在のハードウェア デバイスです。

カテゴリ

Hardware (ハードウェア)

説明

eFUSE 暗号化されたデバイスで、指定のハードウェア デバイス (hw_device) の PROGRAM.FILE プロパティに割り当てられているビットストリームを、program_hw_devices コマンドを使用してデバイスにプログラムされたビットストリームと比較します。

verify_hw_devices コマンドは、ビットストリームと hw_device に関連付けられているマスク ファイルを使用して、マスク ファイルでマークされているビットのみを比較します。マスク ファイルは、write_bitstream コマンドを使用してビットストリーム ファイルと共に作成でき、create_hw_bitstream コマンドを使用して hw_device に関連付けられます。



重要: 暗号化されたビットストリームでプログラムされたデバイスは、-key オプションを使用してキーがプログラムされていることを検証する以外は、検証できません。

verify_hw_devices コマンドを実行すると、正常に実行された場合はリードバック データがプログラムされたビットストリームと一致することがレポートされ、正常に実行されなかった場合はエラーが返されます。

引数

-key efuse (オプション): 暗号キーが指定した hw_device 上の eFUSE レジスタにプログラムされているかを検証します。

-user_efuse <arg> (オプション): hw_device 上の FUSE_USER レジスタにプログラムされている HEX 値を検証します。

-control_efuse <arg> (オプション): hw_device 上の FUSE_CNTL レジスタにプログラムされている HEX 値を検証します。

-security_efuse <arg> (オプション): hw_device 上の FUSE_SEC レジスタにプログラムされている HEX 値を検証します。

-verbose (オプション): デバイスを検証するときに eFUSE レジスタ値をレポートします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

<hw_device> (オプション): 検証する hw_device オブジェクトを 1 つ以上指定します。hw_device は、get_hw_devices コマンドを使用してオブジェクトとして指定する必要があります。デバイスを指定しない場合、現在のハードウェア デバイス (current_hw_device) が検証されます。

例

次の例では、現在のハードウェア デバイスのビットストリームを検証しています。

```
verify_hw_devices [current_hw_device]
```

関連項目

- [connect_hw_server](#)
- [create_hw_device](#)
- [create_hw_target](#)
- [current_hw_device](#)
- [current_hw_target](#)
- [get_hw_devices](#)
- [get_hw_targets](#)
- [open_hw_target](#)
- [program_hw_devices](#)
- [write_bitstream](#)
- [write_hw_svf](#)

version

Vivado のバージョンおよびバージョンの日付を返します。

構文

```
version [-short] [-quiet] [-verbose]
```

戻り値

Vivado のバージョン

使用法

名前	説明
<code>[-short]</code>	バージョン番号のみを返します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。

カテゴリ

[Report \(レポート\)](#)

説明

ザイリンクス® ツールのバージョン番号を返します。バージョン番号、ビルドの番号と日付、および著作権情報が含まれます。

引数

`-short` (オプション): ツールのバージョン場号のみを返します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

例

次の例では、ツールのバージョン番号のみが返されます。

```
version -short
```

wait_on_hw_ila

すべてのハードウェア ILA データがキャプチャされるまで待機します。

構文

```
wait_on_hw_ila [-timeout <arg>] [-quiet] [-verbose] [<hw_ilas>...]
```

使用法

名前	説明
[-timeout]	タイムアウト時間を分で指定します。10 進数を指定します。デフォルトではタイムアウトはありません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<hw_ilas>]	ハードウェア ILA オブジェクトを指定します。デフォルトは、現在のハードウェア ILA です。

カテゴリ

Hardware (ハードウェア)

説明

ILA デバッグ コアのメモリがキャプチャされたデータ サンプルでフルになるまで、Tcl スクリプトまたは Tcl コマンドの処理を一時停止します。

このコマンドは、run_hw_ila コマンドを実行した後に、データ バッファがフルになるまで Tcl の処理を一時停止するために使用します。wait_on_hw_ila コマンドから戻ると、Tcl コマンドまたはスクリプトの処理が再開されます。

ILA デバッグ コア メモリにサンプル データが挿入され、Tcl の処理が再開したら、キャプチャされたデータ サンプルを ILA デバッグ コア データ オブジェクト (hw_ila_data) にアップロードできます。これには、upload_ila_data コマンドを使用します。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

-timeout <arg> (オプション): ILA デバッグ コア上のすべてのデータをキャプチャするのに待機する時間を指定します。このオプションで指定したタイムアウトに達すると、このコマンドは終了します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_ila> (オプション): 待機するハードウェア ILA (hw_ila) オブジェクトを 1 つ以上指定します。hw_ila は、`get_hw_ilas` または `current_hw_ila` コマンドを使用してオブジェクトとして指定する必要があります。ハードウェア ILA を指定しない場合、現在のハードウェア ILA (`current_hw_ila`) が指定されます。

例

次の例では、現在のハードウェア ILA デバッグ コア上のすべてのデータがキャプチャされるまで待機しています。

```
run_hw_ila hw_ila_1 -trigger_now 1
INFO: [Labtools 27-1964] The ILA core 'hw_ila_1' trigger was armed
      at 2014-Mar-02 13:20:30
wait_on_hw_ila hw_ila_1
display_hw_ila_data [upload_hw_ila_data hw_ila_1]
INFO: [Labtools 27-1966] The ILA core 'hw_ila_1' triggered
      at 2014-Mar-02 13:20:31
```

関連項目

- [current_hw_device](#)
- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [display_hw_ila_data](#)
- [get_hw_devices](#)
- [get_hw_ilas](#)
- [run_hw_ila](#)
- [upload_hw_ila_data](#)

wait_on_hw_sio_scan

ハードウェア SIO スキャンが完了するまで待機します。

構文

```
wait_on_hw_sio_scan [-timeout <arg>] [-quiet] [-verbose] <hw_sio_scans>...
```

使用法

名前	説明
<code>[-timeout]</code>	タイムアウト時間を分で指定します。10 進数を指定します。デフォルトではタイムアウトはありません。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_scans></code>	ハードウェア SIO スキャン オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

指定したシリアル I/O 解析スキャンが完了するまで、Tcl スクリプトまたは Tcl コマンドの処理を一時停止します。

このコマンドは、`run_hw_sio_scan` コマンドを実行した後に、スキャンが完了するまで Tcl の処理を一時停止するために使用します。`wait_on_sio_scan` コマンドから戻ると、Tcl コマンドまたはスクリプトの処理が再開されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-timeout <arg>` (オプション): シリアル I/O 解析スキャンが完了するまで待機する時間を指定します。このオプションで指定したタイムアウトに達すると、このコマンドは終了します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<hw_sio_scans>` (必須): 待機するハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトを 1 つ以上指定します。`hw_sio_scan` は、`create_hw_sio_scan` または `get_hw_sio_scans` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、シリアル I/O 解析スキャンが完了するまで待機しています。

```
run_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_0}] 0]  
wait_on_hw_sio_scan [lindex [get_hw_sio_scans {SCAN_0}] 0]
```

関連項目

- [create_hw_sio_scan](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [remove_hw_sio_scan](#)
- [run_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [write_hw_sio_scan](#)

wait_on_hw_sio_sweep

ハードウェア SIO スイープが完了するまで待機します。

構文

```
wait_on_hw_sio_sweep [-timeout <arg>] [-quiet] [-verbose]
                     <hw_sio_sweeps>...
```

使用法

名前	説明
<code>[-timeout]</code>	タイムアウト時間を分で指定します。10 進数を指定します。デフォルトではタイムアウトはありません。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><hw_sio_sweeps></code>	ハードウェア SIO スイープ オブジェクトを指定します。

カテゴリ

Hardware (ハードウェア)

説明

シリアル I/O 解析スイープ スキャンが完了するまで、Tcl スクリプトまたは Tcl コマンドの処理を一時停止します。

このコマンドは、`run_hw_sio_sweep` コマンドを実行した後に、スイープ スキャンが完了するまで Tcl の処理を一時停止するために使用します。`wait_on_sio_sweep` コマンドから戻ると、Tcl コマンドまたはスクリプトの処理が再開されます。

このコマンドが正常に実行された場合は何も返されず、正常に実行されない場合はエラーが返されます。

引数

`-timeout <arg>` (オプション): シリアル I/O 解析スイープ スキャンが完了するまで待機する時間を指定します。このオプションで指定したタイムアウトに達すると、このコマンドは終了します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<hw_sio_sweeps> (必須): 待機するハードウェア SIO スイープ (hw_sio_sweep) オブジェクトを 1 つ以上指定します。hw_sio_sweep は、create_hw_sio_sweep または get_hw_sio_sweeps コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、SIO スイープ スキャンを開始し、スイープが完了するまで待機しています。

```
run_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]  
wait_on_hw_sio_sweep [lindex [get_hw_sio_sweeps {SWEEP_0}] 0]
```

関連項目

- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [write_hw_sio_sweep](#)

wait_on_run

指定した run が終了するまで Tcl コマンドの実行を停止します。

構文

```
wait_on_run [-timeout <arg>] [-quiet] [-verbose] <run>
```

使用法

名前	説明
[-timeout]	run が完了するまで待機する最大時間 (分) を指定します。デフォルトは -1 です。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<run>	待機する run を指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

指定した run が正常に完了するかエラーが発生して終了するまで、または指定した時間が経過するまで、Tcl コマンドの実行を停止します。

このコマンドでは、run が終了したことは通知されますが、run の結果は返されません。run が正常に完了したかを判断するには、run の PROGRESS プロパティの値を取得します。

```
launch_runs synth_1
wait_on_run synth_1
if {[get_property PROGRESS [get_runs synth_1]] != "100%"} {
    error "ERROR: synth_1 failed"
}
```

wait_on_run コマンドは、実行されている run に使用できます。wait_on_run コマンドを実行したときに指定した run が実行されていない場合は、エラー メッセージが表示されます。既に完了している run に大してはエラー メッセージは表示されません。

注記: このコマンドは、バッチ モードまたは Tcl スクリプトでツールを実行する場合に使用します。GUI からインタラクティブに実行した場合は、無視されます。

引数

-timeout <arg> (オプション): run が終了するまで wait_on_run コマンドで待機する時間 (分) を指定します。これにより、指定した run が終了していなくても、この時間を越えるとツールが Tcl の実行を再開します。タイムアウトを指定しない場合 (run が終了するまでのツールの待機時間を制限しない場合)、デフォルト値の -1 が使用されます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<run>` (必須): 待機する run の名前を指定します。

例

次の例では、`impl_1` という run が実行され、指定した run が終了するか、1 時間経過するまで Tcl の実行が停止されます。

```
launch_runs impl_1
wait_on_run -timeout 60 impl_1
```

関連項目

- [launch_runs](#)

write_abstract_shell

現在のデザインの抽象化シェル チェックポイントを生成します。

構文

```
write_abstract_shell -cell <arg> [-force] [-quiet] [-verbose] <file>
```

戻り値

チェックポイント ファイル名

使用法

名前	説明
-cell	指定のリコンフィギュラブル セルの抽象化シェルを作成します。
[-force]	既存のチェックポイント ファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	デザイン チェックポイント ファイルを指定します。英数字を使用した名前に拡張子 (.dcp) を付けて指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

write_bd_layout

レイアウトをネイティブ、pdf、または svg でエクスポートします。

構文

```
write_bd_layout [-force] [-format <arg>] [-orientation <arg>]  
                [-scope <arg>] [-hierarchy <arg>] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-format]</code>	フォーマット指定します。有効な値は native、pdf、または svg です。ネイティブ フォーマットは、 <code>regenerate_bd_layout -layout_file</code> を使用して表示できます。デフォルトは native です。
<code>[-orientation]</code>	回路図の向きを指定します。有効な値は landscape または portrait です。
<code>[-scope]</code>	エクスポートする範囲を指定します。有効な値は visible、all で、デフォルトは all です。
<code>[-hierarchy]</code>	階層ブロックを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	出力ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

Vivado IP インテグレーターで現在開いているブロック デザインを指定のファイル フォーマットにエクスポートします。

このコマンドを使用すると、ブロック デザインを PDF に出力したり、デザイン プロジェクトに関連する資料で使用するためベクター グラフィック ファイルとして出力したり、`regenerate_bd_layout` コマンドを使用して Vivado IP インテグレーター デザイン キャンバスにブロック デザインのレイアウトを再作成したりできます。

このコマンドを実行すると、記述されたファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定したファイルが存在する場合に上書きします。

`-format [native | pdf | svg]` (オプション): ファイルを出力するフォーマットを指定します。デフォルトのファイルフォーマットは `native` で、`regenerate_bd_layout` コマンドを使用して Vivado IP インテグレーター デザイン キャンバスにブロック デザインのレイアウトを再作成するために使用できる内部 Vivado ツール フォーマットで記述します。SVG は Scalable Vector Graphics (スケーラブル ベクター グラフィックス) フォーマットです。PDF は Portable Document Format (ポータブル ドキュメント フォーマット) です。

注記: `native` フォーマットは、`regenerate_bd_layout` コマンドで使用できます。

`-orientation [landscape | portrait]` (オプション): グラフィック ファイルを横長 (`landscape`) または縦長 (`portrait`) のどちらの向きにするかを指定します。デフォルトは `portrait` です。

`-scope [visible | all]` (オプション): 指定のファイルに出力するブロック デザインの範囲を指定します。デフォルトは `all` で、ブロック デザイン全体が出力されます。`visible` を指定すると、デザイン キャンバスに現在表示されている部分、または現在のズーム値のみが使用されます。

`-hierarchy <arg>` (オプション): 指定した階層 `bd_cell` の図を出力します。ブロックは `get_bd_cells` コマンドを使用して指定できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): ブロック デザインのキャンバスを出力するファイルを指定します。指定したファイルが既に存在する場合、`-force` オプションを使用して既存のプロジェクトを上書きする必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のブロック デザインを指定した PDF ファイルに出力しています。

```
write_bd_layout -format pdf -orientation landscape C:/Data/microblaze.pdf
```

次の例では、ブロック デザインの指定した階層セルを指定した SVG ファイルに出力しています。

```
write_bd_layout -format svg -orientation landscape C:/Data/microblaze.svg
```

関連項目

- [create_bd_design](#)
- [current_bd_design](#)
- [open_bd_design](#)
- [regenerate_bd_layout](#)
- [write_schematic](#)

write_bd_tcl

現在のデザインをディスク上の Tcl ファイルにエクスポートします。

構文

```
write_bd_tcl [-force] [-bd_name <arg>] [-no_mig_contents] [-no_ip_version]
             [-ignore_minor_versions] [-bd_folder <arg>] [-check_ips <arg>]
             [-hier_blks <arg>] [-include_layout] [-exclude_layout] [-make_local]
             [-no_project_wrapper] [-exclude_pfm] [-quiet] [-verbose] <tcl_filename>
```

戻り値

TCL_OK、エラーが発生した場合は TCL_ERROR。

使用法

名前	説明
[-force]	既存のファイルを上書きします。
[-bd_name]	ブロック図の名前を指定します。デフォルトでは、現在のブロック図の名前が使用されます。
[-no_mig_contents]	生成された Tcl スクリプトに MIG PRJ の内容を含めず、PRJ を作業ディレクトリから読み込みます。デフォルトでは、Tcl スクリプトに MIG PRJ の内容が含まれます。
[-no_ip_version]	create_bd_cell コマンドの IP VLVN の一部として IP バージョンを含めないことを指定します。メジャー IP バージョンが変更されている場合に影響がある可能性があるため、注意してください。
[-ignore_minor_versions]	最新のマイナー バージョンを使用してデザインのセルを作成します。たとえば、プロジェクトに含まれている blk_mem_gen IP のバージョンが 7.3、7.4、8.3、8.4 であるとしします。デザインには blk_mem_gen_v7.4 があります。このフラグを使用すると、write_bd_tcl コマンドで「create_bd_cell -type ip -vlnv xilinx.com:ip:blk_mem_gen:7.* bmg_0_v7」という行が生成されます。生成されたこの Tcl スクリプトを使用すると、セル bmg_0_v7 で最新の blk_mem_gen_v7 が使用されます。
[-bd_folder]	リモート BD 機能: Tcl スクリプトを実行した場合に、デザインを生成するフォルダーを指定します。
[-check_ips]	デフォルト値は true で、デザインを再構築する前に IP カタログまたはプロジェクトに IP/モジュールが存在するかどうかをチェックされます。有効な値は true/false、yes/no、または 1/0 です。
[-hier_blks]	Tcl スクリプトで生成するデザインの階層ブロックをカンマで区切ってリストします。指定したブロックに含まれるサブ階層ブロックも含まれます。最上位デザイン部分は生成されません。
[-include_layout]	デフォルトでは、デザインの GUI レイアウトは含まれません。このオプションを使用すると、生成される Tcl スクリプトにレイアウト情報が含まれます。
[-exclude_layout]	このオプションは今後のリリースで削除される予定です。このリリースでは、下位互換性のためにサポートされています。このオプションを使用すると、生成される Tcl スクリプトにレイアウト情報は含まれません。

名前	説明
<code>[-make_local]</code>	リモート BD をローカル BD として書き出します。
<code>[-no_project_wrapper]</code>	BD 作成 Tcl プロシージャをラッパーなしで記述します。
<code>[-exclude_pfm]</code>	デザインの PFM 属性を除外します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><tcl_filename></code>	エクスポート先の Tcl ファイルの名前を指定します。

カテゴリ

IPIntegrator (IP インテグレーター)

説明

現在の IP インテグレーター サブシステム デザインをディスク上の Tcl スクリプト ファイルにエクスポートします。



重要: `<name>` で指定されたディレクトリ パスは既に存在している必要があり、存在しない場合はスクリプトは作成されません。

Tcl スクリプト ファイルを使用すると、IP インテグレーター サブシステム デザインの再作成、再使用、カスタマイズを実行でき、元のサブシステム デザインをアーカイブする必要はありません。

新しいツール リリースで作業する場合は、`write_bd_tcl` コマンドからの出力スクリプトを使用し、Tcl スクリプトを生成したのと同じツール リリースでブロック デザインを作成する必要があります。これにより、スクリプトで使用する IP の必要バージョンが確実に得られます。その後、作成されたブロック デザインを新しいツール リリースに移行します。

このコマンドが正常に実行された場合は `TCL_OK` が返され、正常に実行されなかった場合は、`-quiet` が指定されていなければ `TCL_ERROR` が返されます。

引数

`-force` (オプション): 同じ名前の `bd_tcl` ファイルが存在する場合に上書きします。

`-bd_name <arg>` (オプション): `bd_tcl` ファイルのブロック図に割り当てる名前を指定します。Tcl スクリプトを実行すると、新しいブロック図が指定の名前で作成されます。デフォルトでは、ブロック図の現在の名前が使用されます。

`-no_mig_contents` (オプション): 生成される Tcl スクリプトにメモリ IP PRJ の内容を含めません。デフォルトでは、Tcl スクリプトにメモリ IP PRJ の内容が含まれます。

`-no_ip_version` (オプション): `write_bd_tcl` ファイルに含まれる `create_bd_cell` コマンドの IP を指定する `Vendor:Library:Name:Version (VLNV)` 値で `Version` を指定しません。これにより、`bd_tcl` スクリプトで Vivado IP カタログからの最新バージョンの IP を使用して新しいブロック図が作成されます。

注記: これは、ブロック デザインで使用されている IP が `bd_tcl` ファイルが記述されてから使用するまでの間に大きくバージョン変更されている場合にデザインに影響します。

`-ignore_minor_versions` (オプション): IP VLVN にマイナー バージョンを含めません。このオプションを使用すると、`bd_tcl` スクリプトでメジャー バージョンを維持したまま IP の最新マイナー バージョンを使用できます。たとえば、IP `blk_mem_gen_v7.4` は `-vlvn xilinx.com:ip:blk_mem_gen:7.*` のように記述されます。

`-bd_folder <arg>` (オプション): `bd_tcl` スクリプトを実行したときにブロック図を生成するディレクトリを指定します。このオプションを使用すると、`bd_tcl` スクリプトが実行されたプロジェクトのディレクトリ構造外にブロックデザインを作成できます。

`-check_ips [true | false]` (オプション): ブロック図の作成を開始する前に必要なすべての IP が存在することを確認するチェックを Tcl スクリプトに追加します。存在しない IP がある場合は、スクリプトによりエラーが出力され、ブロック デザインは作成されません。デフォルト値は `true` です。

`-hier_blks <arg>` (オプション): 指定した階層ブロックまたはブロック デザインのモジュールのみの `write_bd_tcl` スクリプトを生成します。



ヒント: 階層ブロック モジュールは、`get_bd_cells` コマンドを使用して `bd_cell` オブジェクトとして指定する必要があります。指定した `bd_cell` が階層ブロックでない場合は、Tcl スクリプトは生成されません。

`-include_layout` (オプション): 生成される Tcl スクリプトにブロック デザインのグラフィカル レイアウト データを含めます。このオプションを使用すると、Vivado IP インテグレーター デザイン キャンバスのブロック デザインの現在のレイアウトが出力 Tcl スクリプトに保持されます。このオプションは、ブロック デザインの最上位からのレイアウト情報を記述するので、`-hier_blks` オプションが指定されている場合は無視されます。

`-make_local` (オプション): ブロック図が現在のプロジェクト外にある場合に、スクリプトを実行したときにブロック図をプロジェクトのローカルにて再生成します。

`-no_project_wrapper` (オプション): デザインの最上位ラッパーを作成せずにブロック図を再作成する Tcl スクリプトを記述します。

`-exclude_pfm` (オプション): 作成する `bd_tcl` スクリプトに SDx プラットフォームを定義するのに使用される PFM プロパティを含めません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<tcl_filename>`: 記述する Tcl ファイルの名前を指定します。拡張子を指定しない場合は、`.tcl` が付けられます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在の IP インテグレーター サブシステム デザインから Tcl スクリプトを作成しています。

```
write_bd_tcl C:/Data/myDesign.tcl
```

次の例では、現在の IP インテグレーター サブシステム デザインから Tcl スクリプトを作成しています。IP バージョンは含まれないようにし、出力された bd_tcl スクリプトを実行したときにブロック図を作成するフォルダーを指定しています。

```
write_bd_tcl -bd_name newMB1 -no_ip_version \  
-bd_folder C:/Block_Designs projMB1
```

注記: 指定したファイル名に拡張子 `.tcl` が追加され、`projMB1.tcl` という名前のファイルが作成されます。

次の例では、指定した階層ブロック セルの Tcl スクリプトを生成しています。同じ名前のスクリプトが存在する場合は上書きします。

```
write_bd_tcl -force -hier_blks [get_bd_cells myHier] \  
-include_layout C:/Data/myHier.tcl
```

注記: この例では、`-hier_blks` オプションが使用されているので、`-include_layout` オプションは無視されます。

関連項目

- [create_bd_design](#)
- [open_bd_design](#)
- [save_bd_design](#)

write_bitstream

現在のデザインのビットストリームを生成します。

構文

```
write_bitstream [-force] [-verbose] [-raw_bitfile] [-no_binary_bitfile]
  [-mask_file] [-readback_file] [-logic_location_file] [-bin_file]
  [-reference_bitfile <arg>] [-cell <arg>] [-no_partial_bitfile] [-quiet]
  <file>
```

使用法

名前	説明
[-force]	既存のファイルを上書きします。
[-verbose]	write_bitstream オプションを表示します。
[-raw_bitfile]	ロー ビット ファイル (.rbt) を生成します。
[-no_binary_bitfile]	バイナリ ビット ファイル (.bit) を生成しません。
[-mask_file]	マスク ファイル (.msk) を生成します。
[-readback_file]	リードバック ファイル (.rbd、.msd) を生成します。
[-logic_location_file]	ロジック ロケーション ファイル (.ll) を生成します。
[-bin_file]	ヘッダーのないバイナリ ビット ファイル (.bin) を生成します。
[-reference_bitfile]	パーシャル ビットストリームを生成する際に使用する基準ビット ファイルを指定します。
[-cell]	指定したセルのパーシャル ビットストリームのみを作成します。
[-no_partial_bitfile]	パーシャル リコンフィギュレーション デザインのパーシャル BIT ファイルを記述しません。
[-quiet]	コマンド エラーを表示しません。
<file>	生成する BIT ファイルの名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

現在のプロジェクトのビットストリーム ファイルを生成します。このコマンドは、インプリメント済みデザインで実行する必要があります。ビットストリームは、開いているインプリメント済みデザインに基づいて出力されます。

write_bitstream コマンドで生成可能なファイルは、次のとおりです。

- BIT ファイル: バイナリ ビット ファイル (.bit)。
- ロー ビット ファイル (ASCII): バイナリ ビットストリーム ファイルと同じ情報が ASCII 形式で含まれるロー ビット ファイル (.rbt)。
- マスク ファイル: ビットストリーム ファイルのコンフィギュレーション データの場所を示すマスク データを含むマスク ファイル (.msk)。

- ロジック ロケーション ファイル: ラッチ、フリップフロップ、LUT、ブロック RAM、I/O ブロック入力および出力のビットストリーム位置を示す ASCII 形式のロジック ロケーション ファイル (.ll)。
- BIN ファイル: 通常のビットストリーム ファイル (.bit) に含まれるヘッダー情報は含まない、デバイス プログラムデータのみを含むバイナリ ファイル (.bin)。
- 基準 BIT ファイル: 現在のビットストリームと指定の基準ビットストリームの違いのみを含むインクリメンタルビットストリーム ファイル。

Vivado ツールでは、インプリメント済みデザインの BITSTREAM.GENERAL.COMPRESS プロパティを設定して圧縮をイネーブルにすると、圧縮ビットストリームを生成できます。デバイス コンフィギュレーション プロパティの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。圧縮を有効にするには、次の Tcl コマンドを使用します。

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
```

Vivado Design Suite では、ビットストリームに含まれるデザインの知的所有権を保護するため、暗号化されたビットストリームも生成できます。暗号化されたビットストリームを作成するには、使用する暗号化のタイプと暗号化キーを定義する必要があります。これには、Vivado IDE で [Tools] → [Edit Device Properties] をクリックし、[Edit Device Properties] ダイアログ ボックスの [Encryption] ページで設定するのが最も簡単です。[Edit Device Properties] ダイアログ ボックスの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

暗号化は、次のようにインプリメント済みデザインに適切なプロパティを定義することにより手動でイネーブルにすることもできます。

```
set_property BITSTREAM.ENCRIPTION.ENCRYPT YES [get_designs impl_1]
set_property BITSTREAM.ENCRIPTION.ENCRYPTKEYSELECT EFUSE [get_designs impl_1]
set_property BITSTREAM.ENCRIPTION.KEY0 8675309 [get_designs impl_1]
```

暗号化に関連するプロパティは、次のとおりです。

- BITSTREAM.ENCRIPTION.ENCRYPT: write_bitstream コマンドでビットストリームを生成する際に暗号化をイネーブルにします。YES または NO に設定できます。
- BITSTREAM.ENCRIPTION.ENCRYPTKEYSELECT: ハードウェア デバイスに暗号キーを保存する方法を指定します。有効な値は、BBRAM (バッテリー バックアップ SRAM) および EFUSE (デバイス上の eFUSE レジスタ) です。



注意: eFUSE はハードウェア上の 1 回のみプログラム可能なセルであり、ファクトリでプログラムされる Device DNA、AES-GCM 暗号キー、およびユーザー指定の値を格納するために使用されます。eFUSE レジスタの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) または『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) を参照してください。

- BITSTREAM.ENCRIPTION.KEY0: BBRAM またはデバイス上の eFUSE FUSE_KEY レジスタに適用する暗号キーを指定します。キーは 256 ビット値で指定でき、ハードウェア デバイス (hw_device) をプログラム、検証、またはリロードバックするために暗号化されたビットストリームにアクセスする際に必要です。



ヒント: BITSTREAM.ENCRIPTION.KEY0 プロパティを指定すると、write_bitstream コマンドでビットストリーム ファイルと同じ名前の NKY ファイル (拡張子 .nky) が記述されます。この暗号化ファイルは、ほかのデザインで BITSTREAM.ENCRIPTION.KEYFILE プロパティを設定することにより使用できます。

- BITSTREAM.ENCRIPTION.KEYFILE: ENCRYPTION.KEY0 プロパティを設定する代わりに、暗号キー ファイル (NKY または NKZ) を指定します。指定した暗号キー ファイルは、ビットストリームの暗号化に使用されます。



重要: BITSTREAM.ENCRIPTION.KEY0 と BITSTREAM.ENCRIPTION.KEYFILE プロパティの両方が設定されている場合は、BITSTREAM.ENCRIPTION.KEY0 プロパティで指定された暗号キーが使用され、メッセージにそのことが示されます。

引数

-force (オプション): 同じ名前の既存のビットストリーム ファイルを上書きします。

-verbose (オプション): write_bitsream コマンドを実行したときにビットストリームに適用されるオプションの詳細を表示します。

-raw_bitfile (オプション): ロー ビット ファイル (.rbt) を生成します。ロー ビット ファイルには、バイナリ ビットストリーム ファイルと同じ情報が ASCII 形式で含まれます。出力ファイル名は <file>.rbt となります。

-no_binary_bitfile (オプション): バイナリ ビットストリーム ファイル (.bit) を生成しません。このオプションは、バイナリ ビットストリーム ファイルを生成せずに、ASCII 形式のビットストリーム ファイルまたはマスク ファイルを生成したり、ビットストリーム レポートを生成したりする場合に使用します。

-mask_file (オプション): マスク ファイル (.msk) を生成します。マスク ファイルには、ビットストリーム ファイルのコンフィギュレーション データが含まれる場所を示すマスク データが含まれます。このファイルは、検証時にビットストリームのどのビットをリードバック データと比較するべきかを判断するために使用します。マスク ビットが 0 の場合はそのビットはビットストリーム データに対して検証され、1 の場合は検証されません。出力ファイル名は <file>.msk となります。

-readback_file (オプション): 必要なリードバック ファイル (.rbd、.msd) を作成することにより、リードバック機能を実行します。

- .rbd: パッド ワードおよびフレームを含む、予測されるリードバック データのみを含む ASCII ファイルです。コマンドは含まれません。
- .msd: パッド ワードおよびフレームを含む、検証用のマスク情報のみを含む ASCII ファイルです。コマンドは含まれません。

-logic_location_file (オプション): ラッチ、フリップフロップ、LUT、ブロック RAM、I/O ブロック入力および出力のビットストリーム位置を示す ASCII 形式のロジック ロケーション ファイル (.ll) を生成します。このロケーション ファイルではビットがフレームおよびビット番号で示されるので、FPGA レジスタの内容を調べるのに役立ちます。

-bin_file (オプション): デバイス プログラム データのみを含むバイナリ ファイル (.bin) を生成します。通常のビットストリーム ファイル (.bit) に含まれるヘッダー情報は含まれません。

-reference_bitfile <arg> (オプション): 参照ビットストリーム ファイルを読み込み、指定した参照ファイルからの変更部分のみを含むインクリメンタル ビットストリーム ファイルを生成します。このパーシャル ビットストリームは、既存のデバイスをアップデートされたデザインでインクリメンタルにプログラムする場合に使用できます。

-cell <arg> (オプション): 指定したセルまたはデザイン階層のブロック レベルのパーシャル ビットストリーム ファイルを生成します。ビットストリーム ファイルには、指定したセルまたはモジュールのプログラム情報のみが含まれます。

-no_partial_bitfile (オプション): パーシャル リコンフィギュラブル モジュールまたはデザインのパーシャル BIT ファイルを記述しません。PR フローの詳細は、『Vivado Design Suite ユーザー ガイド: Dynamic Function eXchange』 (UG909) を参照してください。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

<file> (必須): 生成するビットストリーム ファイル (.bit) の名前を指定します。ファイルの拡張子を指定しない場合は、.bit が追加されます。.bit 以外の拡張子を指定することはできません。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、圧縮をイネーブルにし、指定した名前のビットストリーム ファイルを生成しています。

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]  
write_bitstream design1.bit
```

次の例では、バイナリおよび ASCII 形式のビットストリーム ファイルを生成しています。

```
write_bitstream -raw_bitfile C:/Data/design1
```

注記: 適切なファイル拡張子が追加されます。

関連項目

- [launch_runs](#)
- [program_hw_devices](#)
- [verify_hw_devices](#)

write_bmm

BMM ファイルを生成します。

構文

```
write_bmm [-force] [-quiet] [-verbose] <file>
```

戻り値

BMM ファイル名

使用法

名前	説明
<code>[-force]</code>	既存の BMM ファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	デザインの BMM ファイルを指定します。英数字を使用した名前に拡張子 (.bmm) を付けて指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ブロック RAM メモリ マップ (BMM) ファイルは、FPGA 上の各ブロック RAM がアドレス ブロックという連続アドレス空間にどのようにグループ化されているかを記述するテキスト ファイルです。

`write_bmm` コマンドは、現在のデザインからの BMM 情報を指定のファイルにエクスポートします。インプリメント済みデザインでは、BMM ファイルに配置情報が含まれます。`data2mem` コマンドは、BMM ファイルを入力として使用し、プログラム データをシミュレーション、デバイス プログラム、または SDK でのソフトウェア開発で使用する適切な形式に変換します。

このコマンドを実行すると、出力ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定した名前の BMM ファイルが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 記述する BMM ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のデザインの BMM ファイルを記述しています。

```
write_bmm C:/Data/design1.bmm
```

write_bsd1

デザインのコンフィギュレーション後の BSDL ファイル (.bsd) を生成します。

構文

```
write_bsd1 [-force] [-bsd <arg>] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-force]</code>	既存の .bsd ファイルを上書きします。
<code>[-bsd]</code>	アップデートされた汎用 BSDL ファイルを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	出力ファイル名を指定します。拡張子 .bsd はオプションです。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

ターゲット デバイスのコンフィギュレーション後のバウンダリスキャン アーキテクチャを反映した、現在のデザインのバウンダリスキャン記述言語 (BSDL) ファイル (.bsd) を生成します。

デバイスのバウンダリスキャン アーキテクチャは、デバイスをコンフィギュレーションしたときに変更されます。これは、バウンダリスキャン レジスタとパッドの間の接続が一部変更される可能性があるからです。これらの変更は、コンフィギュレーション後の BSDL ファイルによりバウンダリスキャン テスターに渡す必要があります。使用可能なコンフィギュレーション モードの詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

`write_bsd1` コマンドは、Vivado Design Suite インストール エリアからターゲット パーツ用のコンフィギュレーション前の BSDL ファイルを読み込み、現在のデザインからのコンフィギュレーション後のデータと統合します。

このコマンドを実行すると、出力 BSDL ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定した BSDL ファイルが存在する場合に上書きします。

`-bsd <arg>` (オプション): アップデートする既存の BSDL ファイルを指定します。このオプションは、Vivado Design Suite インストールからの汎用 BSDL ファイルを現在のデザインからのコンフィギュレーション後のコンフィギュレーション データでアップデートする場合に使用します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 出力ファイル名を指定します。拡張子 `.bsd` はオプションです。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定した場所に BSDL ファイルを生成しています。

```
write_bsd1 -force C:/Data/project/design1.bsd
```

関連項目

- [write_bitstream](#)

write_cfgmem

フラッシュ メモリをプログラムするためのファイルを作成します。

構文

```
write_cfgmem [-force] -format <arg> -size <arg> [-interface <arg>]
             [-checksum] [-disablebitswap] [-loadbit <arg>] [-loaddata <arg>]
             [-quiet] [-verbose] <file>
```

使用法

名前	説明
[-force]	既存のファイルを上書きします。
-format	生成するファイルのフォーマットを指定します。
-size	ターゲットのメモリのサイズを MB で指定します。2 のべき乗で指定する必要があります。
[-interface]	デバイスをプログラムするのに使用するインターフェイスを指定します。デフォルトは SMAPx8 です。
[-checksum]	各ファイルの 32 ビット チェックサムを算出します。異なるバイト値を指定しない場合は、メモリに 0xFF の値が挿入されます。デフォルトは 0xFF です。
[-disablebitswap]	BIT ファイルのバイトのビット スワップをディスエーブルにします。
[-loadbit]	BIT ファイルをメモリに指定のアドレスから読み込みます。
[-loaddata]	データをメモリに指定のアドレスから読み込みます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	生成するファイルの名前を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

このコマンドは、デザイン特定のコンフィギュレーション ビットストリーム ファイル(.bit) および指定のデータ ファイルを、program_hw_cfgmem コマンドを使用してフラッシュ メモリ デバイ스에プログラムするのに使用する指定のメモリ コンフィギュレーション ファイル フォーマットに変換します。サポートされるメモリ コンフィギュレーション ファイルのフォーマットは、MCS、BIN、および HEX です。



ヒント: デフォルトでは、write_cfgmem コマンドで生成したファイルでは、元の入力 BIT ファイルのバイトと比較して、バイト内のビットがスワップされています。-disablebitswap オプションを使用すると、下に説明するようにビットスワップをディスエーブルにできます。

ザイリンクス® FPGA にデザイン特定のデータを読み込む (プログラムする) プロセスは、コンフィギュレーションと呼ばれます。create_hw_cfgmem を使用して、ハードウェア デバイスのコンフィギュレーションおよびブートに使用するフラッシュ メモリ デバイスを定義します。

ハードウェア コンフィギュレーション メモリ (hw_cfgmem) オブジェクトを作成し、ハードウェア デバイス (hw_device) に関連付けたり、write_cfgmem コマンドで作成したメモリ コンフィギュレーション ファイルからのビットストリームおよびその他のデータでコンフィギュレーション メモリをプログラムできます。hw_cfgmem オブジェクトをプログラムするには、program_hw_cfgmem コマンドを使用します。

write_cfgmem -loadbit コマンドを使用すると、1 つまたは複数の指定したビットストリーム ファイルをメモリ コンフィギュレーション ファイルに読み込み、デバイスの使用可能なメモリの指定した開始アドレスから上位または下位に向かって挿入します。-loaddata オプションでファイルを読み込む開始アドレスを指定すると、データ ファイルをメモリ コンフィギュレーション ファイルに追加することもできます。



ヒント: -loadbit および -loaddata オプションを使用してデバイスのメモリにデータを挿入する場合、ビットストリームとデータ ファイルが使用可能なメモリに収まり、お互いに上書きされることがないように注意してください。データの競合が発生すると、write_cfgmem コマンドがエラーとなります。

write_cfgmem コマンドを実行すると、正常に実行された場合は実行されたプロセスが返され、正常に実行されなかった場合はエラーが返されます。

引数

-force (オプション): 同じ名前のファイルが存在する場合に上書きします。

-format [BIN | HEX | MCS] (必須): 生成するメモリ コンフィギュレーション ファイルのフォーマットを指定します。サポートされる値は BIN、HEX、MCS です。

-size <arg> (必須): ターゲットの PROM デバイスのサイズ制限を MB で指定します。2 のべき乗で指定する必要があります。

-interface <arg> (オプション): PROM デバイスをプログラムするのに使用するインターフェイスを指定します。有効な値は、SMAPx8 (デフォルト)、SMAPx16、SMAPx32、SERIALx1、SPIx1、SPIx2、SPIx4、SPIx8、BPIx8、BPIx16 です。このオプションは、バイト スワップがオンかオフかも指定します。デフォルトのインターフェイスは SMAPx8 です。



重要: コンフィギュレーション メモリ ファイルの指定したインターフェイス フォーマットは、program_hw_cfgmem コマンドでフラッシュ メモリ デバイスを正しくプログラムするために重要です。このオプションを使用する際は、ターゲット コンフィギュレーション メモリ パーツで生成されたファイルと一致するようにしてください。

-checksum (オプション): PROM ファイルの 32 ビット チェックサムを算出します。異なるバイト値を指定しない場合は、デバイスのメモリにデフォルトの 0xFF の値が挿入されます。このオプションは、メモリ コンフィギュレーション ファイルに含まれるチェックサム値を生成します。この値は、デバイス プログラムのチェックサムと一致する必要があります。このオプションは、フラッシュ メモリに正しいデータがプログラムされたことを確認するために使用してください。

-disablebitswap (オプション): ビットストリーム ファイルのバイトのデフォルトのビット スワップをディセーブルにします。デフォルトでは、write_cfgmem コマンドで記述したファイルでは、元の入力 BIT ファイルのバイトと比較して、バイト内のビットがスワップされています。このオプションは、この出力ファイルのビット スワップをディセーブルにします。

-loadbit <arg> (オプション): 1 つまたは複数のビットストリーム ファイルの読み込みを開始する PROM デバイスの開始アドレスを指定します。このオプションは、次の形式の文字列で指定します。

```
"up|down 0x0 <bitfile1>.bit <bitfile2>.bit"
```

説明:

- `up` | `down`: 1 つまたは複数の BIT ファイルをメモリに、指定したアドレスから開始して上位方向または下位方向に読み込みます。
- `0x0`: ビットストリームを読み込む開始アドレスを 16 進数値で指定します。
- `<bitfile>.bit`: フラッシュ メモリ デバイスに読み込むビットストリーム ファイル (`.bit`) を指定します。複数のビットストリーム ファイルを指定すると、ファイルがデジタイズ チェーンに結合されます。



ヒント: `-loadbit` オプションは 1 回しか指定できませんが、複数のビットストリーム ファイルを異なる開始アドレスから読み込むのに必要な数だけ引数を指定できます。

```
-loadbit "up 0 bitfile1.bit up 0xFFFFFFFF bitfile2.bit"
```

`-loaddata <arg>` (オプション): コンフィギュレーション デバイスのメモリに開始アドレスから読み込むデータ ファイルを指定します。`-loaddata` オプションの文字列の形式は `-loadbit` オプションの引数と同じで、メモリ コンフィギュレーション ファイルに追加する方向、開始アドレス、データ ファイル名を指定します。データ ファイルは、フラッシュ メモリ デバイスに追加のフォーマットなしでそのまま追加されます。

- `up` | `down`: 1 つまたは複数のデータ ファイルをメモリに、指定したアドレスから開始して上位方向または下位方向に読み込みます。
- `0x0`: データ ファイルを読み込む開始アドレスを 16 進数値で指定します。
- `<data_file>`: フラッシュ メモリ デバイスに読み込むデータ ファイルを指定します。複数のデータ ファイルを指定すると、ファイルがデジタイズ チェーンに結合されます。

注記: `-loadbit` および `-loaddata` はオプションですが、いずれか 1 つを指定してメモリ コンフィギュレーション ファイルのデータを指定する必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 作成するメモリ コンフィギュレーション ファイルの名前を指定します。ファイルの拡張子は、指定したフォーマット (`.mcs`) と一致している必要があり、ファイル名の一部として指定する必要はありません。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、サイズ制限を 64 MB に設定し、指定したビットストリーム ファイルを開始アドレスから上位方向に読み込んで、指定したメモリ コンフィギュレーション ファイルを MCS フォーマットで作成しています。

```
write_cfgmem -format MCS -size 64 -loadbit "up 0x0 \
C:/Data/Vivado_Debug/project_debug/project_debug.runs/impl_1/
sinegen_demo.bit" \
config_memory1
```

関連項目

- [create_hw_cfgmem](#)
- [current_hw_device](#)
- [delete_hw_cfgmem](#)
- [get_cfgmem_parts](#)
- [get_property](#)
- [program_hw_cfgmem](#)
- [readback_hw_cfgmem](#)
- [set_property](#)
- [write_bitstream](#)

write_checkpoint

現在のデザインのチェックポイントを書き出します。

構文

```
write_checkpoint [-force] [-cell <arg>] [-logic_function_stripped]
                [-encrypt] [-key <arg>] [-incremental_synth] [-quiet] [-verbose] <file>
```

戻り値

チェックポイント ファイル名

使用法

名前	説明
<code>[-force]</code>	既存のチェックポイント ファイルを上書きします。
<code>[-cell]</code>	指定したセルのチェックポイントを記述します。
<code>[-logic_function_stripped]</code>	LUT および RAMB の INIT 文字列を固定の値に変換します。結果のネットリストは機能しません。
<code>[-encrypt]</code>	保護されていないモジュールを IEEE 1735 IP セキュリティ バージョン 2 を使用して暗号化します。
<code>[-key]</code>	-encrypt オプションと共に使用し、キー ファイルを指定します。デフォルトでは、サイリンクスの公開キーが使用されます。
<code>[-incremental_synth]</code>	インプリメンテーションを再利用するのに使用する合成アーカイブ ファイルをエクスポートします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	デザイン チェックポイント ファイルを指定します。英数字を使用した名前に拡張子 (.dcp) を付けて指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

デザインをデザイン プロセスの任意の段階で保存し、必要に応じてツールにすばやくインポートし直せるようにします。デザイン チェックポイント (DCP) には、ネットリスト、制約、およびインプリメント済みデザインの配置配線情報が含まれます。



ヒント: プロジェクト モードでは、合成後 DCP にはタイミング制約はありません。タイミング制約は、`open_run` または `link_design` コマンドを実行したとき、またはインプリメンテーション `run` を実行したときに、デザインにアノテートされます。タイミング制約を含む DCP を作成するには、`opt_design` またはインプリメンテーション `run` の完了後にデザイン チェックポイントを作成します。

チェックポイント ファイルをインポートするには、`read_checkpoint` コマンドを使用します。

引数

`-force` (オプション): 同じ名前のチェックポイント ファイルが存在する場合に上書きします。

`-cell <arg>` (オプション): 指定した階層セルの内容をチェックポイント ファイルに出力します。出力するように指定できるセルは 1 つのみです。

`-logic_function_stripped` (オプション): LUT および RAM の INIT 値を固定の値に変換し、デバッグ用のチェックポイントを作成します。シミュレーションまたは合成では正しく機能しません。

`-abstract_shell` (オプション): SDx プラットフォーム デザイン用に縮小されたネットリストを生成します。シェルの詳細を含まないフル シェルを抽象化したものがエンド ユーザーに示されるので、全体的なインプリメンテーションの実行時間が短縮されます。

`-rm_cell <arg>` (オプション) `-abstract_shell` オプションでブラック ボックスに変換されるリコンフィギュラブル モジュール セルを指定します。

`-encrypt` (オプション): チェックポイント ネットリストを暗号化します。

`-key <arg>` (オプション): チェックポイントの暗号化および復号化に使用するキーを指定します。指定しない場合は、デフォルトでザイリンクス公開キーが使用されます。

`-incremental_synth` (オプション): 合成チェックポイントがインクリメンタル合成に使用されるようにします。この後のインクリメンタル合成 run に使用されるように、チェックポイントをインクリメンタル合成用にイネーブルにする必要があります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 作成するチェックポイント ファイルの名前を指定します。拡張子を指定しない場合、`.dcp` という拡張子が使用されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定のチェックポイント ファイルを作成しています。同じ名前のチェックポイント ファイルが存在する場合は上書きします。

```
write_checkpoint C:/Data/checkpoint1 -force
```

注記: 指定のファイル名に拡張子 `.dcp` が付けられ、既存の `checkpoint1.dcp` ファイルが上書きされます。

関連項目

- [read_checkpoint](#)

write_csv

パッケージ ピンとポート配置情報をエクスポートします。

構文

```
write_csv [-force] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ピン プランニング CSV ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

パッケージ ピンおよびポート配置情報を CSV (Comma Separated Value) ファイルに記述します。

CSV ファイルの形式および要件は、『Vivado Design Suite ユーザー ガイド: I/O およびクロックの配置』(UG899) を参照してください。

引数

`-force` (オプション): 同じ名前の CSV が存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する CSV ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のプロジェクトから CSV ファイルがエクスポートされます。

```
write_csv C:/Data/pinList.csv
```

関連項目

- [read_csv](#)

write_debug_probes

デバッグ プローブをファイルに記述します。

構文

```
write_debug_probes [-cell <arg>] [-no_partial_ltxfile] [-force] [-quiet]  
                  [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
[-cell]	リコンフィギャラブル パーティション セルの階層名を指定します。
[-no_partial_ltxfile]	パーシャル LTX ファイルを生成しません。
[-force]	既存のファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	デバッグ プローブ ファイル名 (デフォルト拡張子 .ltx) を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Debug \(デバッグ\)](#)

説明

現在のデザインに追加された ILA デバッグ コアおよび信号プローブを含む Vivado ロジック解析プローブ ファイルを作成します。デバッグ プローブ データ ファイルの拡張子は通常 .ltx です。

ILA コアは、`create_debug_core` コマンドを使用してデザインに追加できます。ILA プローブは、`create_debug_port` コマンドを使用してデザインに追加し、`connect_debug_port` コマンドを使用してデザインのネットに接続します。

デバッグ プローブ ファイルの詳細および使用方法は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

引数

`-cell <arg>` (オプション): パーシャル プローブ ファイルをエクスポートするリコンフィギャラブル パーティション セルの階層名を指定します。

`-force` (オプション): 指定のファイルが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述するデバッグ プローブ ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のデザインからデバッグ プローブ ファイルを記述しています。

```
write_debug_probes C:/Data/designProbes.ltx
```

関連項目

- [create_debug_core](#)
- [implement_debug_core](#)

write_edif

現在のネットリストを EDIF ファイルとしてエクスポートします。

構文

```
write_edif [-pblocks <args>] [-cell <arg>] [-force] [-security_mode <arg>]
           [-logic_function_stripped] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名またはディレクトリ名

使用法

名前	説明
[-pblocks]	指定した Pblock のネットリストをエクスポートします。-cell と共に使用することはできません。
[-cell]	指定したセルのネットリストをエクスポートします。-pblocks と共に使用することはできません。
[-force]	既存のファイルを上書きします。
[-security_mode]	all に設定すると、デザインの一部を暗号化する場合に、デザイン全体が 1 つの暗号化ファイルに記述されます。デフォルトは multifile です。
[-logic_function_stripped]	LUT および RAMB の INIT 文字列を固定の値に変換します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	出力ファイルを指定します。-pblocks、-cell を指定している場合は、ディレクトリを指定します。

カテゴリ

FileIO (ファイル入力および出力)

説明

現在のネットリストを EDIF ファイルとして書き出すか、指定の Pblock または階層セルの内容を EDIF ネットリスト ファイルとして出力します。

-pblocks または -cell オプションのいずれかを使用する場合、この引数で各 Pblock またはセルの EDIF ネットリスト ファイルを保存するディレクトリ名を指定します。EDIF ネットリスト ファイルの名前は、その Pblock またはセルと同じになります。指定したディレクトリが存在しない場合は、エラーが返されます。

引数

-pblocks <arg> (オプション): 指定した Pblock の内容を EDIF ネットリスト ファイルとして出力します。各 Pblock の内容が個別の EDIF ファイルに書き込まれます。これらのファイルはデザイン ソース ファイルとしてネットリスト プロジェクトに追加できますが、update_design を使用してデザインに読み込むことはできません。

update_design を使用して EDIF ネットリストを記述するには、-cell オプションを使用します。

`-cell <arg>` (オプション): 指定した階層セルの内容を EDIF ネットリスト ファイルとして出力します。出力するよう指定できるセルは 1 つのみです。

注記: `-pblocks` と `-cell` オプションを同時に使用することはできません。これらはオプションですが、一度に指定できるのはいずれか 1 つのみです。

`-force` (オプション): 指定した名前の EDIF ファイルが存在する場合に上書きします。

`-security_mode [multifile | all]` (オプション): デザインに暗号化された IP がある場合に、複数の EDIF ファイルを記述するか、デザイン全体を 1 つのファイルに記述するかを指定します。

- `multifile`: これがデフォルト設定です。デフォルトではデザイン全体のネットリストが指定したファイルに記述されますが、デザインに保護された IP が含まれている場合、保護されているモジュールの内容を含む暗号化されたファイルが作成されます。これにより、デザインの保護されたエレメントと保護されていないエレメントを含む、複数の EDIF ファイルが生成されます。
- `all`: 暗号化されたセルと暗号化されていないセルの両方を指定した 1 つのファイルに記述します。

`-logic_function_stripped` (オプション): LUT および RAM の INIT 値を固定の値に変換し、デバッグ用のネットリストを作成します。シミュレーションまたは合成では正しく機能しません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する EDIF ファイルの名前を指定します。EDIF ネットリストのデフォルトのファイル拡張子は、`.edn` です。`-pblocks` または `-cell` オプションを使用している場合は、ここで指定した名前はディレクトリ名を指します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、デザイン全体の EDIF ネットリスト ファイルを指定したファイル名で書き出します。

```
write_edif C:/Data/edifOut.edn
```

次の例では、デザインのすべての Pblock の EDIF ネットリストが出力されます。ファイルは、指定したディレクトリに書き込まれます。

```
write_edif -pblocks [get_pblocks] C:/Data/FPGA_Design/
```

関連項目

- [read_edif](#)

write_hw_ila_data

ハードウェア ILA データをファイルに記述します。

構文

```
write_hw_ila_data [-force] [-csv_file] [-vcd_file] [-legacy_csv_file]
                  [-quiet] [-verbose] <file> [<hw_ila_data>] [<hw_ila_data>]
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-csv_file]</code>	CSV ファイルのみをエクスポートします。
<code>[-vcd_file]</code>	VCD ファイルのみをエクスポートします。
<code>[-legacy_csv_file]</code>	CSV ファイルを基数情報なしでエクスポートします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ハードウェア ILA データ ファイル名を指定します。
<code>[<hw_ila_data>]</code>	ハードウェア ILA データ オブジェクトを指定します。デフォルトは、現在のハードウェア ILA データです。

カテゴリ

Hardware (ハードウェア)

説明

指定のハードウェア ILA データ (hw_ila_data) オブジェクトに格納されている ILA デバッグ コアのサンプル データをバイナリ ファイルに記述し、ディスクに保存します。

hw_ila_data オブジェクトは、ハードウェア デバイス (hw_device) でハードウェア ILA (hw_ila) をトリガーしたとき、または upload_hw_ila_data コマンドを使用して物理 FPGA デバイス (hw_device) からキャプチャされたデータをアップロードする際に作成されます。

write_hw_ila_data コマンドを使用すると、hw_ila_data オブジェクトのデータを後で使用するために、バイナリ ファイルとしてディスクに保存できます。ILA デバッグ コアのデータを Vivado ロジック解析機能にリードバックするには、read_hw_ila_data コマンドを使用します。これにより、新しい hw_ila_data オブジェクトが作成されます。

このコマンドを実行すると、記述されたファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 同じ名前のファイルが存在する場合に上書きします。

`-csv_file` (オプション): CSV ファイルのみをエクスポートします。これにより、`write_hw_ila_data` ILA データがデフォルトのバイナリ ILA ファイル フォーマットではなく、スプレッドシートやサードパーティ アプリケーションにインポート可能な CSV ファイルの形式でエクスポートされます。

`-vcd_file` (オプション): VCD (Value Change Dump) ファイルのみをエクスポートします。これにより、`write_hw_ila_data` コマンドを実行したときに、ILA データがデフォルトのバイナリ ILA ファイル フォーマットではなくサードパーティ アプリケーションまたはビューアーにインポート可能な VCD ファイルの形式でエクスポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述する ILA データ ファイルの名前を指定します。ILA データ ファイルのデフォルトのファイル拡張子は `.ila` です。デフォルトのファイル拡張子は、`-csv_file` オプションを使用した場合は `.csv`、`-vcd_file` オプションを使用した場合は `.vcd` です。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`<hw_ila_data>` (オプション): 指定したファイルに記述するハードウェア ILA データを指定します。`hw_ila_data` は、`get_hw_ila_data` または `current_hw_ila_data` コマンドを使用してオブジェクトとして指定する必要があります。`hw_ila_data` オブジェクトを指定しない場合、現在の `hw_ila_data` が指定したファイルに記述されます。

例

次の例では、`hw_ila` デバッグ コアのデータを `hw_ila_data` オブジェクトにアップロードし、そのデータ オブジェクトを指定した ILA データ ファイルに記述しています。ファイルが既に存在する場合は、上書きされます。

```
write_hw_ila_data -force design1_ila_data [upload_hw_ila_data hw_ila_1]
```

次の例では、`hw_ila` をトリガーし、キャプチャされた `hw_ila_data` を CSV ファイルに記述しています。

```
run_hw_ila hw_ila_1
write_hw_ila_data -csv_file C:/Data/design1_ila_data [current_hw_ila_data]
```

関連項目

- [current_hw_ila](#)
- [current_hw_ila_data](#)
- [get_hw_ilas](#)
- [get_hw_ila_datas](#)
- [run_hw_ila](#)
- [upload_hw_ila_data](#)

- [wait_on_hw_ila](#)

write_hw_platform

現在のデザインのザイリンクス シェル アーカイブを生成します。

構文

```
write_hw_platform [-fixed] [-force] [-include_bit] [-include_emulation]  
                  [-minimal] [-quiet] [-verbose] [<file>]
```

戻り値

シェル ファイル名

使用法

名前	説明
<code>[-fixed]</code>	固定シェルを記述します。
<code>[-force]</code>	既存のザイリンクス シェル アーカイブ ファイルを上書きします。
<code>[-include_bit]</code>	シェルに BIT ファイルを含めます。
<code>[-include_emulation]</code>	ハードウェア エミュレーション サポートを生成してシェルに含めます。
<code>[-minimal]</code>	シェルに最小限のファイルのみを含めます。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<file>]</code>	デバイス サポート ファイルを指定します。英数字を使用した名前に拡張子 .xsa を付けて指定します。

カテゴリ

[FileIO](#) (ファイル入力および出力)、[Platform](#) (プラットフォーム)、[Vitis](#)

説明

現在のデザインのハードウェア プラットフォームとして使用するザイリンクス サポート アーカイブ (XSA) を生成します。

すべてのプラットフォームは、コンパイル中に動的にインプリメントされます。つまり、アクセラレータ ロジックは、ハードウェア プラットフォーム デザインに含まれるロジックの一部またはすべてと共にインプリメントされます。XSA は、ハードウェア プラットフォームの必須部分を表します。XSA を作成するのに使用するハードウェア プラットフォーム デザインには、Vivado IP インテグレーター サブシステム デザインと、コンフィギュレーションされてデバイス I/O に接続された必要なボード インターフェイスすべてが含まれます。Vivado プロジェクトには、XSA を定義するために必要な複数の XSA および PFM プロパティを含める必要があります。

このコマンドを実行すると、生成された XSA ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-fixed` (オプション): 固定シェル XSA を上書きします。このファイルは、ソフトウェア開発フローでプラットフォームを使用できるようにします。アクセラレーションには使用されません。

`-force` (オプション): 指定した名前の XSA ファイルが存在する場合に上書きします。

`-include_bit` (オプション): XSA に現在のデザインのビットストリームを含めます。デフォルトでは、`write_hw_platform` コマンドにより MCS ファイルが作成され、ビットストリームは破棄されます。このオプションを指定すると、ビットストリーム ファイルが保持され、プラットフォームのデバッグに使用されます。

`-include_emulation` (オプション): XSA を作成すると同時に、ハードウェア エミュレーションをサポートするのに必要なデータをエクスポートします。このオプションは、ハードウェア プラットフォームの特別機能が必要で、通常は推奨されません。

`-minimal` (オプション): XSA に SDx フローの実行に必要なファイルのみを含めます。それ以外は追加しません。このオプションを指定すると、`write_hw_platform` コマンドで XSA に内容を追加する `-include_bit`、`-include_emulation` オプションなどのオプションは無視されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する XSA ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のプロジェクトの XSA を生成しています。同じ名前の XSA ファイルが存在する場合は上書きされます。

```
write_XSA -force C:/Data/zc702.xsa
```

関連項目

- [open_hw_platform](#)
- [validate_hw_platform](#)

write_hw_platform_metadata

現在のデザインの統合 JSON メタデータ ファイルを生成します。

構文

```
write_hw_platform_metadata [-quiet] [-verbose] [<file>]
```

戻り値

統合 JSON メタデータ ファイル名

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code>[<file>]</code>	統合 JSON ファイルを指定します。英数字を使用した名前に拡張子 <code>.json</code> を付けて指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Platform \(プラットフォーム\)](#)、[Vitis](#)

説明

現在のデザインで表されるプラットフォームの JSON メタデータ ファイルを生成します。

このコマンドを実行すると、生成された JSON ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する JSON ファイルの名前を指定します。ファイル拡張子 `.json` を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例は、現在のプロジェクトの JSON ファイルを生成します。

```
write_hw_platform_metadata C:/Data/zc102_platform.json
```

関連項目

- [open_hw_platform](#)
- [validate_hw_platform](#)
- [write_hw_platform](#)

write_hw_sio_scan

スキャン データをファイルに記述します。

構文

```
write_hw_sio_scan [-force] [-quiet] [-verbose] <file> <hw_sio_scan>
```

戻り値

出力ファイル名

使用法

名前	説明
<code>[-force]</code>	既存のファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ハードウェア SIO スキャン ファイルの名前を指定します。
<code><hw_sio_scan></code>	ハードウェア SIO スキャン データ オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

`run_hw_sio_scan` コマンドが完了した後、ハードウェア SIO スキャン (`hw_sio_scan`) オブジェクトを記述します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用のアイ スキャン ハードウェアを使用してリンクのスキャンを実行すると有益です。Vivado シリアル I/O 解析機能では、リンク スキャンを作成、実行、および保存できます。

このコマンドを使用すると、スキャンの実行が完了した後にスキャンをディスクに保存できます。ファイルのフォーマットは、スキャンの実行中に検出された値の CSV ファイルです。

このコマンドを実行すると、出力されたファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定のファイルが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述するスキャン ファイルの名前を指定します。拡張子を指定しない場合は、デフォルトの拡張子 `.csv` が使用されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

`<hw_sio_scan>` (必須): ディスクに記述する `hw_sio_scan` オブジェクトを指定します。`hw_sio_scan` は、`get_hw_sio_scans` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定の `hw_sio_scan` オブジェクトをディスクに記述しています。同じ名前のファイルが存在する場合は上書きします。

```
write_hw_sio_scan -force C:/Data/Vivado_Debug/LoopBack_1.csv \
[get_hw_sio_scans {SCAN_3}]
```

関連項目

- [create_hw_sio_scan](#)
- [create_hw_sio_sweep](#)
- [get_hw_sio_scans](#)
- [get_hw_sio_sweeps](#)
- [run_hw_sio_scan](#)
- [run_hw_sio_sweep](#)
- [remove_hw_sio_scan](#)
- [stop_hw_sio_scan](#)
- [wait_on_hw_sio_scan](#)

write_hw_sio_sweep

スイープ データをディレクトリに記述します。

構文

```
write_hw_sio_sweep [-force] [-quiet] [-verbose] <directory> <hw_sio_sweep>
```

戻り値

出力ディレクトリ名

使用法

名前	説明
<code>[-force]</code>	既存のディレクトリを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><directory></code>	ハードウェア SIO スイープ ディレクトリへのパスを指定します。
<code><hw_sio_sweep></code>	ハードウェア SIO スイープ データ オブジェクトを指定します。

カテゴリ

[Hardware \(ハードウェア\)](#)

説明

`run_hw_sio_sweep` コマンドが完了した後、ハードウェア SIO スイープ (`hw_sio_sweep`) オブジェクトを記述します。

リンクのマージンを解析するには、ザイリンクス UltraScale デバイスまたは 7 シリーズ FPGA 専用の機能を使用してリンクのスキャンを実行すると有益です。また、リンクに対して GT の異なる設定を使用して複数のスキャンを実行するのも有益です。デザインにどの設定が最適かを判断するのに役立ちます。Vivado シリアル I/O 解析機能では、リンク スイープ (ある値の範囲で実行するリンク スキャンのグループ) を定義、実行、および保存できます。

このコマンドは、`run_hw_sio_sweep` コマンドによりデータが挿入された指定のリンク スイープ オブジェクトをディスクに保存します。

このコマンドを実行すると、作成されたディレクトリの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定のディレクトリが存在する場合に上書きします。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<directory>` (必須): `run_hw_sio_sweep` コマンドの結果である複数のスキャン ファイルを保存するディレクトリの名前を指定します。

`<hw_sio_sweep>` (必須): ディスクに記述する `hw_sio_sweep` オブジェクトを指定します。`hw_sio_sweep` は、`get_hw_sio_sweeps` コマンドを使用してオブジェクトとして指定する必要があります。

例

次の例では、指定した `hw_sio_sweep` オブジェクトをディスクに記述しています。

```
write_hw_sio_sweep sweep_results [get_hw_sio_sweeps {SWEEP_1}]
```

関連項目

- [create_hw_sio_scan](#)
- [create_hw_sio_sweep](#)
- [current_hw_device](#)
- [get_hw_sio_scans](#)
- [get_hw_sio_sweeps](#)
- [remove_hw_sio_sweep](#)
- [run_hw_sio_sweep](#)
- [stop_hw_sio_sweep](#)
- [wait_on_hw_sio_sweep](#)

write_hw_svf

current_hw_target の SVF ファイルを生成します。

構文

```
write_hw_svf [-force] [-quiet] [-verbose] <file_name>
```

使用法

名前	説明
<code>[-force]</code>	SVF ファイルが既に存在する場合に上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file_name></code>	SVF ファイルの名前を指定します。

カテゴリ

Hardware (ハードウェア)

説明

Vivado ハードウェア マネージャーでは、SVF (Serial Vector Format) ファイルを使用したハードウェア デバイスのプログラムがサポートされています。SVF ファイルは、プログラム命令とコンフィギュレーション データを含む ASCII ファイルです。これらのファイルは、ATE マシンおよびエンベデッド コントローラーでバウンダリスキャン操作を実行するために使用されます。SVF ファイルには、ビットストリームをザイリンクス デバイスに直接プログラム、またはフラッシュ メモリ デバイスに間接プログラムするのに必要な JTAG コマンドが含まれます。SVF ファイルは、`write_hw_svf` コマンドを使用して記述し、`execute_hw_svf` コマンドを使用して開いているハードウェア ターゲット (`hw_target`) に適用します。詳細は、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) を参照してください。

`hw_svf` ファイルを作成するプロセスは、次のとおりです。

1. `create_hw_target` コマンドを使用して SVF ターゲットを作成します。
2. SVF ターゲットを開きます。
3. `create_hw_device` コマンドを使用して、SVF ターゲットに 1 つまたは複数のデバイスを作成します。
4. `program_hw_devices` などのコマンドを使用してデバイスをプログラムします。
5. `write_hw_svf` コマンドを使用して操作コマンドの SVF ファイルを記述します。

上記の手順 4 で `hw_devices` をプログラムすると、操作の SVF コマンドが一時ファイルにキャッシュされます。`write_hw_svf` コマンドは、このキャッシュにファイル名を指定して保存し、指定のファイル パスに移動します。

注記: このコマンドはキャッシュされた SVF コマンドを消去するので、`write_hw_svf` コマンドを実行するとキャッシュがクリアされ、新規デバイスのコマンドのキャプチャが再開されます。

このコマンドを実行すると、正常に実行されたことを示すメッセージが返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-force (オプション): 指定した名前のファイルが存在する場合に上書きします。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<file_name>: 記述する SVF ファイルの名前を指定します。Vivado ツールではファイル拡張子が自動的に付けられないので、ファイル名はファイル拡張子 .svf を付けて指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定したディレクトリに SVF ファイルを記述しています。

```
program_hw_devices [lindex [get_hw_devices] 0]
write_hw_svf C:/Data/k7_design.svf
```

次の例では、SVF ターゲット上に複数のデバイスを作成する際の正しい順序を示します。SVF ターゲットが作成されて開かれ、現在のターゲットにザイリンクス デバイス、ユーザー パーツ、および 2 つ目のザイリンクス デバイスが作成されています。2 つのザイリンクス デバイスに対してビットストリーム プロパティが定義され、デバイスがプログラムされて、SVF ファイルが記述されます。

```
open_hw
connect_hw_server
create_hw_target my_svf_target
open_hw_target
create_hw_device -part xc7k325t
create_hw_device -idcode 01234567 -irlength 8 -mask ffffffff -part
userPart1
create_hw_device -part xc7k325t
set_property PROGRAM.FILE {C:/Data/k7_design.bit} [lindex [get_hw_devices]
0]
set_property PROGRAM.FILE {C:/Data/ku_design.bit} [lindex [get_hw_devices]
2]
program_hw_devices [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 2]
write_hw_svf C:/Data/myDesign.svf
```

次の例では、SVF ターゲット上にデバイスを作成し、そのデバイスに関連付けるコンフィギュレーション メモリ オブジェクト (hw_cfgmem) を作成して、デバイスおよびコンフィギュレーション メモリをプログラムし、そのコマンド シーケンスを SVF ファイルに保存しています。

```
create_hw_target my_svf_target
open_hw_target
set_device [create_hw_device -part xc7k325t]
set_property PROGRAM.FILE {C:/Data/k7_design.bit} $device
create_hw_cfgmem -hw_device $device -mem_dev [lindex \
```

```
[get_cfgmem_parts {28f00am29ew-bpi-x16}] 0]
set cfgMem [current_hw_cfgmem]
set_property PROGRAM.ADDRESS_RANGE {use_file} $cfgMem
set_property PROGRAM.BLANK_CHECK 0 $cfgMem
set_property PROGRAM.BPI_RS_PINS {none} $cfgMem
set_property PROGRAM.CFG_PROGRAM 1 $cfgMem
set_property PROGRAM.CHECKSUM 0 $cfgMem
set_property PROGRAM.ERASE 1 $cfgMem
set_property PROGRAM.UNUSED_PIN_TERMINATION {pull-none} $cfgMem
set_property PROGRAM.VERIFY 1 $cfgMem
set_property PROGRAM.FILES [list {C:/data/flash.mcs} ] $cfgMem
create_hw_bitstream -hw_device $device [get_property \
PROGRAM.HW_CFGMEM_BITFILE $device]
program_hw_devices $device
program_hw_cfgmem -hw_cfgmem $cfgMem
write_hw_svf C:/Data/myDesign.svf
```

関連項目

- [create_hw_bitstream](#)
- [create_hw_cfgmem](#)
- [create_hw_device](#)
- [create_hw_target](#)
- [execute_hw_svf](#)
- [get_cfgmem_parts](#)
- [open_hw_target](#)
- [program_hw_cfgmem](#)
- [program_hw_devices](#)
- [set_property](#)

write_hwdef

ソフトウェア開発に使用するハードウェア定義を記述します。

構文

```
write_hwdef [-force] [-quiet] [-verbose] <file>
```

戻り値

コマンドが正常に実行されたか、エラーが発生したか。

使用法

名前	説明
<code>[-force]</code>	既存のハードウェア定義ファイルを上書きします。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ハードウェア定義ファイルを指定します。英数字を使用した名前に拡張子 (.hwdef) を付けて指定します。

カテゴリ

[Project \(プロジェクト\)](#)

説明

ソフトウェア開発キット (SDK) で使用するハードウェア定義 (.hwdef) ファイルを記述します。

`write_hwdef` コマンドは、Vivado Design Suite から SDK でのソフトウェア開発への移行を簡略化します。このコマンドは、MicroBlaze や Zynq-7000 SoC などのエンベデッド プロセッサを使用するブロック デザインを含む最上位デザイン用に Vivado Design Suite で出力ファイルを生成すると自動的に実行されます。ブロック デザインは、`create_bd_design` コマンドを使用して、Vivado Design Suite の IP インテグレーターで作成します。

`write_hwdef` コマンドは `place_design` の後に実行され、拡張子が .hwdef のハードウェア コンテナ ファイルを作成します。このコンテナ ファイルには、デバイスのメタデータおよびハードウェア デザイン ファイルが含まれます。

`write_hwdef` コマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 同じ名前のハードウェア定義ファイルが存在する場合に上書きします。このオプションを指定しない場合、既存のファイルは上書きされません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): ハードウェア定義ファイルの名前を指定します。指定するファイル名には、パスとファイル拡張子を含めることができます。拡張子を指定しない場合は、デフォルトの拡張子 `.hwdef` が使用されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定のハードウェア定義ファイルを作成しています。

```
write_hwdef -force C:/Data/ug940/lab1/zynq_design.hdf
```

関連項目

- [create_bd_design](#)

write_ibis

現在のフロアプランの IBIS モデルを書き出します。

構文

```
write_ibis [-force] [-allmodels] [-nopin] [-no_pin_mapping]
           [-truncate <arg>] [-component_name <arg>] [-ibs <arg>] [-pkg <arg>]
           [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
[-force]	既存の .ibs ファイルを上書きします。
[-allmodels]	アーキテクチャで使用可能なバッファ モデルをすべて含みます。デフォルトでは、フロアプランで使用されるバッファ モデルのみが含まれます。
[-nopin]	パッケージ (ダイパッドからパッケージピンへのパス) のピンごとの記述を含めません。パッケージは、すべてのピンに適用される 1 つの RLC 伝送ラインモデルに削減され、[Package] セクションで定義されます。デフォルトでは、このオプションはオフになっています。IBISWriter は、パッケージのピンごとの記述を RLC メトリックとして [Define Package Model] セクションに含めます。
[-no_pin_mapping]	UltraScale、UltraScale+、および Versal の [Pin Mapping] セクションを出力しません。
[-truncate]	出力ファイルの信号名の最大長を指定します。最大長を超えるとその部分は表示されません。このオプションを使用すると、信号名を 20、40、または 0 (無制限) で切り捨てることができます。デフォルトでは、IBIS バージョン 4.2 の仕様に準拠するよう信号名は 40 文字で切り捨てられます。
[-component_name]	デフォルトの代わりに複数の FPGA デザインで使用する新しいコンポーネント名を指定します。
[-ibs]	アップデートされた汎用 IBIS モデル ファイルを指定します。
[-pkg]	アップデートされたピンごとの寄生パッケージ データ ファイルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	出力ファイル名を指定します。拡張子 .ibs はオプションです。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

現在のデザインのターゲット デバイスの IBIS モデルを書き出します。デザインからのネットリストおよびインプリメンテーションの詳細と使用可能なピンごとの寄生パッケージ情報がまとめられ、デザイン専用のカスタム IBIS ファイルが作成されます。

`write_ibis` コマンドではデザイン情報が IBIS モデルにまとめられるので、このコマンドを実行するには RTL、ネットリスト、またはインプリメント済みデザインが開いている必要があります。

引数

`-force` (オプション): 指定した名前の IBIS ファイルが存在する場合に上書きします。

`-allmodels` (オプション): ターゲット デバイスのバッファ モデルをすべてエクスポートします。デフォルトでは、デザインで使用されるバッファ モデルのみが記述されます。

`-nopin` (オプション): ダイ パッドからパッケージ ピンへのパスのピンごとのコード記述をオフにします。IBIS モデルには、[Package] セクションのすべてのピンが 1 本の RLC 伝送ラインで記述されます。デフォルトでは、データが存在する場合、パッケージのピンごとのモデリングが RLC メトリックとして [Define Package Model] セクションに含まれます。

`-no_pin_mapping` (オプション): UltraScale、UltraScale+、および Versal デバイスの [Pin Mapping] セクションを出力しません。

`-truncate <arg>` (オプション): 出力ファイルの信号名の最大長を指定します。最大長を超えるとその部分は表示されません。有効な値は、20、40、0 (制限なし) です。デフォルトでは、IBIS バージョン 4.2 の仕様に準拠するよう信号名は 40 文字に切り詰められます。

`-component_name <arg>` (オプション): デフォルト値 (デバイス ファミリ) を変更する新しいコンポーネント名を指定します。

`-ibs <arg>` (オプション): アップデートされた汎用 IBIS モデル ファイルを指定します。これは、ツールのインストールに含まれる parts ディレクトリの IBIS モデルを上書きするために使用します。このオプションは、インストール ディレクトリに汎用モデルのないパーツすべてで必要となります。

`-pkg <arg>` (オプション): アップデートされたピンごとの寄生パッケージ データ ファイルを指定します。これは、ツールのインストール ディレクトリに含まれる parts ディレクトリの寄生パッケージ ファイルを上書きするために使用します。このオプションは、インストール ディレクトリに汎用モデルのないパーツすべてで必要となります。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する IBIS ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、ターゲット デバイスのバッファ モデルすべてがエクスポートされ、信号名が切り詰められないよう設定され、書き込むファイル名およびパスが指定されています。

```
write_ibis -allmodels -truncate 0 C:/Data/FPGA_Design/ibisOut.txt
```

write_inferred_xdc

推論された XDC タイミング制約を含むファイルを記述します。

構文

```
write_inferred_xdc [-force] [-all] [-append] [-async_clocks]
                  [-all_async_reg] [-clock_groups] [-clocks] [-excl_clocks] [-exceptions]
                  [-io_constraints] [-merge_existing_constraints] [-name <arg>] [-quiet]
                  [-verbose] [<file>]
```

使用法

名前	説明
[-force]	既存のファイルを上書きします。
[-all]	不足しているクロック以外のすべての制約を生成します。不足しているクロックは、-clocks オプションを使用すると生成されます。
[-append]	制約をファイルの最後に追加します。上書きはしません。
[-async_clocks]	非同期クロック グループを検索します。
[-all_async_reg]	安全および危険なクロック乗せ換えに不足している ASYNC_REG プロパティを検出します。
[-clock_groups]	非同期および排他的なクロック グループを検索します。-async_clocks、-excl_clocks オプションと同等です。
[-clocks]	不足しているクロック定義を検索します。
[-excl_clocks]	論理的、物理的に排他的なクロック グループを検索します。
[-exceptions]	不足している例外を検索します。
[-io_constraints]	不足している入力および出力遅延を検索します。
[-merge_existing_constraints]	既存のユーザー定義制約を生成された制約に追加します。
[-name]	Vivado IDE で実行している場合にレポートされた制約を表示する結果の名前を指定します。ほかのオプションと共に使用できます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	制約を記述するファイルの名前を指定します。

カテゴリ

[FileIO](#) (ファイル入力および出力)、[Timing](#) (タイミング)

説明

write_inferred_xdc コマンドは、開いている合成済みまたはインプリメント済みデザインで定義する必要のある制約を検索するのに使用できます。既存の XDC ファイルに定義されてデザインに追加されたタイミング制約ではなく、Vivado タイミング エンジンで自動的に生成されたタイミング制約を記述します。

まず `write_inferred_xdc -clocks` を実行して、推奨されるクロックおよび生成クロック制約を定義します。推奨されるクロック制約は、1 ns の周期で定義されます。推奨される制約を編集して、デザインの要件に合ったクロック周期のクロックおよび生成クロックを作成します。

編集した制約ファイルは、`read_xdc` または `add_files`、および `update_timing` コマンドを使用してデザインに追加できます。

完全にクロックが指定されたデザインから推論されたタイミング制約すべてを記述するには、`-clock_groups` や `-async_clocks` などのさまざまなオプションを使用して `write_inferred_xdc` コマンドを複数回実行することが必要な場合があります。推論された制約を記述して読み込むプロセスを反復実行し、推論された制約すべてをキャプチャできます。下の例を参照してください。

このコマンドを実行すると、正常に実行された場合は実行されたプロセスが返され、正常に実行されなかった場合はエラーが返されます。

引数

`-force` (オプション): 指定の出力ファイルが存在する場合に上書きします。

`-all` (オプション): 現在のデザインの不足しているクロックを除くすべての XDC 制約を記述します。不足しているクロックを取得するには、`-clocks` オプションを使用します。

`-append` (オプション): コマンドの出力を指定したファイルに上書きするのではなく追加します。

注記: `-append` オプションは、`-file` オプションを使用している場合にのみ使用可能です。

`-async_clocks` (オプション): `set_clock_groups -asynchronous` 制約を使用して定義する必要のある非同期クロック グループを検索します。

`-all_async_reg` (オプション): 安全および危険なクロック乗せ換えに不足している `ASYNC_REG` プロパティを検出します。

`-clock_groups` (オプション): 非同期で排他的なクロック グループを検索します。同じコマンドで `-async_clocks` および `-excl_clocks` オプションを指定するのと同じです。

`-clocks` (オプション): `create_clock` および `create_generated_clock` 制約を使用して定義する必要のある不足しているクロックと生成クロック定義を検索します。



ヒント: ほかの制約オプションを指定しない場合、これが `write_inferred_xdc` コマンドで生成されるデフォルトのレポートです。

`-excl_clocks` (オプション): `set_clock_groups -physically-exclusive` 制約または `set_clock_groups -logically-exclusive` 制約を使用して定義する必要のある物理的および論理的に排他的なクロック グループを検索します。

`-exceptions` (オプション): デザインのタイミング パスのデフォルトを変更する `set_false_path` または `set_multicycle_path` などのタイミング制約で定義する必要のある不足しているタイミング例外を検索します。

`-io_constraints <arg>` (オプション): `set_input_delay` や `set_output_delay` などの不足している I/O 制約を検索します。

`-merge_existing_constraints` (オプション): `write_inferred_xdc` コマンドで現在レポートされている推論された制約のタイプに一致する既存のユーザー定義の制約をレポートします。



ヒント: 既存の制約は指定のファイルにコメントとして記述されるので、Vivado Design Suite に `read_xdc` コマンドを使用して既存の制約と競合することなく推論された制約ファイルを読み込むことができます。

`-name <arg>` (オプション): Vivado IDE で実行している場合にレポートされた制約を表示する結果の名前を指定します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 推論された XDC 制約を記述するファイルの名前を指定します。`write_inferred_xdc` コマンドではファイル拡張子は自動的に追加されないため、ファイル名の一部としてファイル拡張子を指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のデザインで推論されるクロック制約を記述しています。

```
write_inferred_xdc -clocks C:/Data/design1_inferred_clocks.xdc
```

次の例では、`write_inferred_xdc` コマンドを複数回実行して推論された制約すべてをキャプチャしています。

```
write_inferred_xdc -clocks clocks.xdc
source clocks.xdc
write_inferred_xdc -all all.xdc
source all.xdc
write_inferred_xdc -async_clocks async.xdc
source async.xdc
```

関連項目

- [create_clock](#)
- [create_generated_clock](#)
- [report_exceptions](#)
- [report_timing](#)
- [set_clock_groups](#)
- [set_false_path](#)
- [set_input_delay](#)
- [set_multicycle_path](#)
- [set_output_delay](#)
- [write_xdc](#)

write_ip_tcl

指定の IP を再作成する Tcl スクリプトを生成します。

構文

```
write_ip_tcl [-force] [-no_ip_version] [-ip_name <arg>] [-show_defaults]
             [-multiple_files] [-quiet] [-verbose] [<objects>] [<tcl_filename>...]
```

戻り値

IP の Tcl ファイル

使用法

名前	説明
[-force]	既存のファイルを上書きします。
[-no_ip_version]	create_ip コマンドの IP VLVN の一部として IP バージョンを含めないことを指定します。メジャー IP バージョンが変更されている場合に影響がある可能性があるので、注意してください。
[-ip_name]	IP の名前を指定します。このオプションでは、1 つの IP のみを指定できます。
[-show_defaults]	IP のデフォルトのパラメーター値を含むコンポーネントを追加します。
[-multiple_files]	引数として指定した各 IP の .tcl ファイルを作成します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<objects>]	Tcl を生成する IP を get_ips <instance name> コマンドを使用して指定します。
[<tcl_filename>]	生成する Tcl ファイルのファイル パスを指定します。パスがディレクトリで複数の IP を指定している場合は、各 IP のファイルが作成されます。デフォルトは ./ です。

カテゴリ

Object (オブジェクト)、Project (プロジェクト)、IPFlow (IP フロー)

write_iphys_opt_tcl

iPhysOpt スクリプトを記述します。

構文

```
write_iphys_opt_tcl [-place] [-binary] [-quiet] [-verbose] [<output>]
```

使用法

名前	説明
[-place]	配置情報を記述します。
[-binary]	バイナリ フォーマットで記述します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<output>]	iPhysOpt スクリプトを出力する Tcl ファイルを指定します。

カテゴリ

[Tools \(ツール\)](#)

説明

物理最適化には配置後にのみ提供されるタイミング データが必要なので、phys_opt_design コマンドは配置前には実行できません。ただし、インタラクティブ物理最適化機能 (iphys_opt_design) を使用すると、配置後のデザインに実行された物理最適化を記述し、それを配置前のデザイン ネットリストに適用できます。インタラクティブ物理最適化の詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』 (UG904) を参照してください。

インタラクティブ物理最適化は、次の 2 つの方法で使用できます。

- 全体的な配置結果とデザイン パフォーマンスを向上するため、配置前のネットリストに配置後の物理最適化を適用する。
- 物理最適化を必要に応じて繰り返すことができるように Tcl スクリプトに保存する。

write_iphys_opt_tcl コマンドは、実際に物理最適化が実行された配置後のデザインに対してのみ実行可能です。



ヒント: report_phys_opt コマンドを使用すると、デザインに対して実行された物理最適化をレポートできます。

出力は、phys_opt_design コマンドで実行された最適化をリストした iphys_opt_design コマンドのシーケンスを含む Tcl スクリプト ファイルです。iphys_opt Tcl スクリプトを編集して、実行された指定の最適化を変更できます。Tcl スクリプトは、配置後のデザインに実行された物理最適化の日時を含む履歴となります。



重要: iphys_opt Tcl スクリプトには、phys_opt_design コマンドで実行される特定の最適化が含まれますが、配置および配線の変更や結果は含まれません。

このコマンドが正常に実行された場合は何も返されず、正常に実行されなかった場合はエラーが返されます。

引数

`-place` (オプション): 物理最適化 Tcl コマンドと共に、デザインの最適化されたセルの配置データを記述します。デフォルトでは、`iphys_opt` Tcl スクリプトには配置データは含まれません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<output>` (必須): 記述するインタラクティブ物理最適化 Tcl ファイルの名前を指定します。ファイルのパス、名前、および拡張子を指定する必要があります。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のデザインで実行された物理最適化を指定の Tcl スクリプトに記述しています。

```
write_iphys_opt_tcl C:/Data/myDesign_physopt.tcl
```

関連項目

- [iphys_opt_design](#)
- [phys_opt_design](#)
- [read_iphys_opt_tcl](#)
- [report_phys_opt](#)

write_mem_info

デザインのメモリ マップ情報を .mmi ファイルに記述します。

構文

```
write_mem_info [-force] [-quiet] [-verbose] <file>
```

戻り値

.mmi ファイル名

使用法

名前	説明
[-force]	既存のメモリ情報 XML ファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	デザイン メモリ情報ファイルを指定します。英数字を使用した名前に拡張子 .mmi を付けて指定します。

カテゴリ

FileIO (ファイル入力および出力)

説明

デザインのメモリ マップを作成するための BRAM の配置とアドレス範囲を定義するメモリ マップ情報 (MMI) ファイルを記述します。



重要: write_mem_info コマンドを実行する際は、適切なプログラムに必要な BRAM の配置データとアドレス範囲がメモリ情報に含まれるようにするため、インプリメント済みデザインを開いておく必要があります。

write_mem_info コマンドで記述される MMI ファイルは、サイリンクス デバイス上の各ブロック RAM がアドレスブロックという連続アドレス空間にどのようにグループ化されているかを記述するテキスト ファイルです。

MMI ファイルにはブロック メモリ マップ (BMM) ファイルと同様のメモリ マップ情報が含まれますが、ビットストリーム (BIT ファイルと結合するために updatemem で読み込み可能なフォーマットで記述されています。

updatemem は、MMI ファイルを使用して、特定のアドレス範囲にマップする物理的な BRAM リソースを特定します。updatemem の実行の詳細は、『Vivado Design Suite ユーザー ガイド: エンベデッド プロセッサ ハードウェア デザイン』 (UG898) を参照してください。

このコマンドを実行すると、作成されたファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

-force (オプション): 同じ名前の MMI ファイルが存在する場合に上書きします。このオプションを指定しない場合、既存のファイルは上書きされません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述する MMI ファイルの名前を指定します。指定するファイル名には、パスとファイル拡張子を含めることができます。拡張子を指定しない場合は、デフォルトの拡張子 `.mmi` が使用されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、開いているデザインから MMI ファイルを作成しています。

```
write_mem_info C:/Data/design1.mmi
```

関連項目

- [write_bmm](#)

write_peripheral

ペリフェラル コンポーネントをディスクに保存します。

構文

```
write_peripheral [-quiet] [-verbose] <peripheral>
```

使用法

名前	説明
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><peripheral></code>	ペリフェラル オブジェクトを指定します。

カテゴリ

[Project \(プロジェクト\)](#)、[IPFlow \(IP フロー\)](#)、[CreatePeripheral \(ペリフェラルの作成\)](#)

説明

指定した AXI ペリフェラル オブジェクトを `component.xml` ファイルの形式でディスクに記述します。ペリフェラルは、`create_peripheral` コマンドで指定した IP リポジトリの指定したの名前の下に保存されます。

引数

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<peripheral>` (必須): 記述するペリフェラル オブジェクトを指定します。ペリフェラルは `create_peripheral` コマンドで作成し、このコマンドおよびその他の関連コマンドで処理しやすいように Tcl 変数に格納してください。例を参照してください。

例

次の例では、VLNV 属性を指定して新しい AXI ペリフェラルを作成し、そのペリフェラル オブジェクトを後の処理用に Tcl 変数に格納して、そのペリフェラルに AXI スレーブ インターフェイスを追加し、出力ファイルを生成して、component.xml ファイルをディスクに記述しています。新しいペリフェラルのディレクトリが現在のファイルセットの IP_REPO_PATH プロパティに追加され、IP カタログに新しいペリフェラルが追加されます。

```
set_perifObj [ create_peripheral {myCompany.com} {user} {testAXI1} \
    {1.3} -dir {C:/Data/new_periph} ]
add_peripheral_interface {S0_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
add_peripheral_interface {S1_AXI} -interface_mode {slave} \
    -axi_type {lite} $perifObj
generate_peripheral -driver -bfm_example_design \
    -enable_interrupt $perifObj
write_peripheral $perifObj
set_property ip_repo_paths C:/Data/new_periph [current_fileset]
update_ip_catalog -rebuild
```

関連項目

- [add_peripheral_interface](#)
- [create_peripheral](#)
- [generate_peripheral](#)

write_project_tcl

現在のプロジェクトを再作成する Tcl スクリプトをエクスポートします。

構文

```
write_project_tcl [-paths_relative_to <arg>] [-origin_dir_override <arg>]
  [-target_proj_dir <arg>] [-force] [-all_properties] [-no_copy_sources]
  [-no_ip_version] [-absolute_path] [-dump_project_info] [-use_bd_files]
  [-internal] [-quiet] [-verbose] <file>
```

戻り値

このコマンドが正常に実行された場合は true (0)、エラーが発生した場合は false (1)

使用法

名前	説明
<code>[-paths_relative_to]</code>	ソース ファイルの相対パスの基準ディレクトリを変更します。デフォルトは、スクリプトの出力ディレクトリ パスです。
<code>[-origin_dir_override]</code>	origin_dir ディレクトリ変数を指定します。デフォルトは、-paths_relative_to オプションで指定された値です。
<code>[-target_proj_dir]</code>	プロジェクトを復元するディレクトリを指定します。デフォルトは現在のプロジェクト ディレクトリ パスです。
<code>[-force]</code>	既存の Tcl スクリプト ファイルを上書きします。
<code>[-all_properties]</code>	プロジェクト オブジェクトのすべてのプロパティ (デフォルトおよびデフォルトでないもの) を記述します。
<code>[-no_copy_sources]</code>	元のプロジェクトのローカルであってもソースをインポートしません。デフォルトは 1 です。
<code>[-no_ip_version]</code>	create_bd_cell コマンドの IP VLVN の一部として IP バージョンを含めません。デフォルトは 1 です。
<code>[-absolute_path]</code>	すべてのファイル パスを絶対パスにします。
<code>[-dump_project_info]</code>	オブジェクトの値を記述します。
<code>[-use_bd_files]</code>	BD を作成するプロシージャを記述するのではなく、BD のソース ファイルを直接使用します。
<code>[-internal]</code>	生成された Tcl スクリプトに基本的なヘッダー情報を記述します。
<code>[-quiet]</code>	コマンドをメッセージを表示せずに実行します。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	生成する Tcl スクリプト ファイルの名前を指定します。

カテゴリ

[xilinx_tclstore](#) (ザイリンクス Tcl Store)、[projutils](#) (プロジェクト ユーティリティ)

説明

現在のプロジェクトを再作成する Tcl スクリプトを生成します。

生成されるスクリプトには、プロジェクトの作成、プロジェクト タイプの設定、ファイルセットの作成、ソース ファイルの追加/インポート、run および run プロパティの定義を実行する Tcl コマンドが含まれます。



重要: 新しいプロジェクトは、Tcl スクリプトが読み込まれた現在の作業ディレクトリ (CWD) に作成されます。write_project_tcl コマンドで生成されたスクリプトは、生成したディレクトリから実行する必要があります。スクリプトを別のディレクトリから実行する場合は、Tcl シェルで set コマンドを使用して <origin_dir_loc> 変数とその別のディレクトリに設定するか、スクリプト内で <origin_dir> 変数を定義して、CWD とスクリプトで参照されているソース ファイルの相対パスが維持されるようにする必要があります。

この Tcl プロジェクト スクリプトとさまざまなデザイン ソースを、ソース ファイル管理およびプロジェクトのアーカイブ用にバージョン管理システムに保存できます。

引数

-paths_relative_to <arg> (オプション): ソース ファイルの相対パスの origin_dir 変数を変更します。

-origin_dir_override <arg> (オプション): origin_dir ディレクトリ変数を指定の値に設定します。デフォルトは、-paths_relative_to オプションで指定した値です。

-target_proj_dir <arg> (オプション): プロジェクトを再作成するディレクトリ パスを指定します。このオプションで指定したディレクトリ パスで create_project コマンドが記述されます。デフォルトは現在のプロジェクト ディレクトリです。

-force (オプション): 指定したプロジェクト スクリプト ファイルが存在する場合に上書きします。スクリプト ファイルが存在する場合に -force が指定されていないと、エラー メッセージが表示されます。

-all_properties (オプション): プロジェクトのすべてのプロパティを記述します。デフォルトでは、デフォルトでないプロパティのみが記述されます。プロジェクト、ファイルセット、ファイル、run などのすべてのオブジェクトに対して set_property コマンドが記述されます。



ヒント: -all_properties オプションを指定しない場合、デフォルトでないプロパティのみがスクリプトに記述されます。

-no_copy_sources (オプション): 元のプロジェクトにローカルであっても、ソースをインポートしません。元のプロジェクトのローカルにあるファイルは新しいプロジェクトにインポートされません。



重要: このオプションを使用する場合、ブロック図を含むデザインに -use_bd_files オプションを指定する必要があります。

-absolute_path (オプション): 元のプロジェクト ディレクトリに対してすべてのファイル パスを絶対パスにします。

-dump_project_info (オプション): すべてのオブジェクトの現在の値と、そのデフォルト値を示すファイルを作成します。これらのファイルは、作成し直したプロジェクトで問題をデバッグするのに役立ちます。

-use_bd_files (オプション): ブロック図を再作成する Tcl スクリプトを記述するのではなく、Vivado IP インテグレーターからのブロック図ソース (.bd) を直接使用します。デフォルトでは、write_project_tcl コマンドでされるプロジェクト Tcl スクリプト内にブロック図を再作成する Tcl プロシージャが含まれます。-use_bd_files オプションを使用すると、ブロック デザイン ソースがコピーされるか、-no_copy_sources オプションが使用されている場合はほかのデザイン ソースと同様に参照されます。

-internal (オプション): 生成された Tcl スクリプトに基本的なヘッダー情報を含めます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): `write_project_tcl` コマンドで作成するスクリプト ファイルの名前を指定します。ファイル拡張子を指定しない場合、`.tcl` が付けられます。

例

次の例では、現在のプロジェクトに対して `recreate.tcl` という Tcl スクリプトを生成しています。

```
write_project_tcl recreate.tcl
```

次の例では、`recreate.tcl` という現在のプロジェクトの Tcl スクリプトを `./script` にエクスポートし、`create_project` コマンド用に `/tmp/test` ディレクトリを指定します。Vivado Tcl シェルで `recreate.tcl` スクリプトを実行すると、プロジェクトが `/tmp/test` ディレクトリに再作成されます。

```
write_project_tcl -target_proj_dir "/tmp/test" ./script/recreate.tcl
```

次の例では、現在のプロジェクトの Tcl スクリプトを生成し、デフォルト値のものおよびデフォルト値でないものを含め、すべてのプロパティを記述しています。

```
write_project_tcl -all_properties recreate.tcl
```

次の例では、現在のプロジェクトの Tcl スクリプトを生成していますが、このプロジェクトにローカルのファイルはインポートしません。再作成されたプロジェクトでは、これらのファイルが参照されます。

```
write_project_tcl -no_copy_sources -use_bd_files recreate.tcl
```



重要: ブロック図を含むデザインでは、`-use_bd_files` オプションを使用した場合に `-no_copy_sources` オプションも使用する必要があります。

次の例では、現在の作業ディレクトリにある現在のプロジェクトの `recreate.tcl` スクリプトを生成し、`./my_test` ディレクトリにプロジェクトを再作成して、そのプロジェクトに含まれるファイルのリストを表示した後、そのプロジェクトを閉じています。

```
open_project ./test/test.xpr
write_project_tcl -force recreate.tcl
close_project
file mkdir my_test
cd my_test
source ../recreate.tcl
get_files -of_objects [get_filesets sources_1]
report_property [current_project]
close_project
```

次の例では、bft_test という新しいプロジェクトを作成し、ファイルを追加してファイルセット プロパティを設定し、現在の作業ディレクトリの bft.tcl という Tcl スクリプトを生成した後、./my_bft ディレクトリにプロジェクトを再作成して、そのプロジェクトのファイル (test_1.v および test_2.v) をリストし、verilog_define プロパティの値を表示して、そのプロジェクトを閉じています。

```
create_project bft_test ./bft_test
add_files test_1.v
add_files test_2.v
set_property verilog_define {a=10} [get_filesets sources_1]
write_project_tcl -force bft.tcl
close_project
file mkdir my_bft
cd my_bft
source ../bft.tcl
get_files -of_objects [get_filesets sources_1]
get_property verilog_define [get_filesets sources_1]
close_project
```

関連項目

- [add_files](#)
- [archive_project](#)
- [close_project](#)
- [create_project](#)
- [current_project](#)
- [get_files](#)
- [get_property](#)
- [open_project](#)
- [report_property](#)
- [set_property](#)

write_qor_suggestions

QoR 推奨項目をファイルに記述します。

構文

```
write_qor_suggestions [-strategy_dir <arg>] [-tcl_output_dir <arg>]
                      [-force] [-of_objects <args>] [-quiet] [-verbose] <file>
```

使用法

名前	説明
<code>[-strategy_dir]</code>	ストラテジ RQS および Tcl ファイルを作成するディレクトリを指定します。ディレクトリを指定すると、各ストラテジに対して 1 つの RQS ファイルと複数の Tcl ファイルが保存されます。
<code>[-tcl_output_dir]</code>	Tcl ファイルを作成するディレクトリを指定します。QoR 推奨項目の Tcl ファイルはこのオプションで指定したディレクトリに生成されます。
<code>[-force]</code>	既存の推奨項目ファイルを上書きします。
<code>[-of_objects]</code>	QoR 推奨項目オブジェクトのリストを指定します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	QoR 推奨項目ファイルを指定します。英数字を使用した名前に拡張子 <code>.rps</code> を付けて指定します。

カテゴリ

FileIO (ファイル入力および出力)、Feasibility (設計実現可能性)、Timing (タイミング)

説明

`report_qor_suggestions` コマンドで生成された QoR 推奨項目をファイルに記述します。最新のレポートからの推奨項目と、`read_qor_suggestions` コマンドでデザインに読み込んだ推奨項目を 1 つの RQS ファイルにまとめることができます。

特定の QoR 推奨項目を記述するには、`-of_objects` オプションを使用します。このオプションを指定しない場合、すべての推奨項目が記述されます。

推奨項目の管理には、RQS オブジェクトを使用することをお勧めしますが、コマンドは Tcl ファイルを使用して表示および実行できます。`-tcl_output_dir` オプションを使用すると、プロパティ ベースの自動推奨項目が Tcl スクリプトに記述されます。

デザインの解析に機械学習を使用するインプリメンテーション ストラテジは、`report_qor_suggestions` コマンドを使用して生成できます。`-strategy_dir` オプションを指定すると、各ストラテジに対して複数の Tcl ファイルと 1 つの RQS ファイルが生成されます。Tcl ファイルは、プロジェクト フローまたは非プロジェクト フローへの統合に便利です。推奨項目はストラテジ情報と共に `run` 特定のファイルに含まれているので、メインの RQS ファイルは使用しないでください。

このコマンドを実行すると、作成された出力ファイルの名前が返されるか、正常に実行されなかった場合はエラーが返されます。

引数

`-strategy_dir <arg>` (オプション): Tcl ファイルと機械学習に基づくインプリメンテーション `run` を設定するコマンドを含む RQS ファイルを作成します。

`-tcl_output_dir <arg>` (オプション): XDC コマンドを含む Tcl スクリプトを作成します。

`-force` (オプション): 指定のファイルが存在する場合に上書きします。

`-of_objects <args>` (オプション): 記述する QoR 推奨項目オブジェクトを `get_qor_suggestions` コマンドを使用して指定します。このオプションを使用すると、特定の QoR 推奨項目をプロパティに基づいて記述できます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): QoR 推奨項目を記述するファイルを指定します。ファイル名は `.rqs` 拡張子を付けて指定する必要があります。指定したファイルが存在する場合、`-force` オプションが指定されていない場合はエラーが返されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例は、QoR 推奨項目をレポートし、指定のファイルに記述します。

```
report_qor_suggestions
write_qor_suggestions C:/Data/qor_results.rqs
```

次の例は、QoR 推奨項目をレポートし、指定のファイルに記述します。

```
report_qor_suggestions
write_qor_suggestions -strategy_dir C:/Data/strategy_dir C:/Data/
qor_suggestions.rqs
```

プロジェクト モードでは、プロジェクト モード用に生成した Tcl スクリプトを読み込む必要があります。`run` ごとに 1 つのスクリプトがあります。非プロジェクト スクリプトの例も示します。

関連項目

- [get_qor_suggestions](#)
- [read_qor_suggestions](#)
- [report_qor_suggestions](#)

write_schematic

回路図をエクスポートします。

構文

```
write_schematic [-force] [-format <arg>] [-orientation <arg>]  
                [-scope <arg>] [-name <arg>] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名

使用法

名前	説明
[-force]	既存のファイルを上書きします。
[-format]	フォーマット指定します。有効な値は native または pdf です。ネイティブ フォーマットは、read_schematic を使用して表示できます。デフォルトは native です。
[-orientation]	回路図の向きを指定します。有効な値は landscape または portrait です。
[-scope]	エクスポートする範囲を指定します。有効な値は current_page、visible、all で、デフォルトは current_page です。
[-name]	[Schematic] ウィンドウのタイトルを指定します。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	出力ファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)

説明

Vivado IDE で現在開いている [Schematic] ウィンドウまたは指定した [Schematic] ウィンドウをファイルに出力します。

ファイルは、read_schematic コマンドを使用して Vivado IDE に読み込み可能なネイティブ ASCII ファイルとして記述するか、Vivado Design Suite 外で使用可能な PDF または SVG ファイルとして記述できます。これは、IP パッケージャー フローまたは Vivado IP インテグレーターからの IP コアを記録する場合に有益です。

引数

-force (オプション): 指定の回路図ファイルが存在する場合に上書きします。

`-format [native | pdf | svg]` (オプション): ファイルを出力するフォーマットを指定します。デフォルトのファイルフォーマットは `native` (Vivado Design Suite のネイティブ フォーマット) で、`read_schematic` コマンドを使用してツールに読み込むことができます。SVG は Scalable Vector Graphics (スケーラブル ベクター グラフィックス) フォーマットです。PDF は Portable Document Format (ポータブル ドキュメント フォーマット) です。

注記: `-format` オプションでは、大文字/小文字が区別されます。

`-orientation [landscape | portrait]` (オプション): 回路図を縦長 (`portrait`) または横長 (`landscape`) のどちらの向きで記述するかを指定します。デフォルトは `portrait` です。

`-scope [current_page | visible | all]` (オプション): 回路図の現在のページ (`current_page`)、[Schematic] ウィンドウで表示されている部分 (`visible`)、または回路図すべて (`all`) を記述するかを指定します。デフォルトは `current_page` です。

`-name <arg>` (オプション): ファイルに記述する、開いている [Schematic] ウィンドウの名前を指定します。複数の [Schematic] ウィンドウが開いている場合に、このオプションを使用して記述する [Schematic] ウィンドウを指定します。

注記: 複数の [Schematic] ウィンドウが開いていて `-name` を指定しない場合、最初に開いたウィンドウが記述されます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述する回路図ファイルのパスと名前を指定します。パスの指定はオプションですが、パスを指定しない場合は、現在の作業ディレクトリまたは Vivado ツールを起動したディレクトリに回路図ファイルが作成されます。

例

次の例では、Vivado IDE に表示されている指定の [Schematic] ウィンドウを、Vivado Design Suite に読み込み可能なネイティブ フォーマットで記述しています。ファイルが存在する場合は、上書きします。

```
write_schematic -name Schematic C:/Data/mySchematic.txt -force
```

次の例では、指定の [Schematic] ウィンドウを横長の PDF ファイルに記述しています。

```
write_schematic -format pdf -name "Schematic (2)" C:/Data/mySchematic.pdf \
-orientation landscape
```

関連項目

- [read_schematic](#)
- [write_bd_layout](#)

write_sdf

イベント シミュレーション用のフラットな SDF 遅延ファイルを生成します。

構文

```
write_sdf [-process_corner <arg>] [-cell <arg>] [-rename_top <arg>]
          [-force] [-mode <arg>] [-gzip] [-quiet] [-verbose] <file>
```

使用法

名前	説明
<code>[-process_corner]</code>	SDF 遅延の必要とされるプロセス コーナーを指定します。有効な値は slow、fast で、デフォルトは slow です。
<code>[-cell]</code>	書き込むデザインのルートを指定します (例: des.subblk.cpu)。デフォルトはデザイン全体です。
<code>[-rename_top]</code>	最上位モジュール名を netlist などのカスタム名に置き換えます。デフォルトは、新しい最上位モジュールの名前です。
<code>[-force]</code>	既存の SDF ファイルを上書きします。
<code>[-mode]</code>	SDF のモードを指定します。有効な値は sta (スタティック タイミング解析) または timesim (タイミング シミュレーション) で、デフォルトは timesim です。
<code>[-gzip]</code>	SDF ファイルを GZIP で圧縮します。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><file></code>	ファイル名を指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Feasibility Simulation \(シミュレーション\)](#)、[Timing \(タイミング\)](#)

説明

デザインのセルのタイミング遅延を標準遅延フォーマット (SDF) ファイルに記述します。

出力される SDF ファイルは、`write_verilog` コマンドでを使用して、スタティック タイミング解析およびタイミング シミュレーション用の Verilog ネットリストを作成できます。

引数

`-process_corner [fast | slow]` (オプション): 指定したプロセス コーナーの遅延を記述します。遅延は fast プロセス コーナーよりも slow プロセス コーナーの方が大きくなります。有効な値は、slow または fast です。デフォルトでは、SDF ファイルは slow プロセス コーナー用に記述されます。

`-cell <arg>` (オプション): デザイン階層の指定したセルから SDF ファイルを記述します。デフォルトでは、デザイン全体の SDF ファイルが作成されます。

`-rename_top <arg>` (オプション): 出力 SDF ファイルの最上位モジュールの名前を変更します。

`-force` (オプション): 同じ名前の既存の SDF ファイルを上書きします。

`-mode [timesim | sta]` (オプション): SDF ファイルを記述する際に使用するモードを指定します。有効な値は次のとおりです。

- `timesim`: タイミング シミュレーションで使用する SDF ファイルを出力します。これがデフォルト設定です。
- `sta`: スタティック タイミング解析 (STA) で使用する SDF ファイルを出力します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する SDF ファイルの名前を指定します。SDF ファイルは、`write_verilog` コマンドで `-sdf_anno` および `-sdf_file` オプションを使用することにより Verilog ネットリストで参照されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、指定したディレクトリに SDF ファイルが生成されます。

```
write_sdf C:/Data/FPGA_Design/designOut.sdf
```

関連項目

- [write_verilog](#)

write_verilog

現在のネットリストを Verilog 形式でエクスポートします。

構文

```
write_verilog [-cell <arg>] [-mode <arg>] [-lib] [-port_diff_buffers]
              [-write_all_overrides] [-keep_vcc_gnd] [-rename_top <arg>]
              [-sdf_anno <arg>] [-sdf_file <arg>] [-force] [-include_xilinx_libs]
              [-logic_function_stripped] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名またはディレクトリ名

使用法

名前	説明
[-cell]	書き込むデザインのルートを指定します (例: des.subblk.cpu)。デフォルトはデザイン全体です。
[-mode]	有効な値は design、pin_planning、synth_stub、sta、funcsim、timesim で、デフォルトは design です。
[-lib]	各ライブラリを個別のファイルに記述します。
[-port_diff_buffers]	-mode が port に設定されている場合に差動バッファを出力します。
[-write_all_overrides]	パラメーター値がプリミティブのデフォルト値と同じであっても、デザインのすべてのパラメーター値を記述します。
[-keep_vcc_gnd]	VCC/GND インスタンスをロードのリテラル定数に置換しません。シミュレーション モードでのみ使用可能です。
[-rename_top]	最上位モジュール名を netlist などのカスタム名に置き換えます。デフォルトは、新しい最上位モジュールの名前です。
[-sdf_anno]	sdf_annotate システム タスク文を生成するかどうかを指定します。
[-sdf_file]	SDF ファイルの完全なパスを指定します。デフォルトは <file>.sdf です。
[-force]	既存のファイルを上書きします。
[-include_xilinx_libs]	シミュレーション モデルをライブラリにリンクするのではなく、直接ネットリストに含めます。
[-logic_function_stripped]	LUT および RAMB の INIT 文字列を固定の値に変換します。結果のネットリストは正しく機能しません。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	記述するファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Simulation \(シミュレーション\)](#)

説明

現在のデザインまたは指定したセルの Verilog ネットリストを指定したファイルまたはディレクトリに記述します。出力は IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。

デザインまたは指定のセルの完全なネットリスト、デザインのポート リスト、シミュレーションまたはスタティック タイミング解析用の Verilog ネットリストを出力できます。

引数

`-cell <arg>` (オプション): 指定したセルまたはデザイン階層のブロック レベルから VHDL ネットリストを記述します。出力される Verilog ファイルには、指定したセルまたはモジュール内の情報だけが含まれます。

`-mode <arg>` (オプション): Verilog ファイルを記述する際に使用するモードを指定します。デフォルトでは、デザイン全体の Verilog ネットリストが記述されます。有効なモードは、次のとおりです。

- `design`: デザイン全体の Verilog ネットリストを出力します。これは、すべての配置、インプリメンテーション、配線情報を含むデザインのスナップショットとして使用されます。
- `pin_planning`: デザインの最上位の I/O ポートのみを出力します。
- `synth_stub`: 合成スタブとして使用するデザインの最上位からのポートを出力します。
- `sta`: スタティック タイミング解析 (STA) に使用する Verilog ネットリストを出力します。
- `funcsim`: 論理シミュレーションに使用する Verilog ネットリストを出力します。この出力ネットリストは、合成には使用できません。
- `timesim`: タイミング シミュレーションに使用する Verilog ネットリストを出力します。この出力ネットリストは、合成には使用できません。

`-lib` (オプション): デザインで使用される各ライブラリに個別の Verilog ファイルを作成します。

注記: `-library` オプションは、シミュレーションにのみ使用可能です。Vivado 合成では、すべての Verilog ファイルがデフォルトの work ライブラリにあるとして処理されます。

`-port_diff_buffers` (オプション): 差動ペア バッファとそれらのバッファに関連付けられている内部ワイヤを出力ポート リストに追加します。このオプションは、`-mode pin_planning` または `-mode synth_stub` オプションを使用している場合にのみ有効です。

`-write_all_overrides [true | false]` (オプション): パラメーター値が定義されたプリミティブのデフォルト値と同じであっても、デザインのすべてのパラメーター 値を Verilog 出力に記述します。`false` に設定すると、パラメーター値がプリミティブのデフォルト値と同じである場合は Verilog ファイルに記述されません。このオプションを `true` に設定しても結果は変わりませんが、出力される Verilog ファイルがより詳細なものとなります。

`-keep_vcc_gnd` (オプション): デフォルトでは、シミュレーション用にネットリストを記述する際、または IP インテグレート ブロック デザインからネットリストを記述する際に、VCC と GND プリミティブおよびそれらを駆動するネットが、そのネット上の各ロードのリテラル定数に置換されます。`-keep_vcc_gnd` オプションを使用すると、このデフォルト動作がオフになり、VCC または GND プリミティブが保持されます。

`-rename_top <arg>` (オプション): 出力ファイルの最上位モジュールの名前を変更します。このオプションは `-mode funcsim` または `-mode timesim` を指定した場合にのみ有効で、Verilog ネットリストを最上位シミュレーション テストベンチに組み込むことができます。

`-sdf_anno [true | false]` (オプション): Verilog ネットリストに `$sdf_annotate` 文を追加します。有効な値は `true` (または 1) および `false` (または 0) です。このオプションは、`-mode timesim` を指定している場合にのみ有効で、デフォルトでは `false` に設定されています。

`-sdf_file <arg>` (オプション): 出力 Verilog ファイルに `$sdf_annotate` 文を記述する場合に使用する SDF ファイルのパスとファイル名を指定します。指定しない場合、SDF ファイルのパスと名前は file で指定した Verilog ファイルと同じになり、ファイル拡張子は `.sdf` になります。SDF ファイルは、`write_sdf` コマンドを使用して指定のファイルパスおよび名前でも別に記述する必要があります。

`-force` (オプション): 同じ名前の Verilog ファイルが存在する場合に上書きします。

`-include_xilinx_libs` (オプション): シミュレーション モデルをライブラリから参照するのではなく、出力ネットリスト ファイルに直接記述します。

`-logic_function_stripped` (オプション): LUT および RAM の INIT 値を固定の値に変換し、デバッグ用のネットリストを作成します。シミュレーションまたは合成では正しく機能しません。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、`TCL_OK` が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述する Verilog ファイルのパスと名前を指定します。パスの指定はオプションですが、パスを指定しない場合は、現在の作業ディレクトリまたは Vivado ツールを起動したディレクトリに Verilog ファイルが作成されます。

例

次の例では、デザイン全体の Verilog シミュレーション ネットリスト ファイルを指定したファイル名とパスで記述します。

```
write_verilog C:/Data/my_verilog.v
```

次の例では、`-mode timesim` と `-sdf_anno` オプションが指定されているので、Verilog ネットリストに `$sdf_annotate` 文が追加されます。`-sdf_file` オプションは指定されていないので、SDF ファイルの名前は Verilog ファイルと同じになり、ファイル拡張子は `.sdf` になります。

```
write_verilog C:/Data/my_verilog.net -mode timesim -sdf_anno true
```

注記: `$sdf_annotate` 文に記述される SDF ファイル名は `my_verilog.sdf` です。

次の例では、論理シミュレーション モードを指定し、出力シミュレーション ネットリストで VCC および GND プリミティブが保持されるように指定し、出力ファイルを指定して、Verilog ネットリストを記述しています。

```
write_verilog -mode funcsim -keep_vcc_gnd out.v
```

関連項目

- [write_sdf](#)
- [write_vhdl](#)

write_vhdl

現在のネットリストを VHDL 形式でエクスポートします。

構文

```
write_vhdl [-cell <arg>] [-mode <arg>] [-lib] [-port_diff_buffers]
           [-write_all_overrides] [-keep_vcc_gnd] [-rename_top <arg>] [-arch_only]
           [-force] [-include_xilinx_libs] [-quiet] [-verbose] <file>
```

戻り値

出力ファイル名またはディレクトリ名

使用法

名前	説明
[-cell]	書き込むデザインのルートを指定します (例: des.subblk.cpu)。デフォルトはデザイン全体です。
[-mode]	出力モードを指定します。有効な値は funcsim、pin_planning、synth_stub で、デフォルトは funcsim です。
[-lib]	各ライブラリを個別のファイルに記述します。
[-port_diff_buffers]	-mode が port に設定されている場合に差動バッファーを出力します。
[-write_all_overrides]	パラメーター値がプリミティブのデフォルト値と同じであっても、デザインのすべてのパラメーター値を記述します。
[-keep_vcc_gnd]	VCC/GND インスタンスをロードのリテラル定数に置換しません。シミュレーション モードでのみ使用可能です。
[-rename_top]	最上位モジュール名を netlist などのカスタム名に置き換えます。デフォルトは、新しい最上位モジュールの名前です。
[-arch_only]	アーキテクチャのみを記述し、最上位セルにエンティティ宣言を記述しません。
[-force]	既存のファイルを上書きします。
[-include_xilinx_libs]	シミュレーション モデルをライブラリにリンクするのではなく、直接ネットリストに含めます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<file>	記述するファイルを指定します。

カテゴリ

[FileIO \(ファイル入力および出力\)](#)、[Simulation \(シミュレーション\)](#)

説明

現在のデザインまたは指定したセルの VHDL ネットリストを指定したファイルまたはディレクトリに記述します。

出力は VHDL IEEE 1076.4 VITAL-2000 準拠の VHDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。デザインまたは指定のセルの完全なネットリスト、またはデザインのポート リストを出力できます。

引数

`-cell <arg>` (オプション): 指定したセルまたはデザイン階層のブロック レベルから VHDL ネットリストを記述します。出力される VHDL ファイルには、指定したセルまたはモジュール内の情報だけが含まれます。

`-mode <arg>` (オプション): VHDL ファイルを記述する際に使用するモードを指定します。デフォルトでは、デザイン全体のシミュレーション ネットリストが記述されます。有効なモードは、次のとおりです。

- `funcsim`: 論理シミュレーション モデルとして使用する VHDL ネットリストを出力します。この出力ネットリストは、合成には使用できません。これがデフォルト設定です。
- `pin_planning`: 最上位モジュールのエンティティ 宣言に含まれる I/O ポートのみを出力します。
- `synth_stub`: 合成スタブとして使用するデザインの最上位からのポートを出力します。

`-lib` (オプション): デザインで使用される各ライブラリに個別の VHDL ファイルを作成します。

注記: このオプションは、以前のリリースの `-nolib` オプションに置き換わるもので、動作は反対になります。以前は、`write_vhdl` を実行したときに、`-nolib` を使用しなければ、デザインで使用される各ライブラリにデフォルトで個別の VHDL ファイルが出力されていました。現在では、各ライブラリに個別の VHDL ファイルを生成するには `-lib` オプションを指定する必要があります。

`-port_diff_buffers` (オプション): 差動ペア バッファとそれらのバッファに関連付けられている内部ワイヤを出力ポート リストに追加します。このオプションは、`-mode pin_planning` または `-mode synth_stub` オプションを使用している場合にのみ有効です。

`-write_all_overrides [true | false]` (オプション): パラメーター値が定義されたプリミティブのデフォルト値と同じであっても、デザインのすべてのパラメーター 値を VHDL 出力に記述します。`false` に設定すると、パラメーター値がプリミティブのデフォルト値と同じである場合は VHDL ファイルに記述されません。このオプションを `true` に設定しても結果は変わりませんが、出力されるネットリスト ファイルがより詳細なものとなります。

`-keep_vcc_gnd` (オプション): デフォルトでは、シミュレーション用にネットリストを記述する際、または IP インテグレート ブロック デザインからネットリストを記述する際に、VCC と GND プリミティブおよびそれらを駆動するネットが、そのネット上の各ロードのリテラル定数に置換されます。`-keep_vcc_gnd` オプションを使用すると、このデフォルト動作がオフになり、VCC または GND プリミティブが保持されます。

`-rename_top <arg>` (オプション): 出力ファイルの最上位モジュールの名前を変更します。このオプションは `-mode funcsim` を指定した場合にのみ有効で、VHDL ネットリストを最上位シミュレーション テストベンチに組み込むことができます。

`-arch_only` (オプション): 最上位モジュールのエンティティ 定義を含めず、アーキテクチャのみを出力します。これにより、出力 VHDL ネットリストを別のテストベンチと使用する場合に、出力 VHDL ネットリストの使用が簡略化されます。

`-force` (オプション): 同じ名前の VHDL ファイルが存在する場合に上書きします。

`-include_xilinx_libs` (オプション): シミュレーション モデルをライブラリから参照するのではなく、出力ネットリスト ファイルに直接記述します。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 生成する VHDL ファイルの名前を指定します。ファイル拡張子 `.vhd` または `.vhdl` を指定しない場合、指定した名前はディレクトリであると判断され、VHDL ファイルの名前は最上位モジュールと同じになり、指定したディレクトリに出力されます。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、デザイン全体の VHDL シミュレーション ネットリスト ファイルを指定したファイル名とパスで記述します。

```
write_vhdl C:/Data/bft_top.vhd
```

次の例では、最上位モジュールのエンティティ 定義は VHDL ネットリストに含まれません。

```
write_vhdl C:/Data/vhdl_arch_only.vhd -arch_only
```

関連項目

- [write_verilog](#)

write_waivers

1 つまたは複数の DRC/METHODOLOGY/CDC メッセージ除外をコマンド形式で記述します。

構文

```
write_waivers [-type <arg>] [-objects <args>] [-return_string] [-force]
              [-quiet] [-verbose] [<file>]
```

使用法

名前	説明
[-type]	除外のタイプを指定します。有効な値は、ALL、DRC、METHODOLOGY、CDC です。
[-objects]	記述する DRC、METHODOLOGY、CDC 除外オブジェクトを指定します。
[-return_string]	レポートの結果を文字列オブジェクトとして返します。
[-force]	既存のファイルを上書きします。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	除外を記述するファイルを指定します。

カテゴリ

[Waiver \(除外\)](#)、[Object \(オブジェクト\)](#)

説明

除外を次のデザイン セッションでも使用できるように保存するには、write_waivers コマンドを使用して除外コマンドの XDC ファイルを作成し、デザインを開き直したときに read_xdc コマンドを使用して読み込みます。

引数

-type <arg> (オプション): ファイルに記述する除外のタイプを指定します。有効な値は DRC、METHODOLOGY、および CDC です。

-objects <args> (オプション): 記述する除外オブジェクトを指定します。除外オブジェクトは get_waivers コマンドを使用して指定できます。

-return_string (オプション): 出力を Tcl 文字列として返します。Tcl 文字列は、変数定義でキャプチャしたり、解析またはその他の処理が可能です。

-force (オプション): 除外をファイルに強制的に記述します。カウントが無効であるために除外は記述されないというメッセージが表示された場合に、このオプションを使用してファイルに記述できます。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

`<file>` (必須): 記述するファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、現在のデザインの除外をすべて記述しています。

```
write_waivers C:/Data/design_waivers.xdc
```

次の例では、DRC タイプの除外のみを記述しています。

```
write_waivers -type DRC C:/Data/drc_waivers.xdc
```

関連項目

- [create_waiver](#)
- [delete_waivers](#)
- [get_cdc_violations](#)
- [get_drc_violations](#)
- [get_methodology_violations](#)
- [get_waivers](#)
- [report_cdc](#)
- [report_drc](#)
- [report_methodology](#)
- [report_waivers](#)

write_xdc

制約をサイリックス デザイン制約 (XDC) ファイルに記述します。XDC ファイルのデフォルトのファイル拡張子は .xdc です。

構文

```
write_xdc [-no_fixed_only] [-constraints <arg>] [-cell <arg>] [-sdc]
          [-no_tool_comments] [-force] [-exclude_timing] [-exclude_physical]
          [-add_netlist_placement] [-logic_function_stripped] [-type <args>]
          [-quiet] [-verbose] [<file>]
```

使用法

名前	説明
[-no_fixed_only]	配置が固定されているかどうかにかかわらず、すべての配置をエクスポートします。デフォルトでは、固定された配置のみがエクスポートされます。
[-constraints]	無効と示されている制約を含めます。有効な値は valid、invalid、all で、デフォルトは valid です。
[-cell]	制約をエクスポートする階層セルを指定します。
[-sdc]	すべてのタイミング制約を SDC 互換フォーマットでエクスポートします。
[-no_tool_comments]	UCF から変換する際にツールで生成された詳細コメントを XDC に記述しません。
[-force]	既存のファイルを上書きします。
[-exclude_timing]	タイミング制約をエクスポートしません。
[-exclude_physical]	物理制約をエクスポートしません。
[-add_netlist_placement]	ネットリストの配置制約をエクスポートします。
[-logic_function_stripped]	以前に -logic_function_stripped オプションを使用して実行した write_edif に関連付けられている disable_timing 制約を記述します。
[-type]	エクスポートする制約のタイプを指定します。有効な値は、timing、io、misc、waiver、および physical です。指定しない場合、すべての制約がエクスポートされます。
[-quiet]	コマンド エラーを表示しません。
[-verbose]	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
[<file>]	制約を記述する XDC ファイルを指定します。

カテゴリ

[Timing \(タイミング\)](#)、[FileIO \(ファイル入力および出力\)](#)

説明

制約をサイリックス® デザイン制約ファイル (XDC) に記述します。XDC は、最上位からエクスポートするか (デフォルト)、指定の階層セルからエクスポートできます。



重要: `write_xdc` コマンドでは、制約がデザインで追加または実行された順序で指定のファイルに記述されます。

`write_xdc` コマンドを使用すると、無効な XDC 制約を書き出すことができ、制約の記述方法または適用方法の問題により Vivado Design Suite で無視された制約をレポートできます。これは、特定のデザインに適用されている制約ファイルをデバッグする際に有益です。

このコマンドは、UCF ファイルを含むデザインから XDC ファイルを作成するために使用できます。アクティブな制約ファイルセットからの制約はすべて、それらが複数ファイルからの制約であっても、XDC にエクスポートされません。



ヒント: `write_xdc` コマンドでは、すべての UCF 制約は XDC フォーマットに変換されず、UCF ベースのデザインを XDC に自動的に変換するためのものではありません。UCF 制約を XDC に移行する方法の詳細は、『ISE から Vivado Design Suite への移行ガイド』(UG911) を参照してください。

引数

`-no_fixed_only` (オプション): 配置が固定されているかどうかにかかわらず、すべての LOC を制約ファイルにエクスポートします。デフォルトでは、固定された LOC 制約のみが XDC ファイルに書き込まれます。固定された LOC はユーザーの割り当てた配置で、固定されていない LOC はツールで自動的に割り当てられた配置です。

`-constraints <arg>` (オプション): 有効 (valid)、無効 (invalid) と指定した制約、またはすべての制約 (valid と invalid の両方) をエクスポートします。デフォルトでは、有効な制約のみが XDC ファイルにエクスポートされますが、有効な値は VALID、INVALID、ALL です。

`-cell <arg>` (オプション): 制約を記述する現在のデザインの階層セルの名前を指定します。制約は、指定したセルに相対的に指定した XDC ファイルに書き込まれます。

注記: このオプションを指定する場合は、デザインを開いておく必要があります。

`-sdc` (オプション): 現在のデザインからタイミング制約のみを SDC と完全に互換性のあるファイルフォーマットでエクスポートします。その他の定義された制約は、エクスポートされません。

`-no_tool_comments` (オプション): XDC ファイルにツールで生成されたコメントを追加しません。

`-force` (オプション): 同じ名前のファイルが存在する場合に上書きします。

`-exclude_timing` (オプション): タイミング制約をエクスポートしません。生成された XDC ファイルには、物理制約のみが含まれます。

`-exclude_physical` (オプション): 物理制約をエクスポートしません。生成された XDC ファイルには、タイミング制約のみが含まれます。

`-add_netlist_placement` (オプション): ネットリスト ファイルで定義されている配置制約を XDC ファイルの一部として含めます。

`-type <arg>` (オプション): エクスポートする制約のタイプを指定します。有効な値は、timing、io、misc、waiver、および physical です。一度に複数のタイプを指定できます。タイプを指定しない場合、すべての制約がエクスポートされます。

`-quiet` (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

`-verbose` (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、`set_msg_config` コマンドで定義できます。

<file> (必須): 生成する XDC ファイルの名前を指定します。

注記: パスをファイル名の一部として指定しない場合は、現在の作業ディレクトリまたはツールを起動したディレクトリにファイルが作成されます。

例

次の例では、固定されているセルと固定されていないセルの両方を含む、有効な制約と無効な制約を指定のファイルに記述しています。

```
write_xdc -no_fixed_only -constraints all C:/Data/design.xdc
```

次の例では、固定されているセルと固定されていないセルの両方を含む、無効な制約のみを指定のファイルに記述しています。

```
write_xdc -constraints invalid C:/Data/bad_constraints.xdc
```

次の例では、ネットリストソース ファイルで定義されている配置制約を含めた、物理制約のみを記述しています。

```
write_xdc -exclude_timing -add_netlist_placement C:/Data/physical.xdc
```

関連項目

- [read_xdc](#)

xsim

シミュレーション用のシミュレーション スナップショットを読み込み、シミュレーション オブジェクトを返します。

構文

```
xsim [-view <args>] [-autoloadwcfg] [-runall] [-R] [-maxdeltaid <arg>]
      [-nolog] [-maxlogsize <arg>] [-onfinish <arg>] [-onerror <arg>]
      [-tclbatch <args>] [-t <args>] [-testplusarg <args>] [-vcdfile <arg>]
      [-vcdunit <arg>] [-wdb <arg>] [-tp] [-tl] [-nosignalhandlers]
      [-ieeewarnings] [-stats] [-scNoLogFile] [-sv_seed <arg>]
      [-protoinst <args>] [-cov_db_dir <arg>] [-cov_db_name <arg>]
      [-ignore_assertions] [-ignore_coverage] [-downgrade_error2info]
      [-downgrade_error2warning] [-downgrade_fatal2info]
      [-downgrade_fatal2warning] [-ignore_feature <args>]
      [-downgrade_severity <args>] [-quiet] [-verbose] <snapshot>
```

戻り値

現在のシミュレーション オブジェクト

使用法

名前	説明
[-view]	波形設定ファイルを開きます。このオプションを複数指定して複数のファイルを開くことができます。
[-autoloadwcfg]	<name>.wdb という名前の DB ファイルに対して、<name>#.wcfg という名前のすべての WCFG ファイルを自動的に開きます。-view オプションが使用されている場合は無視されます。
[-runall]	シミュレーションを最後まで実行し、終了します (「run -all; exit」を実行)。
[-R]	シミュレーションを最後まで実行し、終了します (「run -all; exit」を実行)。
[-maxdeltaid]	最大差異を指定します。同じ時間に最大シミュレーション ループ回数を超えると、エラーがレポートされます。デフォルトは 10000 です。
[-nolog]	xsim コマンドライン ツールとの互換性のため提供されているオプションで、無視されます。
[-maxlogsize]	ログ ファイルの最大サイズを MB で指定します。デフォルトは -1 (無制限) です。
[-onfinish]	シミュレーションが終了したときの動作を指定します。有効な値は quit および stop で、デフォルトは stop です。
[-onerror]	シミュレーション ランタイム エラーが発生した場合の動作を指定します。有効な値は quit および stop で、デフォルトは stop です。
[-tclbatch]	バッチ モード実行用の Tcl ファイルを指定します。
[-t]	バッチ モード実行用の Tcl ファイルを指定します。
[-testplusarg]	plusargs が \$test\$plusargs および \$value\$plusargs システム関数で使われるように指定します。
[-vcdfile]	VCD 出力ファイルを指定します。

名前	説明
<code>[-vcdunit]</code>	VCD 出力の単位を指定します。デフォルトは、エンジンの精度ユニットと同じです。
<code>[-wdb]</code>	波形データベース出力ファイルを指定します。
<code>[-tp]</code>	実行中のプロセスの階層名が表示されるようにします。
<code>[-t1]</code>	実行中の文の行番号が表示されるようにします。
<code>[-nosignalhandlers]</code>	信号ハンドラーを使用せずに実行し、セキュリティ ソフトウェアとの競合を回避します。
<code>[-ieeewarnings]</code>	VHDL IEEE 関数からの警告をイネーブルにします。
<code>[-stats]</code>	終了するときにメモリおよび CPU の統計を表示します。
<code>[-scNoLogFile]</code>	SystemC 出力を XSim 出力とは別にします。
<code>[-sv_seed]</code>	制約ランダム ステミュラスのシードを指定します。デフォルトは 1 です。
<code>[-protoinst]</code>	プロトコル解析の .protoinst ファイルを指定します。
<code>[-cov_db_dir]</code>	SystemVerilog 有効範囲 run ディレクトリ。有効範囲データは、<cov_db_dir>/xsim.covdb/<cov_db_name> ディレクトリに保存されます。デフォルトは ./ または同様の xelab オプションで設定されている値です。
<code>[-cov_db_name]</code>	SystemVerilog 有効範囲 run の名前。有効範囲データは、<cov_db_dir>/xsim.covdb/<cov_db_name> ディレクトリに保存されます。デフォルトは、スナップショット名または同様の xelab オプションで設定されている値です。
<code>[-ignore_assertions]</code>	実行時に SystemVerilog の同時処理アサート構文を無視します。
<code>[-ignore_coverage]</code>	実行時に SystemVerilog の機能バレッジを無視します。
<code>[-downgrade_error2info]</code>	SystemVerilog のメッセージの重要度をエラーから情報に下げます。
<code>[-downgrade_error2warning]</code>	SystemVerilog のメッセージの重要度をエラーから警告に下げます。
<code>[-downgrade_fatal2info]</code>	SystemVerilog のメッセージの重要度を致命的から情報に下げます。
<code>[-downgrade_fatal2warning]</code>	SystemVerilog のメッセージの重要度を致命的から警告に下げます。
<code>[-ignore_feature]</code>	実行時に無視する SystemVerilog の機能を指定します。有効な値は assertion で、同時処理アサートを無視します。
<code>[-downgrade_severity]</code>	SystemVerilog の HDL メッセージの重要度を下げます。有効な値は error2warning、error2info、fatal2warning、fatal2info です。
<code>[-quiet]</code>	コマンド エラーを表示しません。
<code>[-verbose]</code>	メッセージの非表示設定を解除し、すべてのメッセージを表示します。
<code><snapshot></code>	デザイン スナップショットまたは WDB ファイルの名前を指定します。

カテゴリ

[Simulation \(シミュレーション\)](#)

説明

`xsim` コマンドは、シミュレーション スナップショットを読み込んでバッチ モード シミュレーションを実行するか、GUI または Tcl ベースのインタラクティブ シミュレーション環境を開きます。スナップショットは、`xelab` コマンドを使用して生成する必要があります。

引数

`-view <arg>` (オプション): シミュレーションの波形アクティビティを保存する波形設定ファイルを開きます。波形設定ファイルには、単に表示する波形オブジェクト (信号、仕切り、グループ、仮想バス) のリストと、その表示プロパティおよびマーカーが含まれます。波形設定ファイルを現在のシミュレーションで保存するには、`save_wave_config` コマンドを使用します。

注記: このオプションを複数指定して複数の波形設定ファイルを開くことができます。

`-autoloadwcfg` (オプション): `<name>.wdb` という名前の波形データベース (WDB) を読み込んだときに、`<name>#.wcfg` という名前の関連の波形設定ファイル (WCFG) すべてを自動的に開きます。`-view` オプションを使用した場合は無視されます。

`-runall` | `-R` (オプション): イベント キューにイベントがなくなるまで、ブレークポイントまたは有効な条件が発生するまで、またはランタイム例外が発生するまでシミュレーションを実行し、シミュレータを終了します。これは、「`run -all; exit`」を実行するのと同様です。

`-maxdeltaid <arg>` (オプション): デルタ サイクルの最大数を 0 より大きい整数で指定します。デフォルト値は 10000 です。シミュレーション実行中に指定したシミュレーション ループ (デルタ サイクル) の最大数を超えると、シミュレータによりエラーがレポートされます。詳細は、『Vivado Design Suite ユーザー ガイド: ロジック シミュレーション』 (UG900) を参照してください。

`-nolog` (オプション): コマンド ライン XSIM ユーティリティとの互換性のために提供されているオプションで、Vivado Design Suite の Tcl で実行する場合は無視されます。

`-maxlogsize <arg>` (オプション): シミュレーション ログ ファイルの最大サイズを MB で指定します。デフォルト値は -1 で、ログ ファイルのサイズに制限はありません。

`-onfinish [stop | quit]` (オプション): シミュレーションの実行を終了したときのシミュレータの動作を指定します。有効な値は `stop` (シミュレーションを停止) または `quit` (シミュレータを終了) で、デフォルトは `stop` です。

`-onerror [stop | quit]` (オプション): エラーが発生したときのシミュレータの動作を指定します。有効な値は `stop` (シミュレーションを停止) または `quit` (シミュレータを終了) で、デフォルトは `stop` です。

`-tclbatch` | `-t` (オプション): シミュレータをバッチ モードで実行する Tcl スクリプトを指定します。

`-testplusarg <args>` (オプション): `plusargs` が `$test$plusargs` および `$value$plusargs` システム関数で使われるように指定します。これらの引数は、プラス記号 (+) で開始することにより、ほかのシミュレータ引数と区別されます。

`-vcdfile <arg>` (オプション): シミュレーション出力を保存する VCD (Value Change Dump) ファイルを指定します。

`-vcdunit <arg>` (オプション): VCD 出力の時間の単位を指定します。デフォルトの単位は、シミュレーション エンジンの精度と同じです。

`-wdb <arg>` (オプション): シミュレーション波形データベース (WDB) ファイルを指定します。シミュレーションが完了すると、シミュレーションがスタティック シミュレータ データベース ファイルに保存されます。保存されたファイルは、`open_wave_database` コマンドで開くことができます。

`-tp` (オプション): 実行されているプロセスの階層名を標準出力に表示します。

-tl (オプション): 実行されている文のファイル名と行番号を標準出力に表示します。

-nosignalhandlers (オプション): シミュレーションにおける OS レベルの信号ハンドラーのインストールをディスエーブルにします。信号ハンドラーをディスエーブルにすると、OS レベルの致命的なエラーにより、エラーの原因が示されずにシミュレーションが突然クラッシュすることがあります。



重要: このオプションは、セキュリティ ソフトウェアによりシミュレータがうまく動作しない場合にのみ使用してください。

-ieeewarnings (オプション): VHDL IEEE 関数の使用により生成される警告をイネーブルにします。これらの警告は通常無視できるので、デフォルトでは非表示になっています。このオプションを使用すると、これらの警告を表示できます。

-stats (オプション): シミュレータを終了するときに、メモリおよび CPU の使用量に関する統計を表示します。

-quiet (オプション): コマンドをメッセージを表示せずに実行します。実行中にエラーが発生しても、TCL_OK が返されます。

注記: コマンドの実行中にコマンド ラインで発生したエラーは返されます。コマンド内で発生したエラーのみが非表示になります。

-verbose (オプション): メッセージの非表示設定を一時的に解除し、コマンドからのすべてのメッセージを返します。

注記: メッセージの非表示設定は、set_msg_config コマンドで定義できます。

<snapshot> (必須): 実行するシミュレーション スナップショットの名前、または表示するために開く WEB ファイルの名前を指定します。スナップショットは、xelab コマンドでコンパイルしておく必要があります。WDB ファイルは、-wdb の xsim オプションを使用して保存されている必要があります。

例

次の例では、指定のシミュレーション スナップショットに対して xsim を実行します。

```
xsim C:/Data/project_xsim/project_xsim.sim/sim_1/behav/testbench_behav
```

関連項目

- [launch_simulation](#)

その他のリソースおよび法的通知

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、[ザイリンクス サポート](#) サイトを参照してください。

ソリューション センター

デバイス、ツール、IP のサポートについては、[ザイリンクス ソリューション センター](#)を参照してください。デザイン アシスタント、デザイン アドバイザリ、トラブルシューティングのヒントなどが含まれます。

トレーニング リソース

ザイリンクスでは、この資料に含まれるコンセプトを説明するさまざまなトレーニング コースおよび QuickTake ビデオを提供しています。次のリンクから関連するトレーニング リソースを参照してください。

- [トレーニング コース: UltraFast 設計手法](#)
- [トレーニング コース: UltraScale および UltraScale+ アーキテクチャでの設計](#)
- [トレーニング コース: Vivado Design Suite を使用した FPGA の設計](#)
- [Vivado Design Suite QuickTake ビデオ: プロジェクトを作成せずにバッチ ファイルを使用する方法 \(非プロジェクト フロー\)](#)
- [Vivado Design Suite QuickTake ビデオ: Vivado で制約ファイルとして Tcl スクリプトを使用](#)

参考資料

- 『Vivado Design Suite ユーザー ガイド: デザイン フローの概要』 (UG892)
- 『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』 (UG893)
- 『Vivado Design Suite ユーザー ガイド: Tcl スクリプト機能の使用』 (UG894)
- 『Vivado Design Suite ユーザー ガイド: 制約の使用』 (UG903)
- 『Vivado Design Suite プロパティ リファレンス ガイド』 (UG912)

Tcl Developer Xchange

Tcl リファレンス資料は、インターネットから入手できます。ザイリンクスでは、Tcl のオープンソース ベースを管理する Tcl Developer Xchange サイトをお勧めしています。

<http://www.tcl.tk>

入門用チュートリアルは、次から入手できます。

<http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html>

SDC について

SDC (Synopsys Design Constraints) は、特にタイミング解析において設計の要件をツールに渡すための業界標準です。SDC 仕様のリファレンス コピーは、次の Synopsys 社のサイトから登録をすると入手できます。

<http://www.synopsys.com/Community/Interoperability/Pages/TapinSDC.aspx>

お読みください: 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとし、また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、<https://japan.xilinx.com/legal.htm#tos> で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うこととなります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。<https://japan.xilinx.com/legal.htm#tos> で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとし、セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとし、また、

© Copyright 2012-2019 Xilinx, Inc. Xilinx、Xilinx のロゴ、Alveo、Artix、ISE、Kintex、Spartan、Versal、Virtex、Vivado、Zynq、およびこの文書に含まれるその他の指定されたブランドは、米国およびその他の各国のザイリンクス社の商標です。PCI、PCIe、および PCI Express は PCI-SIG の商標であり、ライセンスに基づいて使用されています。すべてのその他の商標は、それぞれの所有者に帰属します。