

Vivado Design Suite ユーザー ガイド

制約の使用

UG903 (v2020.1) 2020 年 8 月 17 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
2020 年 8 月 17 日 バージョン 2020.1	
資料全体	N/A

目次

改訂履歴	2
第 1 章: 概要	
UCF 制約から XDC 制約への変換	5
XDC 制約について	5
第 2 章: 制約の入力方法	
制約の入力方法について	7
制約の管理	7
制約の順序	11
制約の入力	17
合成制約の作成	54
インプリメンテーション制約の作成	59
制約の適用範囲の設定	62
制約の効率	68
第 3 章: クロックの定義	
クロックについて	73
プライマリ クロック	74
仮想クロック	77
生成クロック	78
クロック グループ	86
クロック レイテンシ、ジッター、ばらつき	89
第 4 章: I/O 遅延の制約	
I/O 遅延の制約について	91
入力遅延	91
出力遅延	94
第 5 章: タイミング例外	
タイミング例外について	97
マルチサイクル パス	98
フォルス パス	113
最小/最大遅延	116
ケース解析	122
タイミング アークのディスエーブル	124
第 6 章: CDC 制約	
CDC 制約について	126
バス スキュー制約	126

第 7 章: XDC の優先順位

XDC の優先順位について	132
XDC 制約の順序	132
例外の優先順位	132

第 8 章: 物理制約

物理制約について	135
ネットリスト制約	136
I/O 制約	138
配置制約	139
配線制約	142
コンフィギュレーション制約	143

第 9 章: 相対配置マクロの定義

相対配置マクロの定義について	145
デザイン エレメントのセットの定義	145
RPM の作成	146
セルの RPM セットへの割り当て	146
相対ロケーションの割り当て	148
RPM への固定ロケーションの割り当て	152
XDC マクロ	154
RPM の XDC マクロへの変換	166

付録 A: サポートされる XDC および SDC コマンド

XDC ファイルで使用可能なコマンド	168
サポートされる SDC コマンド	169
サポートされない SDC コマンド	180

付録 B: その他のソースおよび法的通知

ザイリンクス リソース	181
ソリューション センター	181
Documentation Navigator およびデザイン ハブ	181
参考資料	182
トレーニング リソース	182
お読みください: 重要な法的通知	183

概要

UCF 制約から XDC 制約への変換

Vivado® 統合設計環境 (IDE) では、ザイリンクス デザイン制約 (XDC) が使用され、ユーザー制約ファイル (UCF) フォーマットはサポートされていません。

ザイリンクス デザイン制約 (XDC) とユーザー制約ファイル (UCF) の制約には、重要な違いがいくつかあります。XDC は、業界標準の Synopsys® Design Constraints (SDC) フォーマットに基づいています。SDC は 20 年以上も使用され改善を重ねてきた、デザイン制約の記述に最もよく利用されているフォーマットです。



ビデオ: UCF 制約の XDC への移行については、[Vivado Design Suite QuickTake ビデオ: UCF 制約の XDC への変換](#)をご覧ください。

UCF には慣れているが XDC は初めてという場合は、『ISE から Vivado Design Suite への移行ガイド』(UG911) [\[参照 1\]](#) の「[UCF 制約の XDC への移行](#)」の章にある「XDC と UCF 制約の違い」を参照してください。この章では、XDC 制約を作成する開始点として、既存の UCF ファイルを XDC に変換する方法が説明されています。



重要: デザインに適切に制約を設定するには、XDC と UCF の基本的な違いを理解する必要があります。UCF から XDC への変換ユーティリティには限界があるので、XDC を正しく理解して作成するのが確実な方法です。このユーザー ガイドでは、各 XDC 制約について説明します。

XDC 制約について

XDC 制約は、業界標準の Synopsys Design Constraints (SDC バージョン 1.9) とザイリンクス独自の物理制約を組み合わせたものです。

XDC には、次の特徴があります。

- 単純な文字列ではなく、Tcl のセマンティクスに従ったコマンドです。
- ほかの Tcl コマンドと同様に、Vivado Tcl インタープリターで処理できます。
- ほかの Tcl コマンドと同様に、順次読み込まれ、解析されます。

XDC 制約は、フローの異なる段階で複数の方法で入力できます。

- 制約を 1 つまたは複数の XDC ファイルに保存します。

メモリ内に XDC ファイルを読み込むには、次のいずれかを実行します。

- `read_xdc` コマンドを使用します。
- XDC ファイルをプロジェクトの制約セットの 1 つに追加します。XDC ファイルで使用可能なビルトイン Tcl コマンドは、`set`、`list`、および `expr` のみです。サポートされているコマンドのリストは、[付録 A 「サポートされる XDC および SDC コマンド」](#) を参照してください。

- ツールで管理されない Tcl スクリプトで制約を生成します。

Tcl スクリプトを実行するには、次のいずれかを実行します。

- `source` コマンドを実行します。
- `read_xdc -unmanaged` コマンドを使用します。
- Tcl スクリプトをプロジェクトの制約セットの 1 つに追加します。



ヒント: XDC ファイルとは異なり、ツールで管理されない Tcl スクリプトには、デザイン オブジェクトを選択したりデザイン制約を定義するため、条件やループ制御構造などの一般的な Tcl コマンドを含めることができます。



重要: Vivado Design Suite では、同じ制約セットに XDC ファイルと Tcl スクリプトの両方を追加できます。変更された制約は、XDC ファイルからのものである場合のみ元の場所に保存されます。ツールで管理されない Tcl スクリプトからのものは元の場所には保存されません。Tcl スクリプトで生成された制約は Vivado Design Suite では管理されないため、インタラクティブには変更できません。詳細は、[第 2 章 「制約の入力方法」](#) を参照してください。



重要: XDC 制約の場合、`source` および `read_xdc` の動作に違いがあります。`source` コマンドでインポートした制約は、インポートした順番ではチェックポイントに保存されません。`read_xdc` でインポートした制約がまず保存されてから、`source` でインポートした制約が保存されます。すべての制約をデザインに適用した順番で保存するには、`source` ではなく、`read_xdc -unmanaged` を使用します。

デザインをメモリに読み込んだ後に特定の制約の構文を検証またはその影響を確認するには、Tcl コンソールおよび Vivado Design Suite のレポート機能を使用します。これは、タイミング制約および物理制約を解析し、デバッグするのに有益な方法です。

制約の入力方法

制約の入力方法について

デザイン制約は、デザインがボード上で正しく機能するようにするために、コンパイル フローで満たす必要のある要件を定義します。すべての制約がコンパイル フローのすべての段階で使用されるわけではありません。たとえば、物理制約はインプリメンテーション段階 (配置および配線) でのみ使用されます。

Vivado® 統合設計環境 (IDE) の合成およびインプリメンテーション アルゴリズムはタイミングドリブンなので、適切なタイミング制約を作成する必要があります。デザインの制約を厳しくしすぎたり緩くしすぎたりすると、タイミング クロージャを達成するのが困難になります。アプリケーションの要件に合った適度な制約を使用する必要があります。

制約の管理

Vivado IDE では、1 つまたは多数の制約ファイルを使用できます。コンパイル フロー全体に 1 つの制約ファイルを使用する方が便利のように思えますが、複雑なデザインでは、すべての制約を 1 つのファイルで管理するのは簡単ではありません。複数のチームによって開発された複数の IP コアや大型ブロックを使用するデザインの場合は、特にそうです。

タイミング制約と物理制約をインポートしたら、ソース ファイルの数またはプロジェクト モード/非プロジェクト モードにかかわらず、`write_xdc` コマンドを使用してすべての制約を 1 つのファイルにエクスポートできます。制約はプロジェクトまたはデザインに読み込まれた順序で指定の出力ファイルに記述されます。`write_xdc -type` コマンド ライン オプションを使用すると、制約のサブセット (タイミング、物理、または除外) を選択してエクスポートできます。



推奨: タイミング制約と物理制約は別のファイルに保存することをお勧めします。また、特定のモジュール用の制約を別のファイルに保存することもできます。

プロジェクト フロー

ザイリンクス デザイン制約 (XDC) ファイルは、新規プロジェクトを作成するとき、または後で Vivado IDE メニューから制約セットに追加できます。

図 2-1 に、プロジェクトに 2 つの制約セットが含まれている例を示します。1 つ目の制約セットには、2 つの XDC ファイルが含まれています。2 つ目の制約セットでは、すべての制約を含む XDC ファイルを 1 つ使用します。

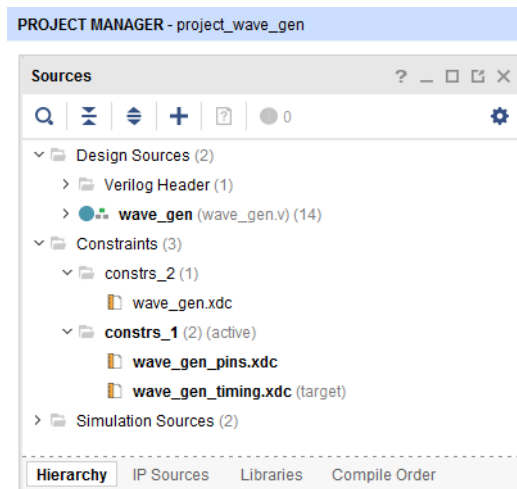


図 2-1: 1 つの XDC ファイルまたは複数の XDC ファイル



重要: プロジェクトに独自の制約を使用する IP が含まれている場合、その制約ファイルは制約セットには表示されず、IP ソース ファイルと共にリストされます。

Tcl スクリプトを、ツールで管理されない制約または Tcl スクリプトとして制約セットに追加することもできます。これらの制約を変更した場合、Vivado Design Suite ではこれらの変更は管理されない Tcl スクリプトには保存されません。Tcl スクリプトおよび XDC ファイルは、Vivado IDE に表示されているのと同じ順序で読み込まれるか (同じ `PROCESSING_ORDER` グループに属している場合)、または `report_compile_order -constraints` コマンドでレポートされるのと同じ順序で読み込まれます。

1 つの XDC ファイルまたは Tcl スクリプトを必要に応じて複数の制約セットで使用できます。制約ファイルおよび制約セットの作成方法、プロジェクトへの追加方法は、『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895) [参照 2] のこのセクションを参照してください。

非プロジェクト フロー

非プロジェクト モードでは、コンパイル コマンドを実行する前に各ファイルを読み込む必要があります。

次に、合成およびインプリメンテーションで複数の XDC ファイルを使用するスクリプト例を示します。

スクリプト例:

```
read_verilog [glob src/*.v]
read_xdc wave_gen_timing.xdc
read_xdc wave_gen_pins.xdc
synth_design -top wave_gen -part xc7k325tffg900-2
opt_design
place_design
route_design
```

アウト オブ コンテキスト制約

Dynamic Function eXchange を使用するデザインの場合は、デザインの一部をアウト オブ コンテキスト (OOC) 方法で合成するのが一般的です。このフローが使用されると、一部の制約を OOC 合成のみに指定できます。たとえば、ブロックが合成済み OOC の場合、ブロックの入力バウンダリで伝搬されるクロックを定義する必要があります。これらのクロックは、OOC の XDC ファイル内で定義します。

プロジェクト モード:

```
add_file constraints_ooc.xdc
set_property USED_IN {synthesis out_of_context} [get_files constraints_ooc.xdc]
```

アウト オブ コンテキストは、GUI (constraints_ooc.xdc ファイルのプロパティ) を介して XDC ファイルでも設定できます。

非プロジェクト モード:

```
read_xdc -mode out_of_context constraints_ooc.xdc
```

合成およびインプリメンテーション制約ファイル

デフォルトでは、プロジェクトに追加されたすべての XDC ファイルおよび Tcl スクリプトが、合成およびインプリメンテーションの両方で使用されます。XDC ファイルまたは Tcl スクリプトに `USED_IN_SYNTHESIS` および `USED_IN_IMPLEMENTATION` プロパティを設定すると、このデフォルトを変更できます。これらのプロパティに有効な値は、`TRUE` または `FALSE` です。



重要: `DONT_TOUCH` 属性は `USED_IN_SYNTHESIS` および `USED_IN_IMPLEMENTATION` のプロパティには従いません。合成 XDC で `DONT_TOUCH` プロパティを使用すると、`USED_IN_IMPLEMENTATION` の値にかかわらず、インプリメンテーションに伝搬されます。

`DONT_TOUCH` 属性の詳細は、[54 ページの「RTL 属性」](#)を参照してください。



重要: モジュール (IP/BD/...) がアウト オブ コンテキスト (OOC) モードで合成される場合、最上位合成 run でこれらのモジュールに対してブラック ボックスが推論されます。このため、最上位合成制約は OOC モジュール内部のピン、ネット、セルなどのオブジェクトを参照できません。最上位制約の中に OOC モジュール内のオブジェクトを参照するものがある場合、制約を合成用 (USED_IN_SYNTHESIS=TRUE / USED_IN_IMPLEMENTATION=FALSE) とインプリメンテーション用 (USED_IN_SYNTHESIS=FALSE / USED_IN_IMPLEMENTATION=TRUE) の 2 つの XDC ファイルに分割する必要があります。インプリメンテーション中はこのような制限はありません。これは、OOC モジュールの DCP からのネットリストが、最上位デザイン ファイル合成する際に生成されたネットリストとリンクされ、ブラック ボックスが解決されるためです。インプリメンテーションで使用するために生成された XDC 出力ファイルは、ほかのユーザー制約と共に適用されます。

たとえば、制約ファイルをインプリメンテーションのみに使用する場合は、次の手順に従います。

1. [Sources] ウィンドウで制約ファイルを選択します。
2. [Source File Properties] ウィンドウで、オプションを次のように設定します。
 - a. [Synthesis] チェック ボックスをオフにします。
 - b. [Implementation] チェック ボックスをオンにします。

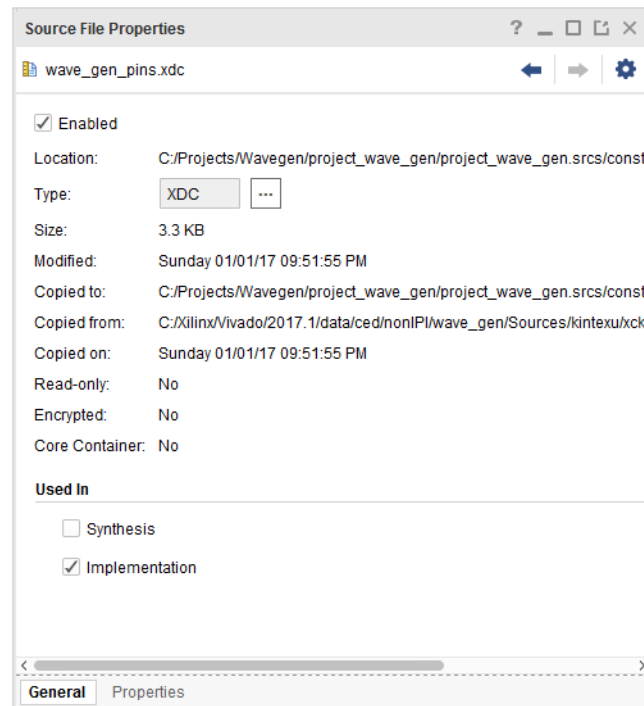


図 2-2: [Source File Properties] ウィンドウ

同等の Tcl コマンドは、次のとおりです。

```
set_property USED_IN_SYNTHESIS false [get_files wave_gen_pins.xdc]
set_property USED_IN_IMPLEMENTATION true [get_files wave_gen_pins.xdc]
```

Vivado を非プロジェクト モードで実行している場合、フローのどの段階でも制約を直接読み込むことができます。このモードでは、USED_IN_SYNTHESIS および USED_IN_IMPLEMENTATION プロパティは関係ありません。

次に、2 つの XDC ファイルをフローの異なる段階で読み込むコンパイル Tcl スクリプト例を示します。

```
read_verilog [glob src/*.v]
read_xdc wave_gen_timing.xdc
synth_design -top wave_gen -part xc7k325tffg900-2
read_xdc wave_gen_pins.xdc
opt_design
place_design
route_design
```

表 2-1: 合成前後の XDC ファイルの読み込み

ファイル名	ファイルを読み込む段階	制約が使用される段階
wave_gen_timing.xdc	合成前	<ul style="list-style-type: none"> • 合成 • インプリメンテーション
wave_gen_pins.xdc	合成後	<ul style="list-style-type: none"> • インプリメンテーション



ヒント: 合成後に読み込まれた制約が、合成前に読み込まれた制約に加えて適用されます。

制約の順序

XDC 制約は順に適用され、規則に従って優先順位が付けられるので、制約の順序をよく確認しておく必要があります。詳細は、[第 7 章「XDC の優先順位」](#)を参照してください。

注記: 複数の物理制約が競合している場合は、最後の制約が適用されます。たとえば、I/O ポートが複数の XDC ファイルにより異なる場所に割り当てられている場合、ポートに最後に割り当てられた場所が適用されます。

Vivado IDE では、デザインを詳細に確認できます。制約を検証するには、次の手順に従います。

1. 適切なレポート コマンドを実行します。
2. [Tcl Console] または [Messages] ウィンドウのメッセージを確認します。

推奨される制約の順序



推奨: デザインで XDC ファイルを 1 つまたは複数使用する場合のどちらでも、制約を次の順序で指定します。

```
## Timing Assertions Section
# Primary clocks
# Virtual clocks
# Generated clocks
# Clock Groups
# Bus Skew constraints
# Input and output delay constraints

## Timing Exceptions Section
# False Paths
# Max Delay / Min Delay
# Multicycle Paths
# Case Analysis
# Disable Timing

## Physical Constraints Section
# located anywhere in the file, preferably before or after the timing constraints
# or stored in a separate constraint file
```

注記: クロックの関係またはクロック伝搬を変更するケース解析制約は、生成クロックを定義する前に定義する必要があります。これには、クロック バッファにその出力クロックに影響するようなケース解析を定義する場合も含まれます。

まずクロック定義から始めます。クロックを作成しないと、クロックをほかの制約で使用することはできません。クロックを定義する前にクロックを参照すると、エラーが発生し、その制約は無視されます。これは、1 つの XDC ファイル内でも、デザイン内のすべての XDC ファイル (または Tcl スクリプト) でも同様です。

制約ファイルの順序は重要です。各ファイルの制約が、ほかのファイルの制約に依存しないようにする必要があります。あるファイルの制約が、ほかのファイルの制約に依存している場合は、依存する制約を含むファイルを最後に読み込む必要があります。2 つの制約ファイルが相互に依存している場合は、1 つのファイルに統合して制約を適切な順序で記述するか、2 つのファイルをさらに分割してそれらのファイルの順序が正しくなるようにする必要があります。

制約の順序の編集

Vivado IDE では、編集した制約は XDC ファイルの元の位置に保存されますが、Tcl スクリプトには保存されません。新しい制約はすべて、「target」とマークされている XDC ファイルの最後に保存されます。制約セットに複数の XDC ファイルが含まれている場合、このターゲット制約ファイルはリストの最後にあるファイルではないことが多く、デザインを開いたとき、またはデザインを読み込み直したときに、最後に読み込まれるとは限りません。そのため、制約ソース ファイルに保存された制約の順序がメモリ内での順序と異なるものになる可能性があります。



重要: 制約ファイルに保存されている最終的な順序が、意図したとおりに機能することを確認する必要があります。順序を変更する必要がある場合は、制約ファイルを直接編集します。これは、特にタイミング制約の場合に重要です。

制約ファイルの順序

IP を含まないプロジェクト フローでは、制約はすべて制約セットに含まれます。デフォルトでは、エラボレート済みデザインまたは合成済みデザインをメモリに読み込むときに、Vivado IDE に表示されている順序で XDC ファイル (または Tcl スクリプト) が読み込まれます。リストの 1 番上のファイルが最初に読み込まれ、1 番下のファイルが最後に読み込まれます。この順序を変更するには、IDE でファイルを選択し、リスト内での位置を変更します。

たとえば図 2-3 では、ドラッグ アンド ドロップで `wave_gen_pin.xdc` を `wave_gen_timing.xdc` の前に移動しています。



図 2-3: Vivado IDE での XDC ファイル順の変更

これと同等の Tcl コマンドは、次のとおりです。

```
reorder_files -fileset constrs_1 -before [get_files wave_gen_timing.xdc] \
[get_files wave_gen_pins.xdc]
```

表 2-2: ファイル順の変更前と変更後

ファイル	変更前のファイル順	変更後のファイル順
<code>wave_gen_timing.xdc</code>	1	2
<code>wave_gen_pins.xdc</code>	2	1

非プロジェクト モードでは、`read_xdc` で読み込む順序が、制約ファイルが評価される順序になります。

IP コアを含む場合の制約ファイルの順序

IP コアは、1 つまたは複数の XDC ファイルと共に配布されることが多く、そのような IP コアが RTL プロジェクト内で生成されると、その XDC ファイルもさまざまなデザイン コンパイル段階で使用されます。

たとえば、図 2-4 では、プロジェクトに含まれている IP の 1 つが、1 つの XDC ファイルと共に配布されています。

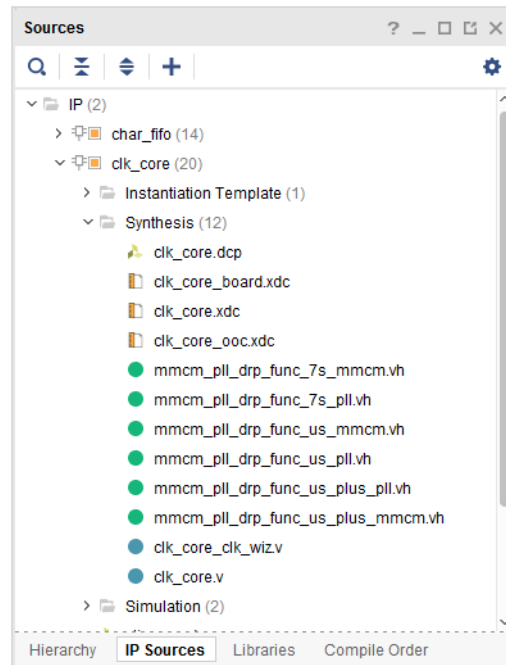


図 2-4: IP ソースの XDC ファイル

デフォルトでは、IP の XDC ファイルがユーザー XDC ファイルの前に読み込まれます。この順序で処理することで、IP に XDC で参照できるクロック オブジェクトを作成できます。また、ユーザー制約が IP の後に評価されるので、IP コアで設定された物理制約をユーザー制約で上書きできます。ただし、ユーザーまたは別の IP で生成されたクロック オブジェクトに依存している IP コアは例外です (`get_clocks -of_objects [get_ports clka]` など)。この場合、IP の XDC がユーザー ファイルの後に読み込まれます。

この動作は、各 XDC ファイルに設定されている `PROCESSING_ORDER` プロパティで指定します。

- **EARLY:** 最初に読み込む必要のあるファイル
- **NORMAL:** デフォルト
- **LATE:** 最後に読み込む必要のあるファイル

IP の XDC では、`PROCESSING_ORDER` プロパティは **EARLY** または **LATE** に設定されます。IP の XDC ファイルの `PROCESSING_ORDER` プロパティが **NORMAL** に設定されていることはありません。`PROCESSING_ORDER` 設定が同じユーザー XDC (または Tcl) ファイルは、Vivado IDE で表示されている順序で読み込まれます。グループ内の順序を変更するには、Vivado IDE の制約セット内でファイルの位置を移動するか、`reorder_files` コマンドを使用します。

`PROCESSING_ORDER` 設定が同じ IP の XDC ファイルは、IP コアをインポートまたは作成した順序で読み込まれます。この順序は、プロジェクトの作成後には変更できません。

ユーザー グループと IP XDC PROCESSING_ORDER グループの間の相対的な順序は、次のとおりです。

1. EARLY とマークされているユーザー制約
2. EARLY とマークされている IP 制約 (デフォルト)
3. NORMAL とマークされているユーザー制約
4. LATE とマークされている IP 制約 (クロック依存関係を含む)
5. LATE とマークされているユーザー制約

注記: ユーザー制約の後に処理するため PROCESSING_ORDER が LATE に設定されている IP の XDC ファイルは、<IP_NAME>_clocks.xdc という名前になります。

次の図に、PROCESSING_ORDER プロパティの設定例を示します。

PROCESSING_ORDER	EARLY
—	EARLY
—	NORMAL
	LATE

図 2-5: PROCESSING_ORDER プロパティの設定例

これと同等の Tcl コマンドは、次のとおりです。

```
set_property PROCESSING_ORDER EARLY [get_files wave_gen_pins.xdc]
```



推奨: Tcl コンソールで `report_compile_order -constraints` コマンドを使用して、IS_ENABLED、USED_IN_SYNTHESIS、および USED_IN_IMPLEMENTATION などの上記のプロパティに基づいてツールで決定された XDC ファイル読み出しシーケンスがレポートされるようにします。

注記: IP をアウト オブ コンテキストで合成すると、IP によりデフォルトのクロック定義を含む _occ.xdc ファイルが必要に応じて供給されます。_occ.xdc の USED_IN プロパティは `synthesis out_of_context implementation` (順序は関係なし) に設定されます。アウト オブ コンテキスト合成中は、_occ ファイルはほかの制約よりも先に処理されます。

読み込み順の変更

制約セット内での XDC ファイルまたはツールで管理されない Tcl スクリプトの読み込み順序を変更するには、次の手順に従います。

1. [Sources] ウィンドウで、移動する XDC ファイルまたは Tcl スクリプトを選択します。
2. そのファイルを、制約セットの適切な位置にドラッグ アンド ドロップします。

図 2-3 の例を Tcl コマンドにすると、次のようになります。

```
reorder_files -fileset constrs_1 -before [get_files wave_gen_timing.xdc] \
[get_files wave_gen_pins.xdc]
```

非プロジェクト モードでは、`read_xdc` または `source` コマンドの順序により、制約ファイルが評価される順序が決定します。

制約を含む IP コアを使用する場合は、2 つのグループの制約が次のように自動的に処理されます。

- クロックに依存しない制約は、PROCESSING_ORDER が EARLY に設定された XDC ファイルに含まれます。
- クロックに依存する制約は、PROCESSING_ORDER が LATE に設定された XDC ファイルに含まれます。

デフォルトでは、ユーザーの XDC ファイルは PROCESSING_ORDER NORMAL グループに含まれます。これらは、EARLY XDC ファイルの後、LATE XDC ファイルの前に読み込まれます。各 PROCESSING_ORDER グループごとに、IP の XDC ファイルが IP コアの [IP Sources] ウィンドウでリストされるのと同じ順序で読み込まれます。たとえば、次の図は XDC ファイルを含むプロジェクト IP コアの 1 つを示しています。

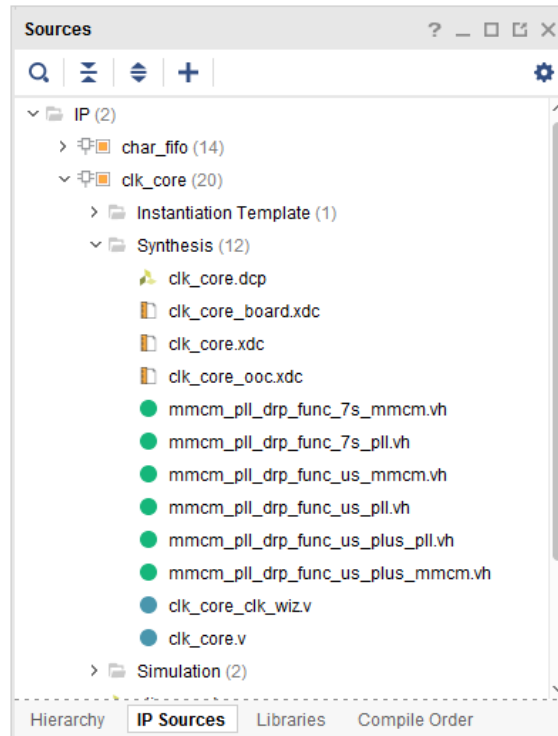


図 2-6: IP ソースの XDC ファイル

デザインを開くと、ログ ファイルに IP の XDC ファイルが最後に読み込まれていることがわかります。

```
Parsing XDC File [C:/project_wave_gen_hdl.srscs/sources_1/ip/clk_core/clk_core.xdc]
for cell 'clk_gen_i0/clk_core_i0/inst'
Finished Parsing XDC File
[C:/project_wave_gen_hdl.srscs/sources_1/ip/clk_core/clk_core.xdc] for cell
'clk_gen_i0/clk_core_i0/inst'
Parsing XDC File
[C:/project_wave_gen_hdl.srscs/sources_1/ip/char_fifo/char_fifo/char_fifo.xdc] for
cell 'char_fifo_i0/U0'
Finished Parsing XDC File
[C:/project_wave_gen_hdl.srscs/sources_1/ip/char_fifo/char_fifo/char_fifo.xdc] for
cell 'char_fifo_i0/U0'
Parsing XDC File
[C:/project_wave_gen_hdl.srscs/constrs_1/imports/verilog/wave_gen_timing.xdc]
Finished Parsing XDC File
[C:/project_wave_gen_hdl.srscs/constrs_1/imports/verilog/wave_gen_timing.xdc]
Parsing XDC File
[C:/project_wave_gen_hdl.srscs/sources_1/ip/char_fifo/char_fifo/char_fifo_clocks.xdc
] for cell 'char_fifo_i0/U0'
Finished Parsing XDC File
[C:/project_wave_gen_hdl.srscs/sources_1/ip/char_fifo/char_fifo/char_fifo_clocks.xdc
] for cell 'char_fifo_i0/U0'
Completed Processing XDC Constraints
```


ユーザー XDC ファイルとは異なり、同じ `PROCESSING_ORDER` グループの IP XDC ファイルが読み込まれる順序を直接変更することはできません。変更が必要な場合は、次を実行します。

1. 対応する IP の XDC ファイルをディスエーブルにします (`IS_ENABLED` を `false` に設定)。
2. そのファイルの内容をコピーします。
3. 制約セットに含まれる XDC ファイルのいずれかにコピーした内容を貼り付けます。
4. コピーされた IP の XDC コマンドを、階層ネストリスト オブジェクトの完全なパス名を使用してアップデートします。IP の XDC 制約は適用範囲が IP インスタンスに限定されるように記述されているので、この作業は必要です。
5. 範囲が限定された制約に対して特別な方法で処理される `get_ports` クエリを確認します。XDC の適用範囲の詳細は、[62 ページの「制約の適用範囲の設定」](#)を参照してください。

制約の入力

Vivado IDE で制約を入力する方法はいくつかあります。テキスト エディターで直接 XDC ファイルを編集する場合を除き、Vivado IDE の制約入力機能にアクセスするにはデザイン データベース (エラボレート済み、合成済み、またはインプリメント済み) を開く必要があります。

メモリ内の制約の保存

編集中の制約を検証するには、メモリにデザインを読み込んでおく必要があります。Vivado IDE ユーザー インターフェイスを使用して制約を編集する場合は、Tcl コンソールで同等の XDC コマンドが実行され、メモリ内で適用されます。編集したタイミング制約を XDC ファイルに保存するには、まずメモリに適用する必要があります。

合成またはインプリメンテーションを実行する前に、メモリ内の制約をプロジェクトの XDC ファイルに保存する必要があります。Vivado IDE では、制約の保存が必要な場合はメッセージが表示されます。

制約を手動で保存するには、次のいずれかを実行します。

- ツールバーの [Save Constraints] をクリックします。
- [File] → [Constraints] → [Save] をクリックします。

注記: メモリ内の制約を保存する際、合成およびインプリメンテーションが最新の状態でなくなる可能性があることを示すダイアログ ボックスが表示されます。このダイアログ ボックスの [Remember Preference] チェック ボックスをオンにすると、今後この警告が表示されないようになります。

これらのコマンドを実行すると、Vivado で次の処理が実行されます。

- 新しい制約はすべて、デザインに関連付けられている制約セットの「target」とマークされている XDC ファイルに保存されます。
- 編集されたすべての制約が、元の XDC ファイルに保存されます。

注記: 制約管理システムでは、可能な限り元の XDC ファイル フォーマットが保持されます。

制約編集フロー

図 2-7 に、推奨される制約編集フローを示します。これら 2 つのフローを混合しないでください。混合すると、制約が失われる可能性があります。推奨されるフローは、次のとおりです。

- 「ユーザー インターフェイスを使用」
- 「手動で編集」

ユーザー インターフェイスを使用

制約は Vivado IDE で管理されるので、XDC ファイルを同時に編集しないでください。Vivado IDE でメモリの内容を保存すると、制約は次のように保存されます。

- 変更された制約は、元のファイルの元の制約に上書きされます。
- 新しい制約は、「target」とマークされたファイルに追加されます。
- XDC ファイルを手動で変更した場合、変更内容はすべて上書きされます。

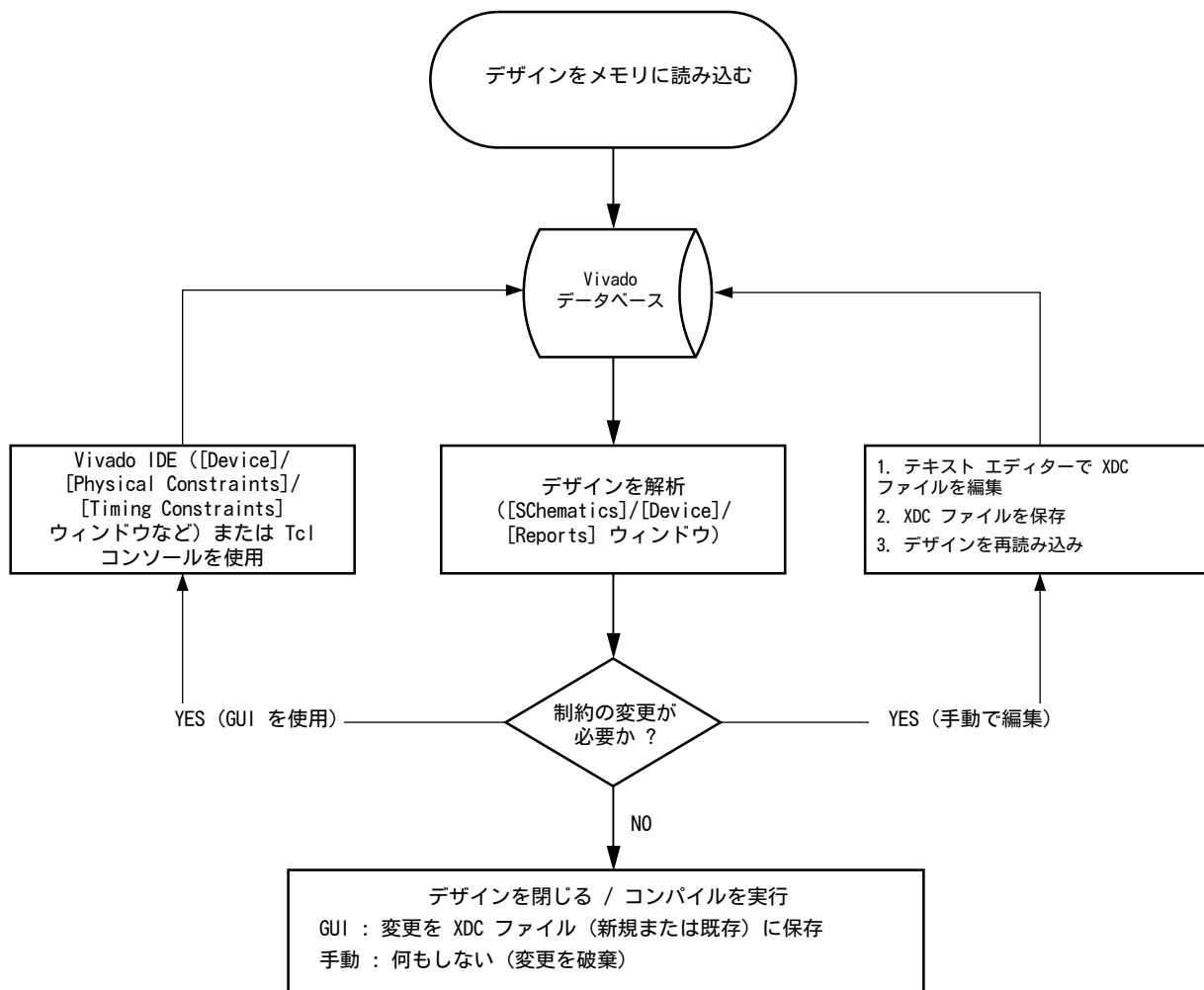
手動で編集

制約を手動で編集する場合、ユーザーが XDC ファイルを変更および管理します。Tcl コンソールを使用して制約の構文を確認することはできますが、デザインを閉じたり読み込み直したりすると、メモリ内の変更は破棄されます。

制約を保存するときに競合がある場合は、次のいずれかを選択するようメッセージが表示されます。

- メモリで加えた変更を破棄する
- 変更を新しいファイルに保存する
- XDC ファイルを上書きする

制約の作成は繰り返し作業です。ケース バイ ケースで、IDE エディターを使用したり、制約ファイルを手動で変更できます。



X12983

図 2-7: 制約編集フロー

各繰り返し実行で、[図 2-7](#) に示されている両方のオプションを同時に使用しないでください。

2 つのオプションを切り替えて使用する場合は、まず制約を保存するか、デザインを読み込み直し、メモリ内の制約が XDC ファイルと一致するようにします。

ピン割り当て

RTL 解析、合成、インプリメンテーション環境で最上位ポートを作成したり、既存の配置を変更するには、次の手順に従います。

1. [I/O Planning] レイアウトを選択します。

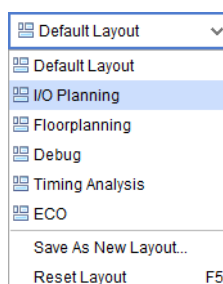


図 2-8: [I/O Planning] レイアウト

2. 表 2-3 に示すウィンドウを開きます。

表 2-3: 最上位ポートを作成、および既存の配置を変更するウィンドウ

ウィンドウ	機能
[Device]	デバイス フロアプランでポートの位置を表示および変更します。
[Package]	デバイス パッケージでポートの位置を表示および変更します。
[I/O Ports]	ポートを選択して [Device] または [Package] ウィンドウにドラッグ アンド ドロップして配置したり、各ポートの現在の割り当てを表示します。
[Package Pins]	各 I/O バンクのリソースの使用状況を表示します。

ピン割り当ての詳細は、『Vivado Design Suite ユーザー ガイド: I/O およびクロック プランニング』(UG899) [\[参照 3\]](#) のこのセクションを参照してください。

フロアプラン

RTL 解析、合成、インプリメンテーション環境で Pblock を作成および変更するには、次の手順に従います。

1. [Floorplanning] レイアウトを選択します。

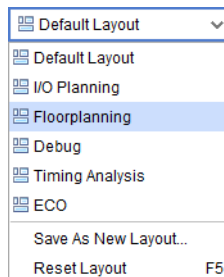


図 2-9: [Floorplanning] レイアウト

2. 表 2-4 に示すウィンドウを開きます。

表 2-4: Pblock を作成および変更するウィンドウ

ウィンドウ	機能
[Netlist]	Pblock に割り当てるセルを選択します。
[Physical Constraints]	既存の Pblock とそのプロパティを確認します。
[Device]	デバイス上の Pblock の形状と場所を作成および変更します。

特定の BEL またはサイトにセル配置制約を作成するには、次の手順に従います。

1. [Netlist] ウィンドウでセルを選択します。
2. セルをドラッグして [Device] ウィンドウの適切な位置に配置します。

フロアプランの詳細は、『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) [参照 4] のこのセクションを参照してください。

Timing Constraints ウィザード

Timing Constraints ウィザードでは、合成済みまたはインプリメント済みのデザインに不足しているタイミング制約が検出されます。『UltraFast 設計手法 (Vivado Design Suite 用)』(UG949) [参照 5] の推奨事項に従っていることを確認するため、ネットリスト、クロック ネットの接続性、および既存のタイミング制約が解析されます。制約の 3 つのカテゴリについてウィザードの 11 ページで解析され、最後にサマリ ページが表示されます。ウィザードには、次のステップが含まれます。

- クロック
 - プライマリ クロック
 - 生成クロック
 - フォワード クロック
 - 外部フィードバック遅延
- 入力および出力ポート
 - 入力遅延
 - 出力遅延
 - 組み合わせ遅延
- クロック乗せ換え
 - 物理的に排他的なクロック グループ
 - 論理的に排他的なクロック グループ (クロック関連性なし)
 - 論理的に排他的なクロック グループ (相互作用あり)
 - 非同期クロック乗せ換え
- [Constraints Summary] ページ

各ステップで、推奨された制約を追加するか、推奨された各制約をオン/オフにしてリストを変更できます。ただし、ウィザードの最初の方のページで推奨された制約をオフにすると、後のページで不足している制約が検出されなくなることがあります。たとえば、あるクロックの作成をオフにすると、そのクロックまたはそのクロックから自動生成されるクロックを基準とする制約は検出されず、推奨されなくなります。

ウィザードの最後のページには、作成される制約のサマリが表示されます。各ハイパーリンクをクリックすると、制約の詳細が表示されるか、ウィザード後に [Timing Constraints] ウィンドウでその新しい制約が表示されるようになります。

[Finish] をクリックするときに、次の推奨レポートが生成されるように選択することもできます。

- [Create Timing Summary report]: check_timing レポートに加え、新しい制約を使用したタイミング スラックがレポートされます。入力した周期または I/O 遅延制約を満たすのが困難な場合は、タイミング違反が表示されることがあります。
- [Create Check Timing report]: check_timing コマンドを実行して不足している制約または不適切な制約が特定されます。
- [Create DRC Report using only Timing Checks]: タイミング DRC が実行されます。



重要: 新しく追加された制約は、[Cancel] をクリックしない限り、ターゲット XDC に自動的に保存されます。ウィザード終了後は、[Timing Constraints] ウィンドウで新しい制約を編集または削除できます。

Timing Constraints ウィザードで、適切にタイミング解析できない危険な状況が発生させる制約が推奨されることはありません。また、デザインをメモリに読み込んだときに既に存在している不適切な制約は、ウィザードでは修正されません。ただし、Vivado Design Suite をプロジェクト モードで使用し、不足しているクロックをすべて作成することにより、無効だった制約が有効になることがあります。詳細は、「[制約の処理順と無効な制約](#)」を参照してください。ウィザードの終了後に check_timing または report_drc を実行して制約の問題がレポートされる場合、ソース XDC ファイルに既に存在していた制約が原因であることがほとんどなので、ウィザードを使用するのではなく、直接問題を解決する必要があります。



ビデオ: Vivado Timing Constraints ウィザードの詳細は、[Vivado Design Suite QuickTake ビデオ: Vivado タイミング制約ウィザードの使用](#)を参照してください。

制約の処理順と無効な制約

Timing Constraints ウィザードでは、クロックを定義したり、クロックを参照する制約が推奨され、プロジェクト モードの場合はターゲット XDC ファイルの最後に、それ以外のモードの場合はすべての制約の最後に保存されます。このため、次の規則を理解しておく必要があります。

- プロジェクト モード: Timing Constraints ウィザードを起動する前に、PROCESSING_ORDER を NORMAL に設定したターゲット XDC ファイルを指定する必要があります。ターゲット XDC ファイルは、メモリで開いているデザインの現在選択されている制約セットに含まれている必要があります。すべての XDC ファイルの中でターゲット XDC ファイルがどこに位置しているかにより、推奨された制約がどこで適用され、保存されるかが決まります。また、ウィザードはできるだけ完全で正確な制約を推奨するため、ターゲット XDC ファイルの後に解析される XDC ファイルに含まれる無効な制約を適用し直そうとします。

たとえば、制約セット constr_1 が選択された synth_1 run からのネットリストがメモリで開いているとします。この制約セットには、3 つの XDC ファイルが a.xdc、b.xdc、c.xdc の順で含まれています。ターゲット XDC ファイルとして b.xdc を選択し、各ファイルに無効な制約が含まれていると、Timing Constraints ウィザードは、次の段階に進んでほかの不足している制約を検出する前に、推奨クロックを適用して c.xdc の無効な制約を適用し直します。

- 非プロジェクト モードまたはデザイン チェック ポイント (DCP) モード: これらのモードではターゲット XDC ファイルを指定できないので、Timing Constraints ウィザードは新しい制約を制約順の最後に推奨して適用します。これは、Tcl コンソールまたは [Timing Constraints] ウィンドウに新しい制約を入力するのと同じです。これらのモードでは、ウィザードは無効な制約を適用し直そうとはしません。制約の依存関係や優先度の問題を回避するために新しい制約を先に適用する必要がある場合は、制約の順序を手動で変更する必要があります。

制約を手動で変更するには、次の手順に従います。

- Vivado Design Suite を使用して新しい制約を作成します。
- 次のいずれかを実行します。

```
write_xdc -exclude_physical timing_constraints.xdc
```

```
write_xdc -type timing timing_constraints.xdc
```

- timing_constraints.xdc を編集して新しい制約を XDC ファイルの上の方に移動します。
- ファイルを保存します。
- 次のコマンドを実行します。

```
reset_timing
```

- f. 次のコマンドを実行して、編集したタイミング制約ファイルを読み込みます。

```
read_xdc timing_constraints.xdc
```

アップデートされたタイミング制約の順序は、[Timing Constraints] ウィンドウを使用して確認できます。新しい制約を確認したら、その順序を DCP に保存できます。

ウィザードを開いているときに使用可能なレポート機能

Timing Constraints ウィザードを開いているときは、データベースに競合が発生しないようにするため、[Tcl Console] ウィンドウの使用やタイミング解析の実行など、Vivado IDE でほとんどの操作が実行できなくなります。ウィザード ウィンドウは、その他の Vivado IDE ウィンドウよりも常に手前に表示されます。Vivado IDE のメニューまたはウィンドウを使用する必要がある場合は、ウィザード ウィンドウを移動する必要があります。

Timing Constraints ウィザードが開いているときに使用できるのは、次の機能のみです。

- クロック ネットワークをレポートおよび表示

ほとんどのウィザードのページにクロック ネットワーク レポートを生成して表示するためのボタンがあり、クロック トポロジ、その起点、一部のクロックの共有セグメントを表示できます。

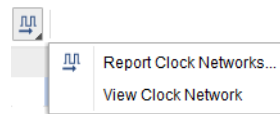


図 2-10: クロック ネットワークをレポートまたは表示するボタン

クロック ネットワーク レポートの詳細は、『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) [参照 4] を参照してください。

- ソース ファイルで名前を検索、またはメモリ内のデザインでオブジェクトを検索

[Edit] メニューから [Find] または [Find In Files] をクリックし、[Find] または [Find In Files] ダイアログ ボックスを開きます。これらのダイアログ ボックスを使用すると、ウィザードで制約を入力しながら、デザインに関する情報を取得できます。

- 回路図を作成および表示

Vivado IDE のメイン ウィンドウでデザイン オブジェクトを選択し、それらを回路図で表示できます。すべての回路図機能を使用できます。Timing Constraints ウィザードでは、[Asynchronous Clock Domain Crossings] ページの [Timing Paths] タブで 1 つまたは複数のエントリを選択した場合にのみ、回路図クロスプローブ機能がサポートされます。

回路図の使用に関する詳細は、『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893) [参照 7] を参照してください。

- [Timing Constraints] ウィンドウを使用してメモリ内の制約を表示します。

ウィザードの各ページには、推奨される制約と同じタイプの既存の制約を示すタブが含まれます。これにより、XDC ファイルに既に含まれる制約の詳細をすばやく確認できます。メモリ内のタイミング制約すべてを表示するため、[Timing Constraints] ウィンドウには、適用範囲の情報も含め、制約の順序すべてが XDC ファイル別に分類されて表示されます。無効な制約も表示されます。

ウィザード内での制約の編集

ウィザードの各ページでは、複数の制約が推奨されます。制約によって、次のいずれかを実行する必要があります。

- 次のいずれかの方法で、作成しない制約をオフにします。
 - リストで制約を 1 つずつオフにして削除します。
 - 表の左上のチェック ボックスをオフにして、すべての制約を削除します。



ヒント: または、[図 2-11](#) に示すように、制約を右クリックして [Do Not Create Constraint] をクリックします。

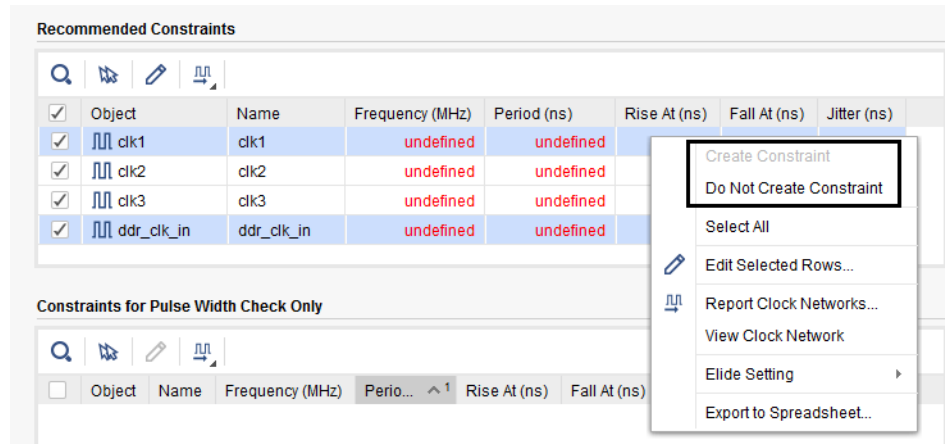


図 2-11: コンテキスト メニューを使用した推奨制約の削除

[図 2-12](#) では、clk1 および ddr_clk_in がオフになっており、スキップされます。

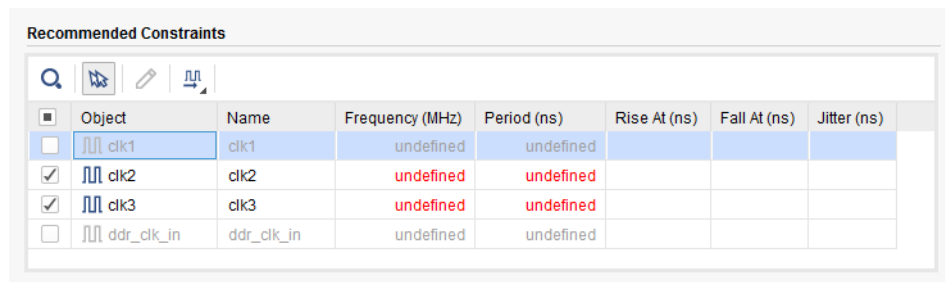


図 2-12: 推奨制約の作成と削除

- [undefined] と表示されているセル ([図 2-12](#) では clk2 および clk3 の [Frequency] または [Period]) をクリックし、値を入力します。

該当する行を選択して [Edit Selected Rows] ボタンをクリックすると、複数の制約を同時に編集できます ([図 2-13](#))。

Recommended Constraints

Object	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
<input type="checkbox"/> clk1	undefined	undefined			
<input checked="" type="checkbox"/> clk2	undefined	undefined			
<input checked="" type="checkbox"/> clk3	undefined	undefined			
<input type="checkbox"/> ddr_clk_in	undefined	undefined			

図 2-13: 複数の推奨制約の編集

次に、必要なフィールド (図 2-14 の [Frequency] および [Period] など) を入力します。

Primary Clock Constraints

Specify the 'period' or 'frequency', and optionally the 'rise at', 'fall at' and 'jitter' values.

Frequency: MHz (required)

Period: ns (required)

Rise at: ns (optional)

Fall at: ns (optional)

Jitter: ns (optional)

図 2-14: 複数の推奨制約に対するパラメーターを入力

複数の制約を一度に編集する機能は、入力および出力遅延制約などを編集する際に特に便利です。

- 変更が不要な場合は、単に制約を確認します。

すべての推奨制約を確認したら、[Next] をクリックして次のページに進みます。入力し忘れたエントリがあると、ウィザードの次のページに進めません。

[Back] ボタンをクリックすると、前のページに戻ることができます。前のページで制約を編集して [Next] をクリックすると、ウィザードはデザインを解析し直し、それに合わせて新しい制約を推奨します。ほとんどの場合、前に推奨されていた制約で変更の影響を受けないものは、そのまま復元されます。推奨された制約を変更せずに前のページを確認しただけの場合は、解析は再実行されないため、無駄なランタイムは発生しません。



重要: Timing Constraints ウィザードを使用して、既存のタイミング制約を編集することはできません。代わりに、[Timing Constraints] ウィンドウを使用する必要があります。

ウィザードで推奨される制約

プライマリ クロック

ウィザードでは、図 2-15 に示すように 2 つのクロック カテゴリが特定されます。

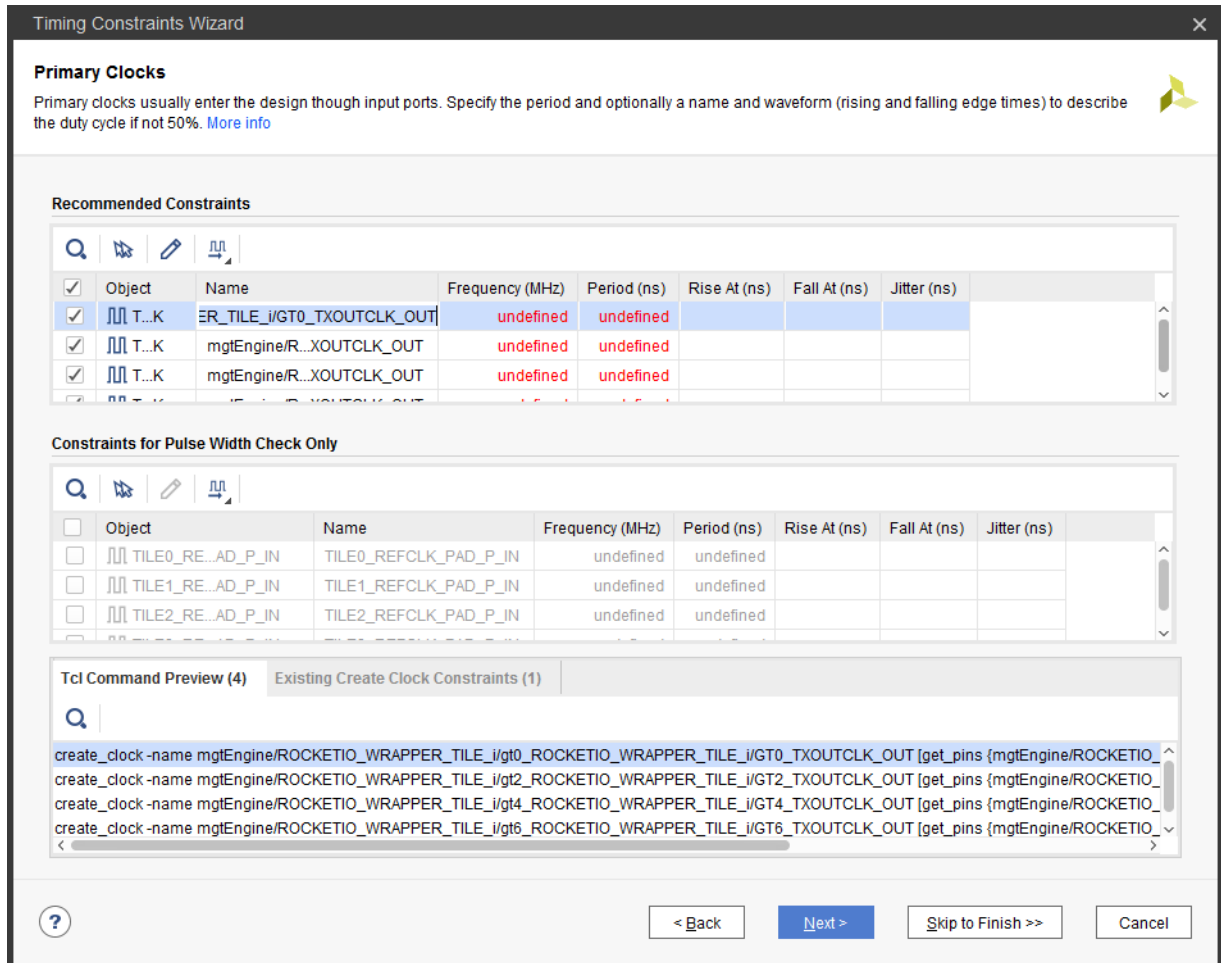


図 2-15: 推奨されるプライマリ クロック

- セットアップ、ホールド、リカバリ、リムーバルチェック用のタイミング スラックを算出するのに必要なプライマリ クロックは、[Recommended Constraints] に表示されます。
- [Constraints For Pulse Width Check Only] には、パルス幅チェック (min_period、max_period、max_skew、min_low_pulse_width、および min_high_pulse_width) を実行するのに必要なクロックのみが表示されます。デフォルトでは、これらのクロックはレポート目的のみに使用され、インプリメンテーション ツールの結果の質には影響しないので、オフになっています。

ウィザードにより、制約の適切なクロックの起点が自動的に特定されます。クロックの起点は一般的には入力クロック ポートですが、タイミングアークのないプリミティブの出力である特殊なケースもあります。たとえば、7 シリーズ デバイスの場合、ウィザードで GT_CHANNEL プリミティブの出力に不足しているプライマリ クロックが特定されます。UltraScale™ デバイスの場合、入力クロックの特性と GT_CHANNEL の設定および接続に基づいて、Vivado Design Suite により GT_CHANNEL 出力クロックが自動的に派生されます。そのため、デザイン境界の GT_CHANNEL セルよりアップストリームにあるプライマリ クロックが推奨されます。

[Generated Clocks] ページ

Timing Constraints ウィザードは、シーケンシャル セルの出力がほかのシーケンシャル セルのクロック ピンを直接またはインターコネクト ロジックを介して駆動する場合、そのシーケンシャル セルの出力に生成クロックの作成を推奨します。PLL または MMCM とは異なり、ユーザー ロジックではマスター クロックの周波数を通倍できないので、[図 2-16](#) に示すように、ウィザードには分周係数を指定するオプションしかありません。

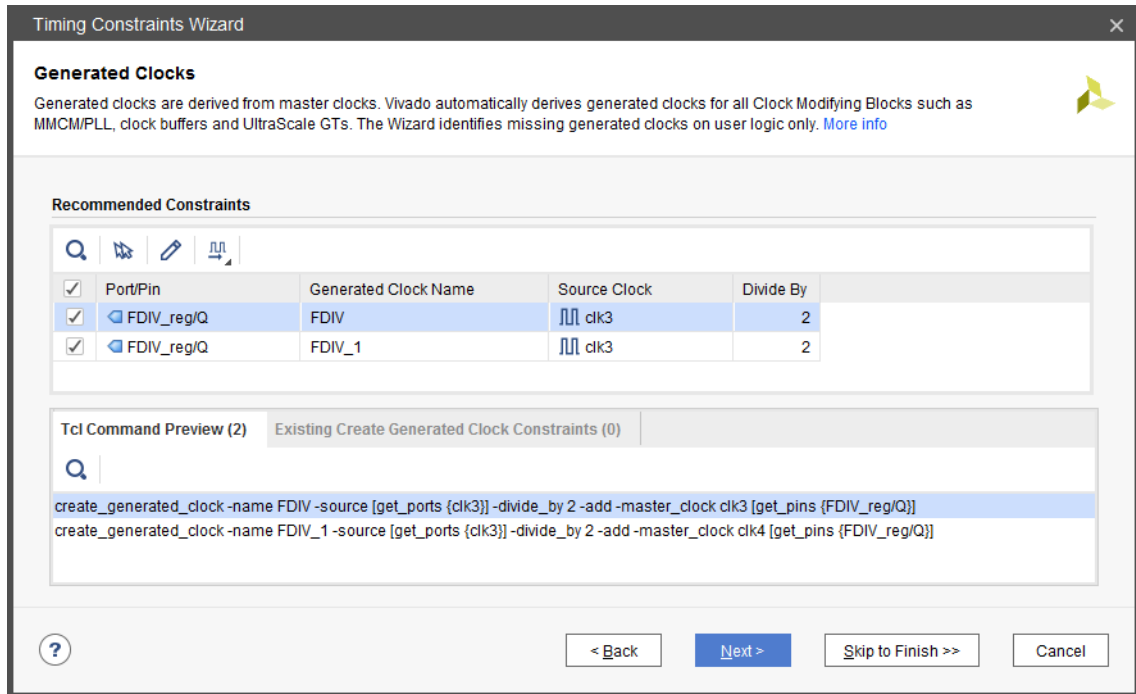


図 2-16: Timing Constraints ウィザードの [Generated Clocks] ページ

複数のマスター クロックが生成クロックの起点に到達すると、対応する生成クロックすべてが、固有の名前と個々のマスター クロックへの明確な参照を使用して生成されます。[図 2-16](#) は、2 つのクロック (clk3 および clk4) がシーケンシャル セル FDIV_reg に到達する場合を示しています。この例では、2 つの生成クロック制約 (FDIV および FDIV_1) が推奨されています。



ヒント: クロック パス上のカスケード接続されたレジスタなどの一部のクロッキング トポロジでは、不足している生成クロックすべてを検出するため、Timing Constraints ウィザードを複数回実行する必要がある場合があります。

[Forwarded Clocks] ページ

定数入力を持つダブル データレート レジスタにより駆動される出力ポートに対して、生成クロック制約が推奨されます。入力される定数の接続に基づいて、生成クロック位相は正 (0 度の位相シフト) または反転 (180 度の位相シフト) のいずれかに調整されます。制約で使用するマスター クロックは、ダブル データレート レジスタのクロックピンに到達するクロックです。図 2-17 の [Recommended Constraints] に示されている [Source Clock] 列を参照してください。

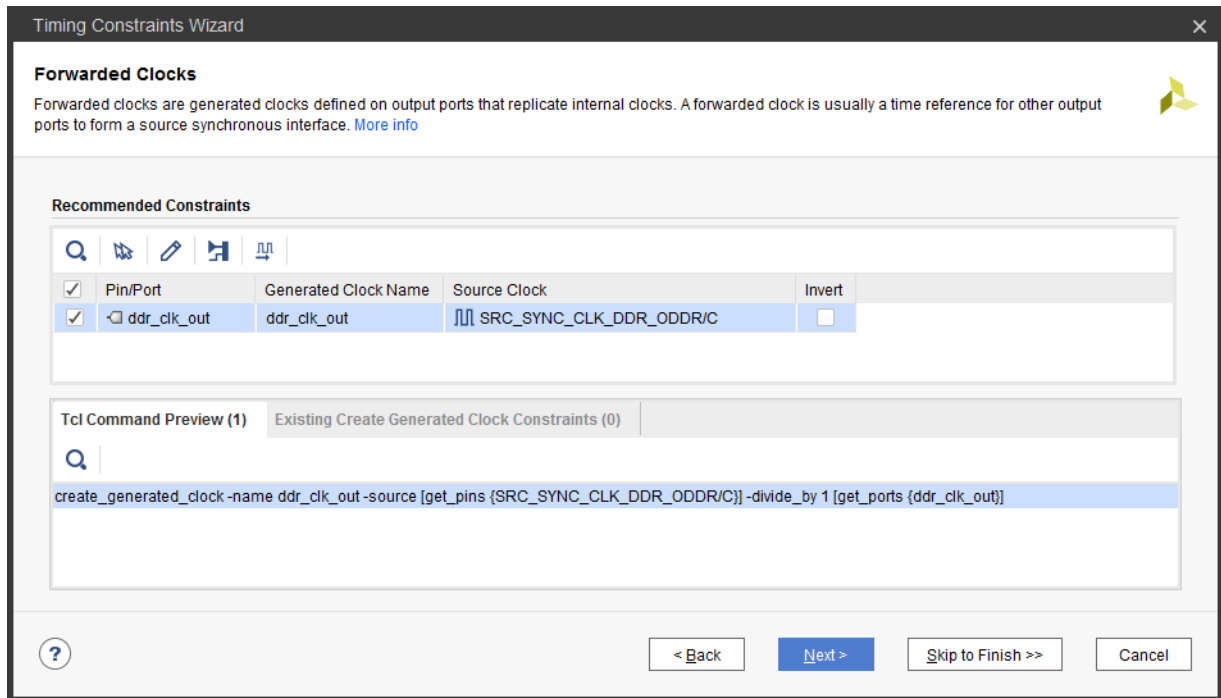


図 2-17: 推奨されるフォワード クロック

7 シリーズ デバイス ファミリに対してウィザードで認識されるトポロジを図 2-18 に示します。マスター クロックまたは出力バッファの特性に制限はありません。

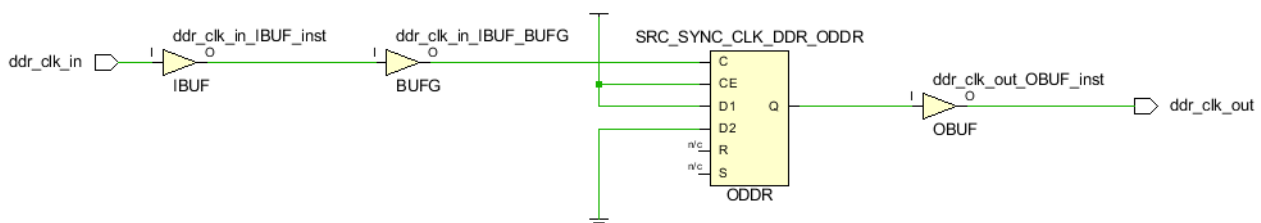


図 2-18: 7 シリーズのフォワード クロックの典型的な回路

UltraScale デバイス ファミリの場合、ODDR および ODDRE1 プリミティブは、プロパティが ODDR_MODE=TRUE に設定された OSERDESE3 に自動的に変更されます。ウィザードで図 2-19 のトポロジが認識され、OSERDESE3/D[0] は 1 に、OSERDESE3/D[4] は 0 (位相シフトなし) に接続されます。

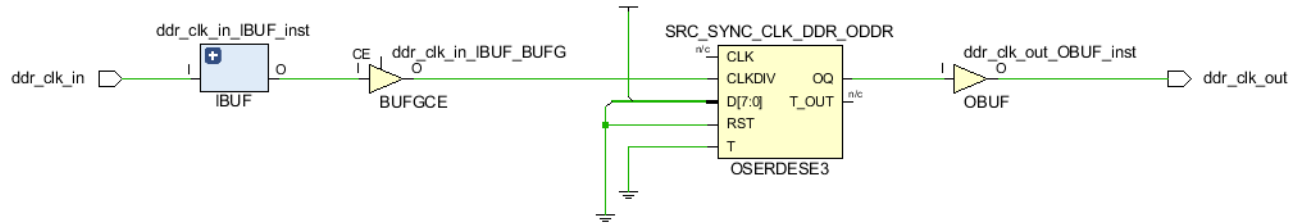


図 2-19: UltraScale のフォワード クロックの典型的な回路

[External Feedback Delays] ページ

Timing Constraints ウィザードでは、デザイン内に含まれる MMCM および PLL セルのフィードバック ループの接続が解析されます。CLKFBIN および CLKFBOUT ピンが、I/O バッファおよび MMCM/PLL プロパティ COMPENSATION=EXTERNAL を使用してデザイン ポートに接続されている場合は、外部遅延制約 (最小と最大) が推奨されます。図 2-20 に、推奨される外部遅延制約の例を示します。

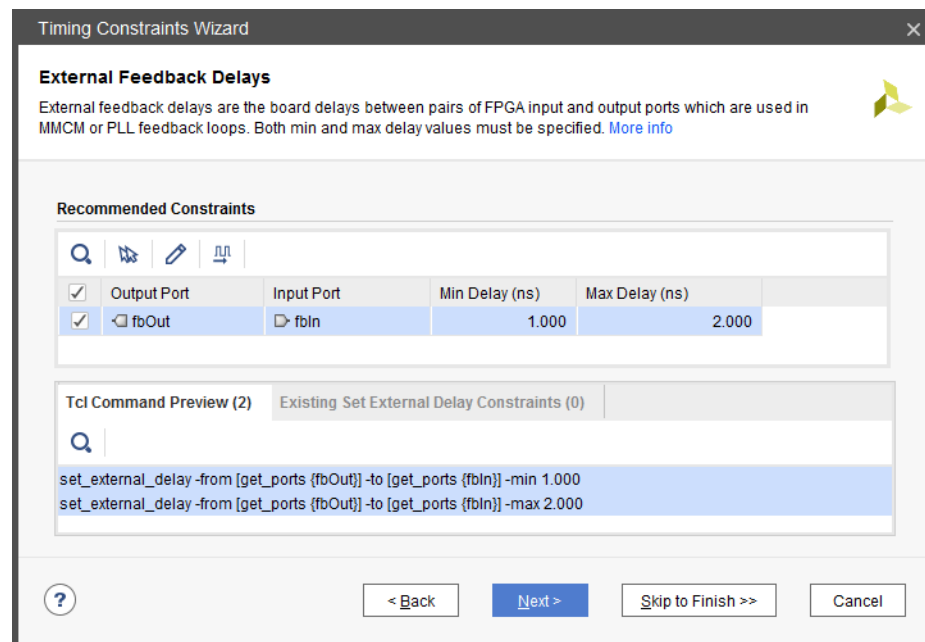


図 2-20: 推奨される外部遅延制約

図 2-21 に、外部フィードバック パス回路の典型的な MMCM を示します。

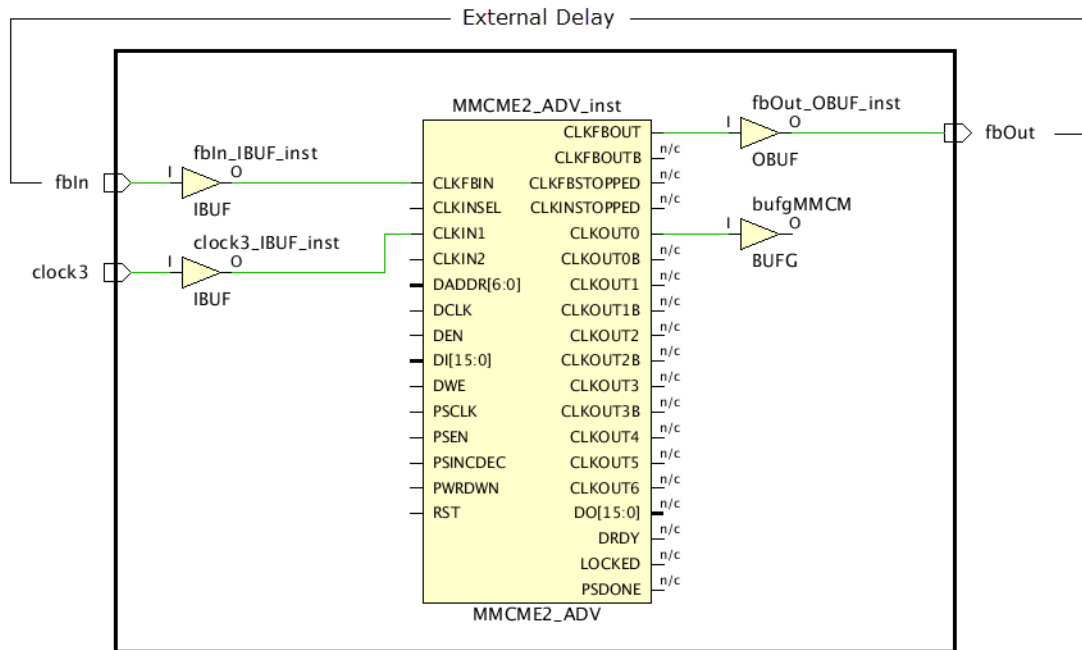


図 2-21: 典型的な MMCM の外部フィードバック パス回路

現在の Vivado Design Suite リリースの Timing Constraints ウィザードでは、フォワード クロックの生成に使用される ODDR など、フィードバック パスにシーケンシャル セルがあると、外部遅延制約を推奨できません。この場合、ウィザード終了後に外部遅延制約を手動で作成するか、[Timing Constraints] ウィンドウを使用して作成する必要があります。

[Input Delays] ページ

Timing Constraints ウィザードでは、入力ポートからのパスがすべて解析され、それらのパスのデザイン内でのデスティネーション クロックとアクティブ エッジが特定されます。この情報を基に、Vivado IDE で提供される XDC テンプレートに基づいて基本的なシステム同期入力遅延制約が推奨されます。テンプレートの詳細は、[52 ページの「XDC テンプレート」](#)を参照してください。[Recommended Constraints] で制約エントリを選択すると、選択されたテンプレートに関連付けられている波形がウィザード ウィンドウ下部にある [Waveform] タブに表示されます。

図 2-22 に、ウィザードで推奨される入力制約の例を示します。これらは部分的にユーザーが変更できます。

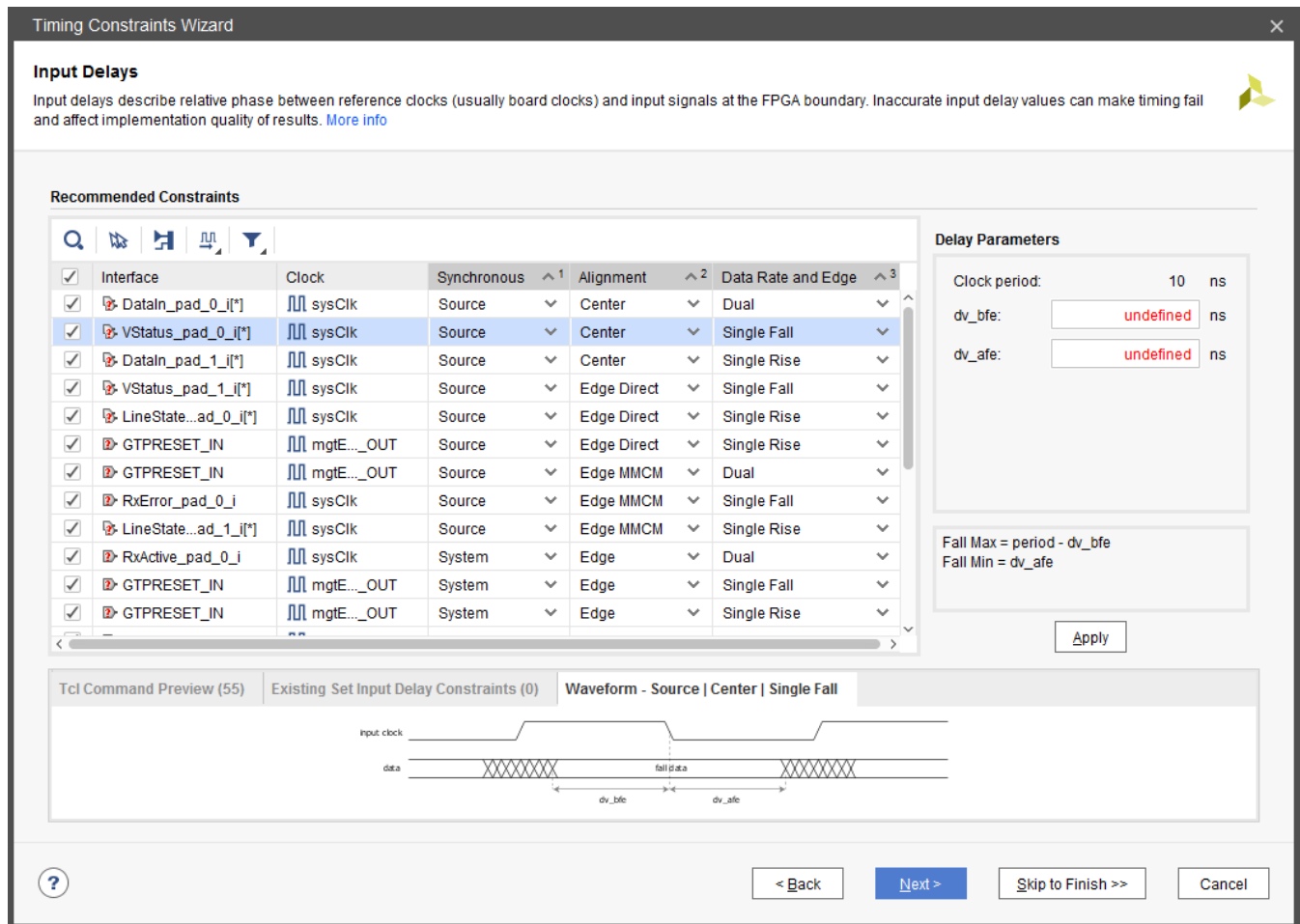


図 2-22: 推奨される入力遅延制約のテンプレート

ボードの実際のインターフェイス タイミングに対応する波形を指定するため、各制約に対して次の 3 つの特性を編集できます。

- [Synchronous]: クロックとデータの間係を指定します。
 - [System] (システム同期インターフェイス用): データが送信されて、1 周期または 1/2 周期ずれたクロック エッジでキャプチャされる場合に使用します。
 - [Source] (ソース同期インターフェイス用): データが送信されて、同じクロック エッジでキャプチャされる場合に使用します。
- [Alignment]: アクティブ クロック エッジに対するデータ遷移アライメントを指定します。
 - システム同期インターフェイスの場合のみ:
 - [Edge]: クロックとデータが同時に遷移する場合に使用します。
 - ソース同期インターフェイスの場合のみ:
 - [Center]: データ有効ウィンドウの真ん中でクロックが遷移する場合に使用します。
 - [Edge Direct]: データ有効ウィンドウの最初でクロックが遷移する場合に使用します。
 - [Edge MMCM]: データ有効ウィンドウの終わりでクロックが遷移する場合に使用します。

- [Data Rate and Edge]: テンプレートで制約が設定されているアクティブ クロック エッジを指定します。ウィザードで推奨されるデフォルト値は、キャプチャ シーケンシャル セルのアクティブ クロック エッジに基づきます。
 - [Single Rise]: 立ち上がりクロック エッジでのみ FPGA の外部にデータが送信される場合に使用します。
 - [Single Fall]: 立ち下がりクロック エッジでのみ FPGA の外部にデータが送信される場合に使用します。
 - [Dual]: 立ち上がりおよび立ち下がりクロック エッジの両方で FPGA の外部にデータが送信される場合に使用します。

推奨されるクロックは、通常、入力パスのシーケンシャル セルに関連付けられているボード クロックです。入力パスの内部クロックが MMCM/PLL 生成クロックの場合、MMCM/PLL を駆動するボード クロックが入力制約の基準クロックとして使用されます。ただし、次のように内部クロック波形とボード クロック波形が異なる場合は例外です。

- 周期が異なる場合

入力制約は、内部クロックと同じ波形の仮想クロックを基準とするので、セットアップ解析は 1 サイクル パス要件で実行されます。仮想クロックは自動的に作成されます。

- 正の位相シフト クロックの場合

仮想クロックが基準クロックとして使用されます。仮想クロックは、ボード クロックと同じ波形を使用して自動的に作成されます。また、仮想クロックと内部クロック間のマルチサイクル パス制約も指定され、デフォルトの解析が 1 周期 + セットアップの位相シフト分に調整されます。仮想クロックとマルチサイクル パスの制約の組み合わせにより Vivado Design Suite タイマーで処理しやすい簡単な制約が提供され、仮想クロックを参照する入力ポートにのみ影響します。

負の位相シフトの場合、デフォルトのセットアップ パス要件は 1 サイクルから位相シフト分を引いた周期なので、仮想クロックとマルチサイクル パス制約は必要ありません。

ウィザード内では、制約に対して選択された基準クロックを変更できません。これには、ウィザード終了後に XDC ファイルを手動で編集するか、[Timing Constraints] ウィンドウを使用して編集する必要があります。

適切なテンプレートを選択したら、ウィザード右側の [Delay Parameters] パネルに遅延パラメーター値を入力し、[Apply] をクリックして入力内容を適用します。

入力遅延の式は、遅延パラメーター フィールドの下に表示されます。図 2-23 に、DDR システム同期インターフェイス テンプレートの [Delay Parameters] パネルを示します。

Delay Parameters

Clock period: 10 ns

trco_min: 0.5 ns

trco_max: 5 ns

tfco_min: ns

tfco_max: undefined ns

trce_dly_min: undefined ns

trce_dly_max: undefined ns

Rise Max = trco_max + trce_dly_max
Rise Min = trco_min + trce_dly_min
Fall Max = tfco_max + trce_dly_max
Fall Min = tfco_min + trce_dly_min

Apply

図 2-23: 入力遅延パラメーターのパネル

クロックとテンプレートが同じ複数の制約を選択して編集すると、遅延パラメーターの入力時間を短縮できます。

制約を入力して適用したら、該当する Tcl 構文を [Tcl Command Preview] タブで確認し、[Next] をクリックして次のページに進みます。



ヒント: Timing Constraints ウィザードでは、フォルス パス制約が設定された入力ポートは無視されます。これは、通常デザインのどのクロックとも既知の位相関係がない非同期リセットを無視する場合などに特に便利です。フォルス パス制約は、ウィザード外でしか作成できません。

[Output Delays] ページ

Timing Constraints ウィザードでは、入力遅延と同様、すべての出力ポートへのパスが解析され、それらのパスのデザイン内でのソース クロックとアクティブ エッジが特定されます。テンプレートの選択規則は、「[Input Delays] ページ」と同じです。図 2-24 に、ウィザードで推奨される出力制約の例を示します。これらは部分的にユーザーが変更できます。

図 2-24: 推奨される出力遅延制約のテンプレート

ボードの実際のインターフェイス タイミングに対応する波形を指定するため、各制約に対して次の 3 つの特性を編集できます。

- [Synchronous]: クロックとデータの間を指定します (詳細は 31 ページの「[Input Delays] ページ」を参照)。
- [Alignment]: アクティブ クロック エッジに対するデータ遷移アライメントを指定します。
 - [Setup/Hold]: FPGA 外でのデータ有効ウィンドウのタイミング特性に基づいてテンプレート遅延パラメーターを指定する場合に使用します。
 - [Skew] (ソース同期のみ): FPGA の出力ピンのスキュー要件に基づいてテンプレート遅延パラメーターを指定する場合に使用します。

- [Data Rate and Edge]: テンプレートで制約が設定されているアクティブ クロック エッジを指定します (詳細は 31 ページの「[Input Delays] ページ」を参照)。

推奨される入力遅延制約と同様、基準クロックは通常ボード クロックになりますが、次の場合は例外です。

- ボード クロックと出力パス内部クロックの周期が異なる場合。

出力制約は、内部クロックと同じ波形の仮想クロックを基準とするので、セットアップ解析は 1 サイクル パス要件で実行されます。仮想クロックは自動的に作成されます。

- ボード クロックに対する出力パス内部クロックの位相シフトが負の場合。

仮想クロックが基準クロックとして使用されます。仮想クロックは、ボード クロックと同じ波形を使用して自動的に作成されます。また、仮想クロックと内部クロック間のマルチサイクル パス制約も指定され、デフォルトの解析が 1 周期 + セットアップの位相シフト分に調整されます。仮想クロックとマルチサイクル パスの制約の組み合わせにより Vivado Design Suite タイマーで処理しやすい簡単な制約が提供され、仮想クロックを参照する出力ポートにのみ影響します。

注記: 正の位相シフトの場合、デフォルトのセットアップ パス要件が 1 サイクルから位相シフト量を引いた値なので、仮想クロックとマルチサイクル パス制約は必要ありません。

- 共有クロック接続に基づく出力パスのタイミング指定にフォワード クロックが特定されている場合。

フォワード クロックは、ウィザードの 3 段階目の [Forwarded Clocks] で作成されているはずです。作成されなかった場合は、ボード クロックまたは仮想クロックが出力遅延制約の基準クロックとして使用されます。

図 2-25 に、7 シリーズ ファミリのフォワード クロックを使用した出力ソース同期パスの基本的な例を示します。ODDR/OSERDES インスタンスは、どちらも同じクロック ネットに接続されます (青のハイライト部分)。clk_vsf_clk_2 生成クロックは、既に vsf_clk_2 出力ポートに定義されています。

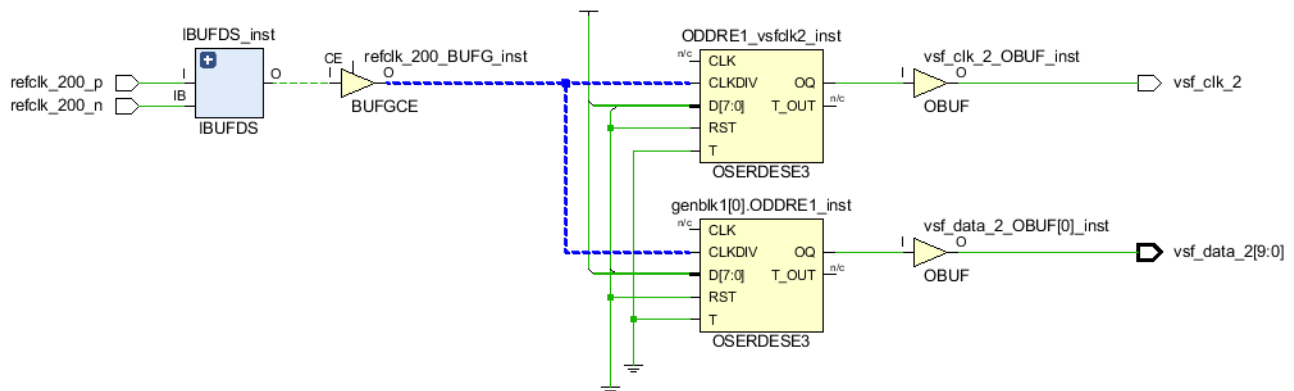


図 2-25: フォワード クロックを使用したソース同期出力パスの例

図 2-26 に、ウィザードの該当する制約を示します。

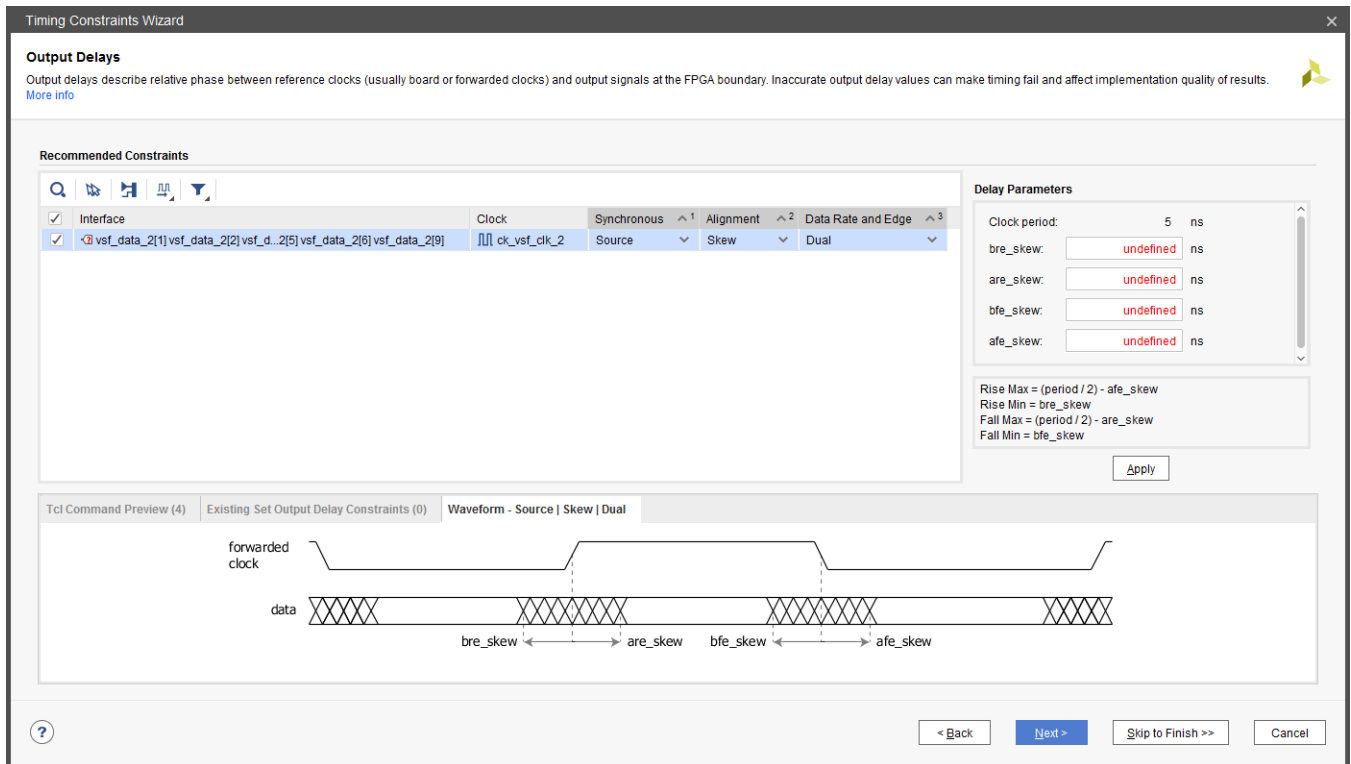


図 2-26: フォワード クロックを使用した推奨されるソース同期出力パス遅延制約

適切なテンプレートを選択したら、遅延パラメーター値を入力する必要があります。クロックとテンプレートが同じ複数の制約を選択して編集すると、遅延パラメーターの入力時間を短縮できます。制約を入力して適用したら、該当する Tcl 構文を [Tcl Command Preview] タブで確認し、[Next] をクリックして次のページに進みます。



ヒント: Timing Constraints ウィザードでは、フォルス パス制約が設定された出力ポートは無視されます。フォルス パス制約は、ウィザード外でしか作成できません。

[Combinatorial Delays] ページ

パスの中には、デバイス内でシーケンシャルセルにより取り込まれず、入力ポートから出力ポートに直接伝搬されるものがあります。入力ポートが出力ポートとシーケンシャルセルの両方に接続されている場合、Timing Constraints ウィザードでは、入力ポートには [Input Delay] ページで制約が設定されているはずなので、入力/出力ポート ペア間に組み合わせ制約は推奨されません。組み合わせパスに対しては、図 2-27 に示すように、デザインポートの入力遅延および出力遅延と共に仮想クロックを定義するように推奨されます。

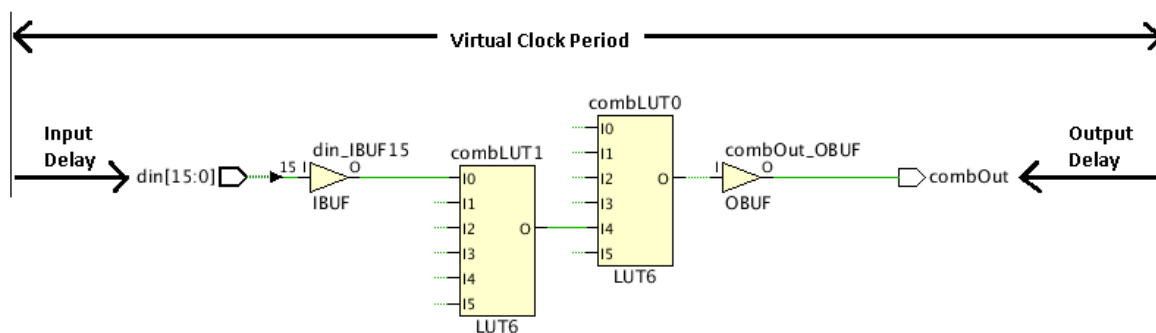


図 2-27: 組み合わせパスの回路図および遅延制約

最終的な組み合わせパス遅延制約は、次のようになります。

- セットアップ解析:
 仮想クロックの周期 - 最大入力遅延 - 最大出力遅延
- ホールド解析:
 0 - 最小出力遅延 - 最小入力遅延

仮想クロックの周期は、制約が設定されたすべての組み合わせパスの中で最大の組み合わせ遅延制約より大きくなるように変更する必要があります。図 2-28 は、入力/出力ポート ペアごとに必要な遅延入力を示しています。

Timing Constraints Wizard

Combinational Delays

Combinational constraints cover paths that traverse the FPGA without being captured by any sequential elements. A virtual clock defines the reference board clock, and input/output delays describe external delays on combinational paths. [More info](#)

Recommended Constraints

Virtual clock period (ns): 8.0 Command: create_clock -period 8.000 -name virtual_clock

	Input Port	Output Port	Input Min Delay (ns)	Input Max Delay (ns)	Output Min Delay (ns)	Output Max Delay (ns)
<input checked="" type="checkbox"/>	in4	out1	0.010	2.000	1.250	1.500
<input checked="" type="checkbox"/>	in5	out2	0.020	2.500	2.250	1.500
<input checked="" type="checkbox"/>	in8	out3	0.010	1.950	1.500	1.700

Tcl Command Preview (12) Existing Set Input Delay, Set Output Delay Constraints (0)

```

set_input_delay -clock [get_clocks {virtual_clock}] -min -add_delay 0.01 [get_ports {in4}]
set_input_delay -clock [get_clocks {virtual_clock}] -max -add_delay 2.0 [get_ports {in4}]
set_output_delay -clock [get_clocks {virtual_clock}] -min -add_delay 1.25 [get_ports {out1}]
set_output_delay -clock [get_clocks {virtual_clock}] -max -add_delay 1.5 [get_ports {out1}]
set_input_delay -clock [get_clocks {virtual_clock}] -min -add_delay 0.02 [get_ports {in5}]
set_input_delay -clock [get_clocks {virtual_clock}] -max -add_delay 2.5 [get_ports {in5}]
set_output_delay -clock [get_clocks {virtual_clock}] -min -add_delay 2.25 [get_ports {out2}]
set_output_delay -clock [get_clocks {virtual_clock}] -max -add_delay 1.5 [get_ports {out2}]
set_input_delay -clock [get_clocks {virtual_clock}] -min -add_delay 0.01 [get_ports {in8}]
set_input_delay -clock [get_clocks {virtual_clock}] -max -add_delay 1.95 [get_ports {in8}]
set_output_delay -clock [get_clocks {virtual_clock}] -min -add_delay 1.5 [get_ports {out3}]
set_output_delay -clock [get_clocks {virtual_clock}] -max -add_delay 1.7 [get_ports {out3}]

```

Buttons: ? < Back Next > Skip to Finish >> Cancel

図 2-28: 推奨される組み合わせパス遅延

入力および出力遅延制約が既存のものより優先されることはありません。同じクロックに対して遅延制約が複数設定されているポートがある場合、その中の最小値が Vivado タイミング解析のホールド解析に使用され、最大値がセットアップ解析に使用されます。

すべての遅延を入力したら、[Next] をクリックして次のページに進みます。



ヒント: Timing Constraints ウィザードを使用せずに、set_max_delay および set_min_delay コマンドを使用して、組み合わせパスに制約を設定することもできます。

[Physically Exclusive Clock Groups] ページ

物理的に排他的なクロックは、同じ起点で定義され、同じクロック ツリーで伝搬されるクロックです。図 2-29 に、同じ入力ポートで定義されている 2 つのプライマリ クロックの例を示します。

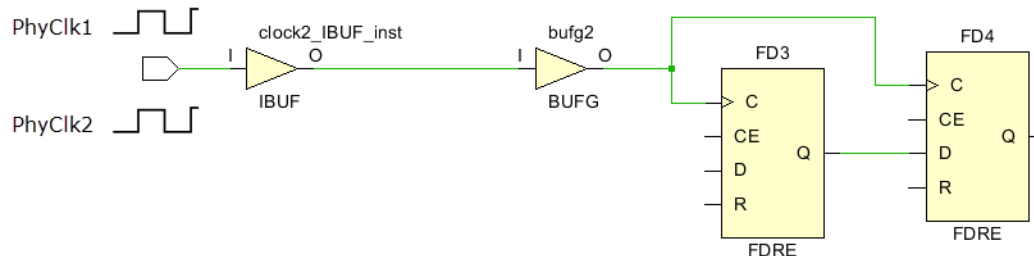


図 2-29: 物理的に排他的なクロックを含むデザイン例

このようなオーバーラップは、1 つのデザインと制約データベースを使用して複数のアプリケーション モードのタイミング解析を実行するには便利ですが、これらのクロックとその派生クロック間でタイミング解析を実行するべきではありません。Timing Constraints ウィザードでは、クロック乗せ換えパスで不要なタイミング解析が実行されないようにするため、図 2-30 に示すように、このようなクロックが特定され、クロック グループ制約が推奨されます。

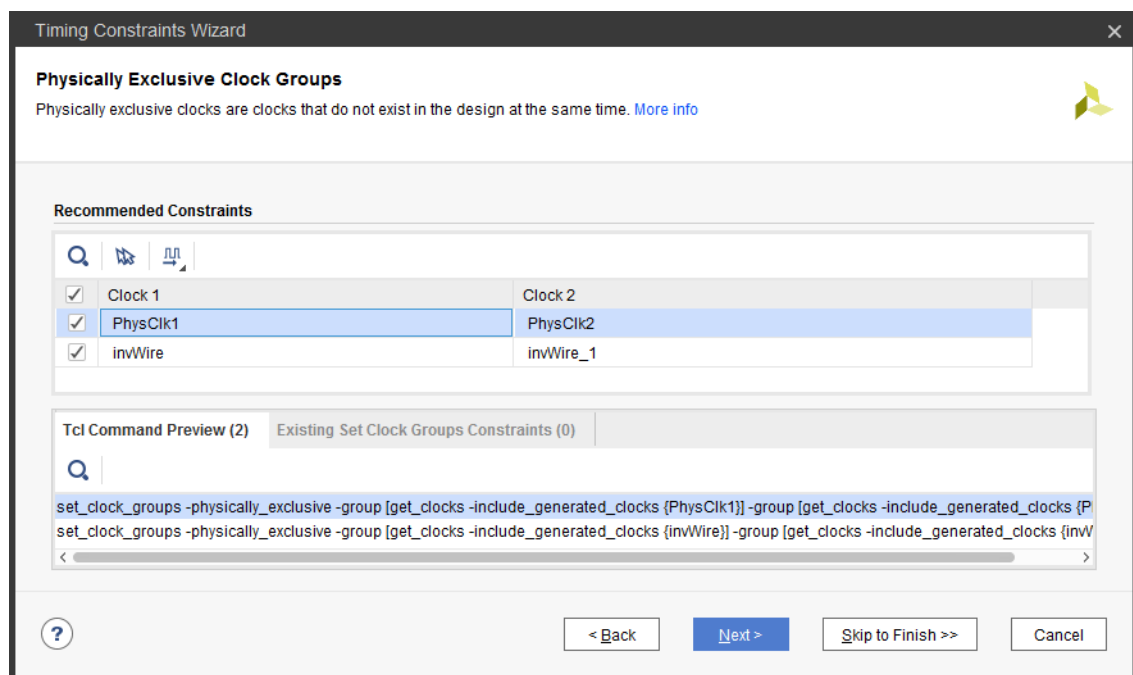


図 2-30: クロック グループ制約を使用するデザイン例

[Logically Exclusive Clock Groups with No Interaction]

論理的に排他的なクロックは異なる起点に定義されるクロックですが、マルチプレクサーまたはその他の組み合わせロジックのため、クロック ツリーの一部が共有されます。Timing Constraints ではこのようなクロックが特定され、それらのクロック間に共有クロック ツリーに接続されているロジックを除いてタイミング パスがない場合は、直接クロック グループ制約を設定することが推奨されます。図 2-31 に、clkA と clkB の 2 つのクロックの例を示します。入力ポートはそれぞれ別ですが、BUFGMUX の出力からオーバーラップしています。

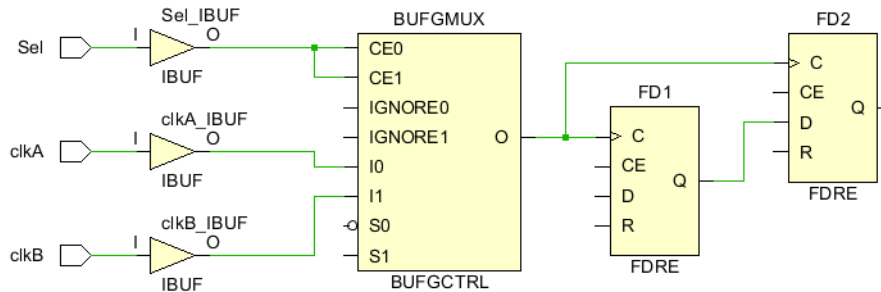


図 2-31: 論理的に排他的なクロック グループ (クロック関連性なし) の例

[Logically Exclusive Clock Groups with Interaction] ページ

Timing Constraints ウィザードでは、共有クロック ツリーに接続されているロジック以外にクロック間にタイミング パスがある論理的に排他的なクロックが特定されます。図 2-32 の clkA と clkB には、クロック ツリーに共有部分があり、共有クロック ツリーから clkA のみへのタイミング パスもあります。

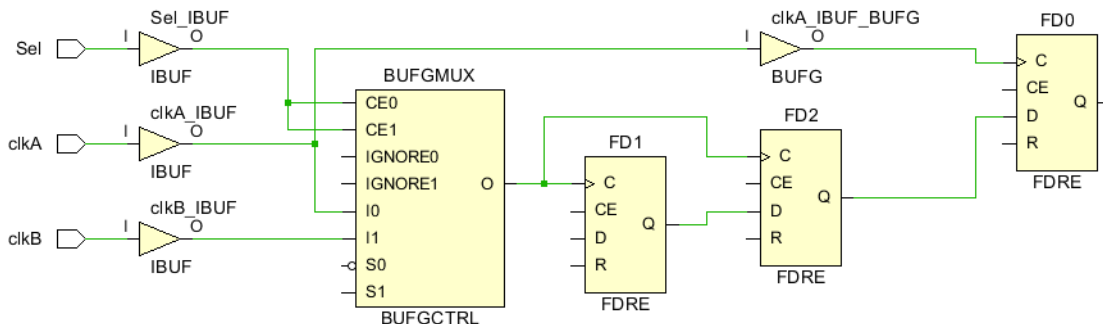


図 2-32: 論理的に排他的なクロック (クロック関連性あり) を含むデザイン例

共有クロック ツリーのクロック乗せ換えパスのみを無視する必要があるので、clkA および clkB のコピーで、共有クロック ツリーにのみ存在する生成クロックを作成するよう推奨されます。クロック グループ制約は生成クロックのみに適用されるので、共有クロック ツリーのロジック外のパスは通常どおりタイミング解析されます。図 2-33 に、上記の例に対してウィザードで推奨される制約を示します。

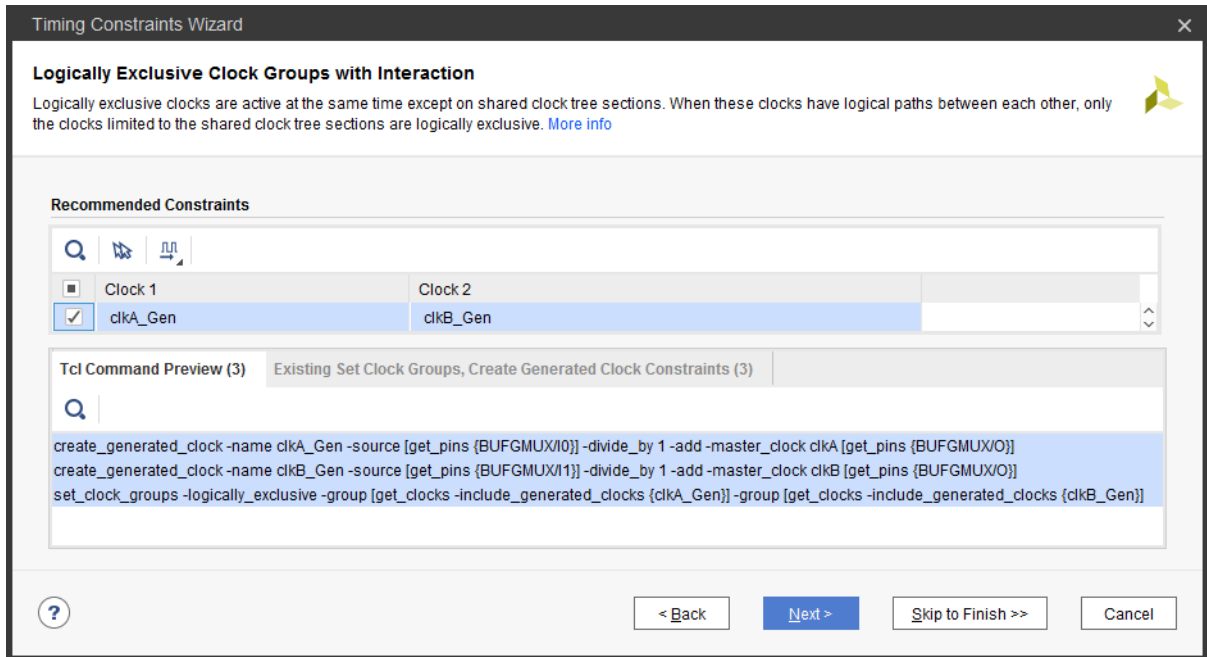


図 2-33: 論理的に排他的なクロック (クロック関連性あり) に推奨される制約

[Asynchronous Clock Domain Crossings] ページ

Timing Constraints ウィザードでは、非同期クロック間のクロック乗せ換え (CDC) パスのトポロジが解析され、それが安全であると判断された場合は、クロック グループまたはフォルス パスが推奨されます。

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。このため、非同期 CDC パスのスラックは正確に算出できません。非同期 CDC パスがタイミング解析されると、非同期クロック間のスキューが大きくなる可能性があるため、タイミングの結果の質にかなり影響し、タイミング クロージャが達成できない可能性があります。このため、ユーザーがこれらのパスに `set_clock_groups`、`set_false_path`、または `set_max_delay -datapath_only` などのタイミング例外を追加して、タイミング解析を完全に無視するか、クロック スキューとクロックのばらつきを無視するように設定する必要があります。また、メタステーブル状態を回避するため、デザインに正しい CDC 回路をインプリメントする必要があります。

Vivado Design Suite では、同期データおよび非同期リセットに対するフリップフロップ ベースのシンクロナイザーのみが特定されます。このようなシンクロナイザーの例については、『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) [参照 4] を参照してください。図 2-34 に、推奨される制約と推奨されない制約の表の例を示します。

Timing Constraints Wizard

Asynchronous Clock Domain Crossings

Asynchronous clock domain crossings occur when data is transferred between two clocks that do not have a known phase relationship. Synchronizers with ASYNC_REG property set to true are recommended on these paths. [More info](#)

Recommended Constraints (clock domain crossings are safe)

<input checked="" type="checkbox"/>	Source Clock	Destination Clock	Constraint	Endpoints	Safe	Unsafe/Unknown	No ASYNC_REG	Max Delay Datapath Only
<input checked="" type="checkbox"/>	clock3a	clock1a	asynch (clock group)	1	1	0	1	0
<input checked="" type="checkbox"/>	clock1a	clock3a	asynch (clock group)	4	4	0	4	0
<input checked="" type="checkbox"/>	clock4a	clock5a	asynch (false path)	1	1	0	1	0

Non-recommended Constraints (missing synchronizer and/or set_max_delay -datapath_only already exists)

<input type="checkbox"/>	Source Clock	Destination Clock	Constraint	Endpoints	Safe	Unsafe/Unknown	No ASYNC_REG	Max Delay Datapath Only
<input type="checkbox"/>	clkB	clkA	Timed - No Common Primary Clock	2	0	2	0	0
<input type="checkbox"/>	clkA	clkB	Timed - No Common Primary Clock	1	0	1	0	0
<input type="checkbox"/>	clock1a	clock2a	Timed - No Common Primary Clock	1	0	1	0	0

Tcl Command Preview (12) Existing Set Clock Groups, Set False Path, Set Bus Skew Constraints (0) Timing Paths

```

set_property ASYNC_REG true [get_cells {Nosync2a Nosync3a}]
set_property ASYNC_REG true [get_cells {Nosync2c Nosync3c}]
set_property ASYNC_REG true [get_cells {Nosync2b Nosync3b}]
set_clock_groups -asynchronous -group [get_clocks {clk_no_synca}] -group [get_clocks {clk_no_syncb}]
set_property ASYNC_REG true [get_cells {FD11a FD12a}]
set_clock_groups -asynchronous -group [get_clocks {clock3a}] -group [get_clocks {clock1a}]
set_property ASYNC_REG true [get_cells {FDcdc2c FDcdc3c}]

```

?

< Back

Next >

Skip to Finish >>

Cancel

図 2-34: 推奨される制約と推奨されない制約の表の例

どちらの表にも、次の情報が表示されます。

- [Source Clock]: ウィザードで特定された CDC パスの始点のクロック。
- [Destination Clock]: ウィザードで特定された CDC パスの終点のクロック。
- [Constraint]: 優先度の高いタイミング例外、例外がない場合はクロック関係の特性。
 - [Recommended Constraints] の表には、制約が作成されると想定されて新しい制約が表示されます。
 - [asynch (clock groups)]: 両方向のタイミングを無視するのが安全な場合。この場合は set_clock_groups 制約が作成されます。
 - [asynch (false path)]: 一方方向のパスを無視することのみが安全な場合。この場合は set_false_path 制約が作成されます。

- [Non-recommended Constraints] の表には、クロック グループまたはフォルス パス例外を適用する前に CDC パスに設定されているタイミング制約が示されます。
 - [Timed - No Common Primary Clock]
 - [Timed - No Common Period]
 - [MaxDelay DataPath]: 少なくとも 1 つのパスに `set_max_delay -datapath_only` 制約が設定されており、それ以外のパスにフォルス パス制約が設定されている場合
- [Endpoints]: ウィザードで特定された CDC パスの終点の数。
- [Synchronized (with ASYNC_REG)]: すべてのシンクロナイザー フリップフロップで `ASYNC_REG` プロパティが `true` に設定された状態で、正しく同期された終点の数。
- [Synchronizer without ASYNC_REG]: 少なくとも 1 つのフリップフロップで `ASYNC_REG` プロパティが `true` に設定されていない場合のシンクロナイザーの数。
- [Unknown]: ウィザードでシンクロナイザーが検出されなかった CDC パスの終点の数。
- [Max Delay Datapath Only]: `set_max_delay -datapath_only` 制約が設定されている CDC パスの終点の数。

表には、可能な場合はクロスプローブ リンクが含まれます。数値をクリックすると、該当する CDC パスがウィンドウ下部の [Paths] タブにリストされます。CDC パスを 1 つまたは複数選択し、[Schematic] ボタンをクリックする (F4 キーを押す) と、メインの Vivado IDE ウィンドウに選択したパスのロジックを表示できます。

推奨される非同期クロック グループ制約

Timing Constraints ウィザードでは、次の条件が満たされる場合、2 つのクロック間に `set_clock_groups -asynchronous` 制約を設定することが推奨されます。

- すべてのパスの両方向にシンクロナイザーが含まれる。
- どちらの方向にも `set_max_delay -datapath_only` が設定されたパスがない場合 (`set_clock_groups` の方が既存の `set_max_delay` よりも優先される)。

推奨されない非同期クロック グループ制約

Timing Constraints ウィザードでは、次のいずれかの理由から推奨されないため、デフォルトではディスエーブルになっている制約が示されます。

- 少なくとも 1 つのパスのいずれかの方向にシンクロナイザーがない。
- 少なくとも 1 つのパスのいずれかの方向に `set_max_delay -datapath_only` が設定されている。

デザインの初期のバージョンではこれらの制約をイネーブルにし、最終的なデザインを作成するときに CDC パスおよびこれらの制約を見直すことができます。

CDC シンクロナイザーおよび ASYNC_REG プロパティ

合成およびインプリメンテーション中に実行されるロジック最適化でシンクロナイザー セルが削除されないようにし、MTBF (平均故障間隔) が最大になるよう配置が最適化されるようにするため、すべてのシンクロナイザー フリップフロップで `ASYNC_REG` プロパティを `true` に設定することをお勧めします。両方の表でデフォルトまたはユーザーによりイネーブルになっているクロック グループ制約に対しては、ウィザードにより `ASYNC_REG` プロパティがすべて `TRUE` に設定されます。

`ASYNC_REG` 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) [参照 11] を参照してください。

CDC 解析および制約の完了

有効な CDC トポロジであっても、単純なシンクロナイザーをベースにしていないものは Timing Constraints ウィザードでは認識されません。report_cdc コマンドを使用すると、CDC パスのより包括的なレポートが表示され、構造的な修正が必要なものが示されます。report_cdc 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) [参照 4] を参照してください。

set_max_delay -datapath_only が存在するためにウィザードで制約が推奨されない場合は、通常はタイミング解析が実行されるその他の CDC パスを個別に確認する必要がある、必要に応じてフォルス パス制約を追加して無視されるようにします。ポイント ツー ポイントのフォルス パス制約は、ウィザード終了後に Tcl コンソールまたは [Timing Constraints] ウィンドウで XDC ファイル内に作成する必要があります。

[Constraints Summary] ページ

Timing Constraints ウィザードの最後のページには、[Finish] をクリックした後に適用され、ターゲット XDC ファイルの最後に保存される新しい制約が表示されます。各ハイパーリンクをクリックすると、その制約の詳細を表示できます。図 2-35 に、[Clock Summary] ページの例を示します。

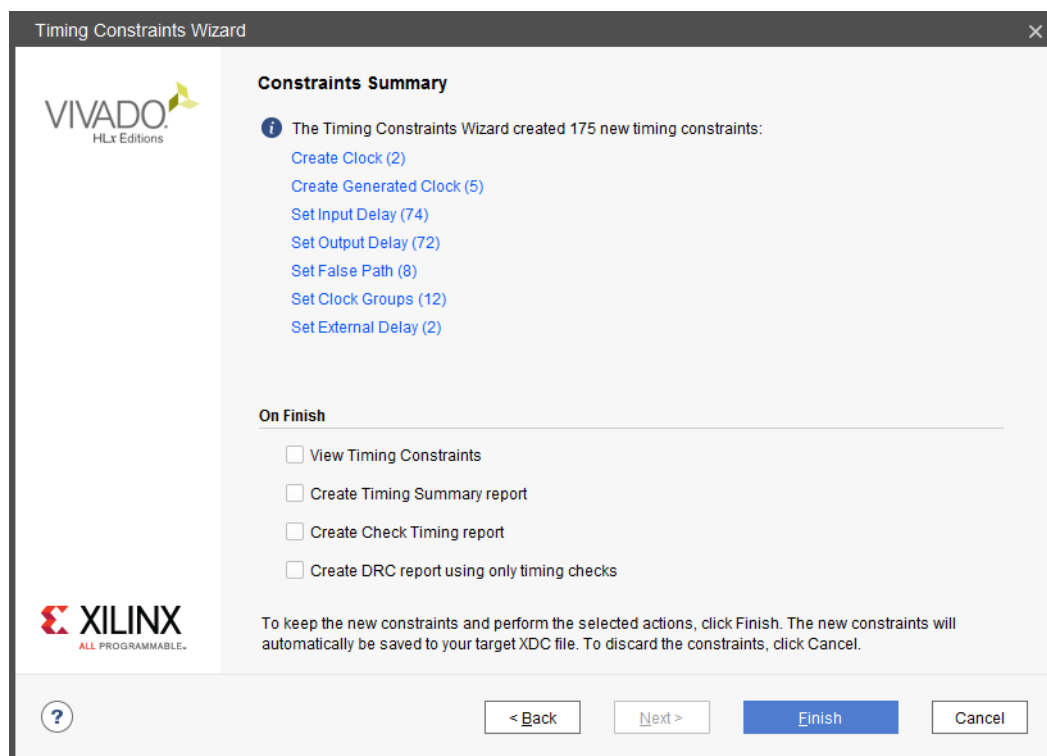


図 2-35: [Clock Summary] ページの例

[Timing Constraints] ウィンドウ

[Timing Constraints] ウィンドウは、合成済みデザインおよびインプリメント済みデザインでのみ使用可能です。エラーボレート済みデザインの制約では、XDC ファイルを直接使用および編集する必要があります。詳細は、[54 ページの「合成制約の作成」](#)を参照してください。

[Timing Constraints] ウィンドウを開くには、[図 2-36](#) に示す 3 つのオプションのいずれかを使用します。

- [Window] → [Timing Constraints] をクリックします。
- Flow Navigator で [Synthesis] → [Synthesized Design] → [Edit Timing Constraints] をクリックします。
- Flow Navigator で [Implementation] → [Implemented Design] → [Edit Timing Constraints] をクリックします。

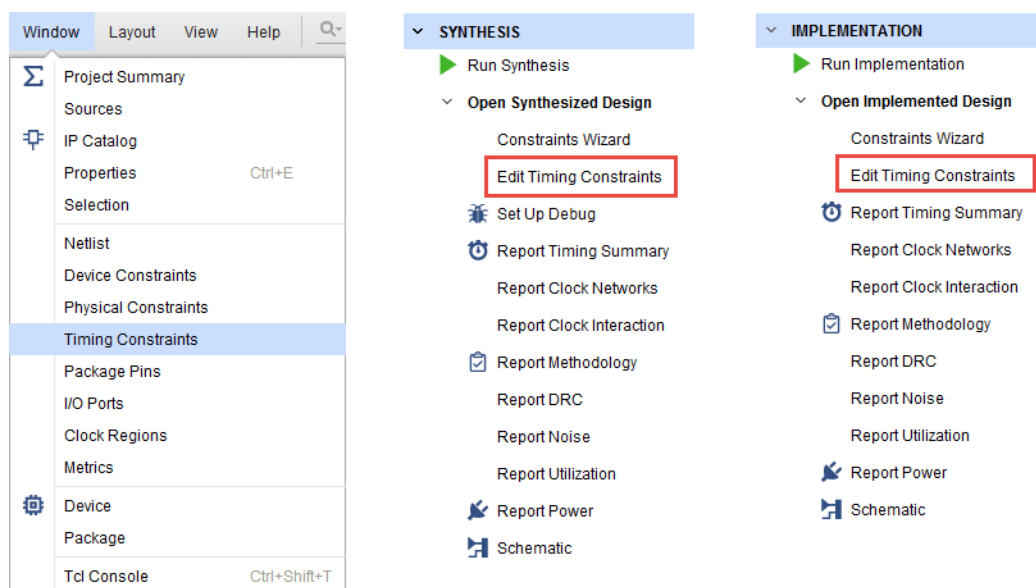


図 2-36: [Timing Constraints] ウィンドウを開く方法

[Timing Constraints] ウィンドウには、メモリ内のタイミング制約が XDC ファイルおよび Tcl スクリプトで記述されているのと同じ順序、または [Tcl Console] ウィンドウで入力したのと同じ順序で表示されます。

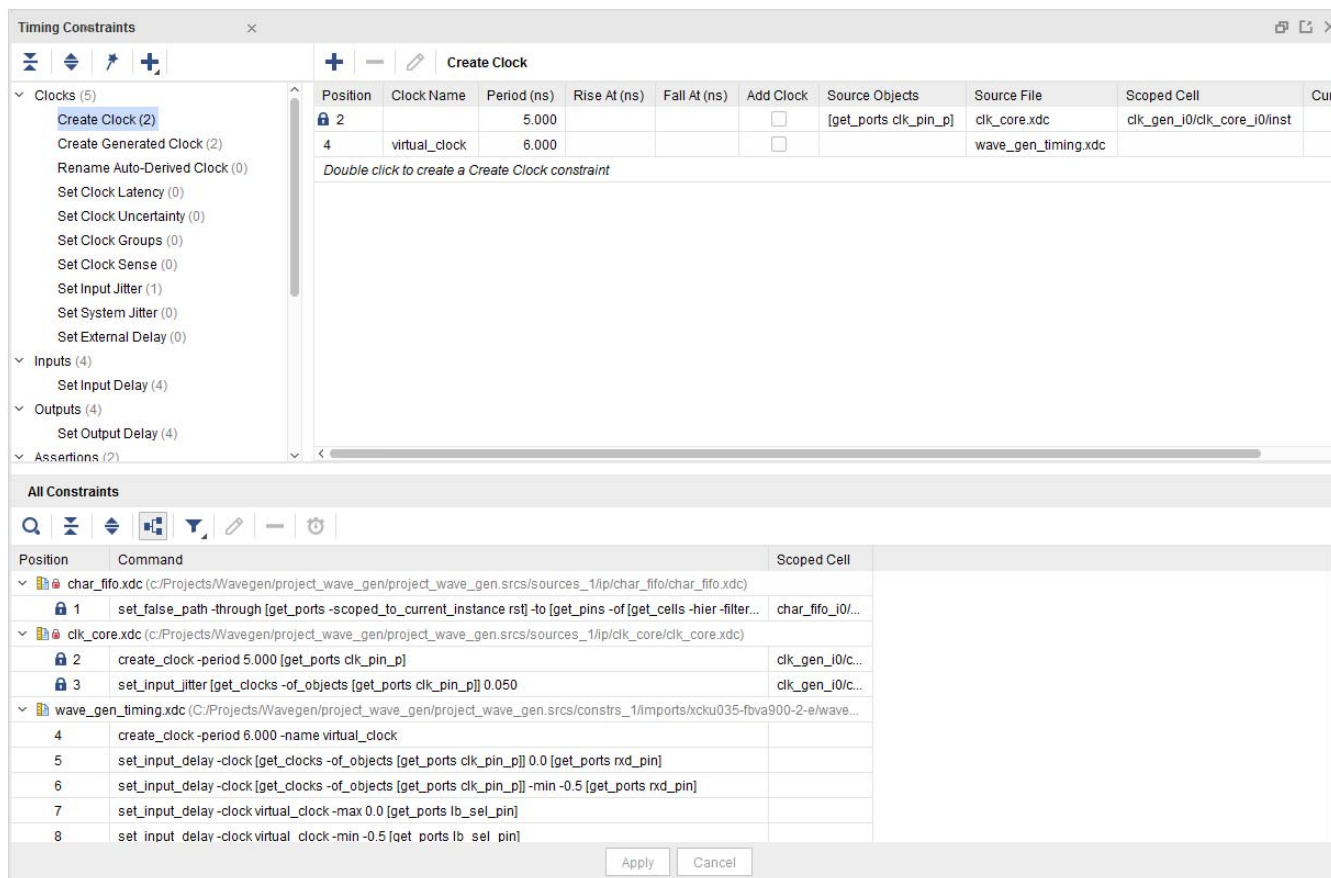





図 2-37: [Timing Constraints] ウィンドウ

一部の制約は、このウィンドウからは編集できません。それらの制約には、編集できないことを示すアイコン が表示されます。

タイミング制約の表

[Timing Constraint] ウィンドウには、既存のタイミング制約の詳細が表形式で示されます。この表を利用して、タイミング制約のオプションを確認および変更します。




 Create Clock

Position	Clock Name	Period (ns)	Rise At (ns)	Fall At (ns)	Add Clock	Source Objects	Source File	Scoped Cell	Current Instance
2		5.000			<input type="checkbox"/>	[get_ports clk_pin_p]	clk_core.xdc	clk_gen_i0/clk_core_i0/inst	
4	virtual_clock	6.000			<input type="checkbox"/>		wave_gen_timing.xdc		

Double click to create a Create Clock constraint

図 2-38: タイミング制約の表

この表には、2 つの列があります。

- [Source File]: 制約が含まれる XDC ファイルまたは Tcl スクリプトの名前
- [Scoped Cell]: 制約が適用されたときの現在のインスタンス名。この名前は通常、専用制約が付属している IP インスタンスに対応します。詳細は、[62 ページの「制約の適用範囲の設定」](#)を参照してください。

表の最後の行をダブルクリックすると、選択したタイプの制約を新しく作成できます。制約を作成するダイアログボックスが表示され、新しい制約の詳細を入力できます。[OK] をクリックして制約をメモリに適用し、ダイアログボックスを閉じます。表の新しい行に新しい制約の情報が表示されます。

既存の制約を編集するには、表の値を直接変更します。編集が完了したら、[Apply] をクリックして制約の変更をメモリに適用します。



重要: 新しい制約または編集した制約を適用しても、XDC ファイルには保存されません。XDC ファイルに保存するには、[Save Constraints] をクリックする必要があります。



重要: IP 制約は、編集または削除できません。IP に含まれる制約を修正するには、該当する IP の XDC ファイルをディスエーブルにして、XDC ファイルへの制約をコピーし、必要に応じて制約を編集する必要があります。

制約の作成 (カテゴリ別)

[Timing Constraints] ウィンドウの左上のペインで制約のカテゴリを選択すると、右側のペインにそのカテゴリの制約の表が表示されます。これにより、同じカテゴリの既に作成されている制約を確認できます。

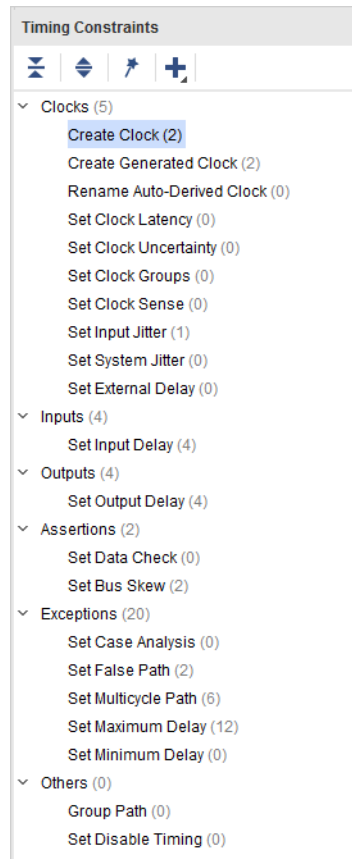


図 2-39: タイミング制約のカテゴリ

制約を作成するには、カテゴリ名をダブルクリックします。各オプションの値を指定するダイアログ ボックスが表示されます。[OK] をクリックすると、ツールで次の処理が実行されます。

1. 構文が検証されます。
2. 構文がメモリに適用されます。
3. 新しい制約が表の最後に追加されます。
4. 新しい制約が全制約のリストの最後に追加されます。

すべての制約

[Timing Constraints] ウィンドウの下部には、メモリに読み込まれているすべての制約が、制約が適用された順に表示されます。制約は、元の XDC ファイルまたは Tcl スクリプトごとにグループ化されます。XDC ファイルが特定の階層セルのみに適用される場合は、ファイル名の横にセル名が表示されます。

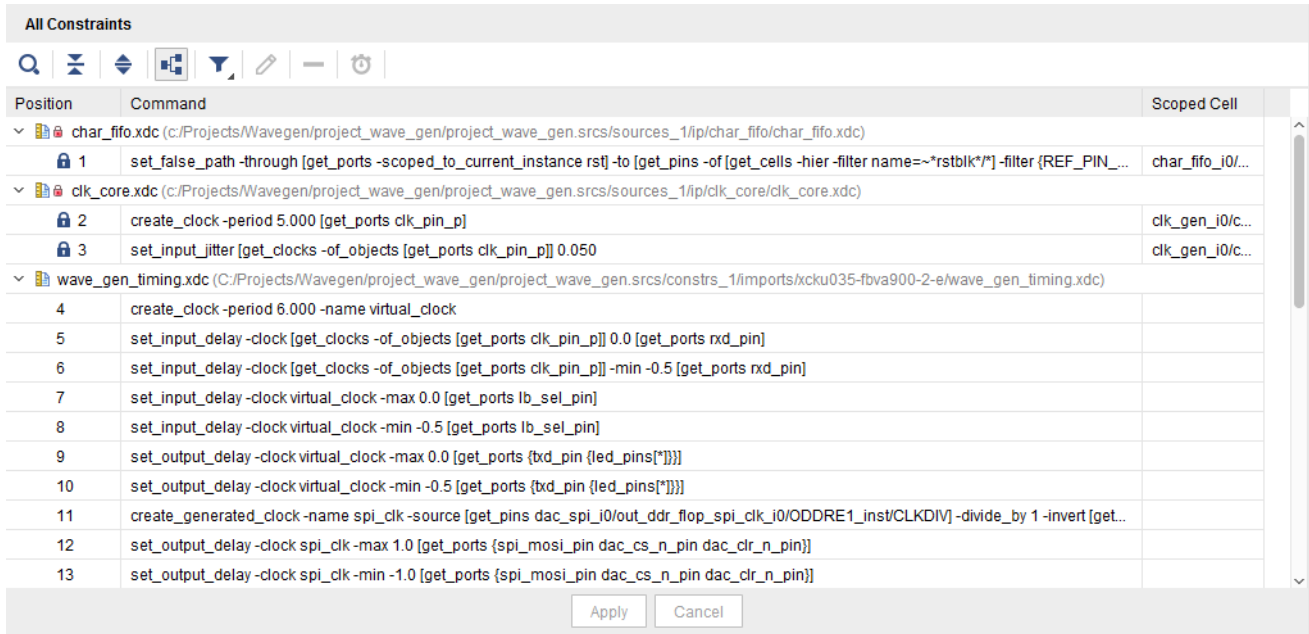


図 2-40: [Timing Constraints] ウィンドウの [All Constraints] リスト (例 1)


各ソースファイルの左側の矢印をクリックするとその制約を表示または非表示にでき、またツールバーの [Expand All] または [Collapse All] ボタンをクリックするとすべてのファイルを展開または非展開にできます。



図 2-41: [Timing Constraints] ウィンドウの [All Constraints] リスト (例 2)



ヒント: [Collapse All] をクリックして制約を非表示にすると、メモリに読み込まれている制約ファイルと、そのファイルの適用範囲をすばやく確認できます。report_compile_order -constraints コマンドを使用しても、同じ情報を表示できます。

[Group by source]  をオフにすると制約が表で表示され、ソース制約ファイルと適用されるセルの情報が [Source File] および [Scoped Cell] 列に表示されます。

All Constraints				
Position	Command	Source File	Scoped Cell	Current Instan...
1	set_false_path -through [get_ports -scoped_to_current_instance rs] -to [get_pins -of [get_cells -hier -filter name=~*rstblk?]*] -filter [REF_PIN_NAME == PRE]]	char_fifo.xdc	char_fifo_i0/U0	char_fifo_i0/U0
2	create_clock -period 5.000 [get_ports clk_pin_p]	clk_core.xdc	clk_gen_i0/clk_core_i0/Inst	
3	set_input_jitter [get_clocks -of_objects [get_ports clk_pin_p]] 0.050	clk_core.xdc	clk_gen_i0/clk_core_i0/Inst	
4	create_clock -period 6.000 -name virtual_clock	wave_gen_timing.xdc		
5	set_input_delay -clock [get_clocks -of_objects [get_ports clk_pin_p]] 0.0 [get_ports rxd_pin]	wave_gen_timing.xdc		
6	set_input_delay -clock [get_clocks -of_objects [get_ports clk_pin_p]] -min -0.5 [get_ports rxd_pin]	wave_gen_timing.xdc		
7	set_input_delay -clock virtual_clock -max 0.0 [get_ports lb_sel_pin]	wave_gen_timing.xdc		
8	set_input_delay -clock virtual_clock -min -0.5 [get_ports lb_sel_pin]	wave_gen_timing.xdc		
9	set_output_delay -clock virtual_clock -max 0.0 [get_ports txd_pin [led_pins[*]]]	wave_gen_timing.xdc		
10	set_output_delay -clock virtual_clock -min -0.5 [get_ports txd_pin [led_pins[*]]]	wave_gen_timing.xdc		
11	create_generated_clock -name spi_clk -source [get_pins dac_spi_i0/out_dds_flop_spi_clk_i0/ODDRE1_inst/CLKDIV] -divide_by 1 -invert [get_ports spi_clk_pin]	wave_gen_timing.xdc		
12	set_output_delay -clock spi_clk -max 1.0 [get_ports [spi_mosi_pin dac_cs_n_pin dac_clr_n_pin]]	wave_gen_timing.xdc		
13	set_output_delay -clock spi_clk -min -1.0 [get_ports [spi_mosi_pin dac_cs_n_pin dac_clr_n_pin]]	wave_gen_timing.xdc		
14	set_multicycle_path -from [get_cells [cmd_parse_i0/send_resp_data_reg[*]]] -include_replicated_objects -to [get_cells [resp_gen_i0/to_bcd_i0/bcd_out_reg[*]]] 2	wave_gen_timing.xdc		
15	set_multicycle_path -hold -from [get_cells [cmd_parse_i0/send_resp_data_reg[*]]] -include_replicated_objects -to [get_cells [resp_gen_i0/to_bcd_i0/bcd_out_reg[*]]] 1	wave_gen_timing.xdc		
16	set_multicycle_path -from [get_cells uart_rx_i0/uart_rx_clk_i0/*] -filter IS_SEQUENTIAL -to [get_cells uart_rx_i0/uart_rx_clk_i0/*] -filter IS_SEQUENTIAL 108	wave_gen_timing.xdc		

図 2-42: [Timing Constraints] ウィンドウの [All Constraints] リスト (例 3)

- 制約を削除するには、その制約を選択して [X] をクリックします。
- 読み取り専用でない制約を編集するには、制約の表を使用します。編集が完了したら、[Apply] をクリックして制約の変更をメモリに適用します。
- 新しい制約を追加するには、前述のダイアログボックスを使用するか、Tcl コンソールに制約を入力します。
<unsaved_constraints> という名前のグループのリストの最後に新しい制約が表示されます。

All Constraints				
Position	Command	Source File	Scoped Cell	
>	wave_gen_timing.xdc (C:/Projects/Wavegen/project_wave_gen/project_wave_gen.srcs/constrs_1/imports/xcku035-fbva900-2-el/wave_gen_timing.xdc)			
>	char_fifo_clocks.xdc (C:/Projects/Wavegen/project_wave_gen/project_wave_gen.srcs/sources_1/ip/char_fifo/char_fifo_clocks.xdc)			
32	set_max_delay -datapath_only -from [get_cells [inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gc.clkx*rd_pntr_gc_reg[*]]] -to [get_cells [inst...		char_fifo_i0/...	
33	set_bus_skew -from [get_cells [inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gc.clkx*rd_pntr_gc_reg[*]]] -to [get_cells [inst_fifo_gen/gconv...		char_fifo_i0/...	
34	set_max_delay -datapath_only -from [get_cells [inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gc.clkx*wr_pntr_gc_reg[*]]] -to [get_cells [ins...		char_fifo_i0/...	
35	set_bus_skew -from [get_cells [inst_fifo_gen/gconvfifo.rf/grf.rf/gntv_or_sync_fifo.gc.clkx*wr_pntr_gc_reg[*]]] -to [get_cells [inst_fifo_gen/gcon...		char_fifo_i0/...	
>	<unsaved_constraints>			
36	set_false_path -from [get_clocks clk_rx_clk_core] -to [get_clocks clk_rx_clk_core]			

図 2-43: [Timing Constraints] ウィンドウの [All Constraints] リスト (例 4)

制約を保存すると、新しい制約はすべて、「target」とマークされている XDC ファイルの最後に保存されます。メモリのデザインに関連付けられている制約セットにターゲット XDC ファイルがない場合、または制約セットに Tcl スクリプトしかない場合は、制約の保存先を指定するようメッセージが表示されます。

制約は定期的に保存してください。[Save] をクリックするか、[File] → [Constraints] → [Save] をクリックします。



重要: 新規または変更された制約を Tcl スクリプトに保存することはできません。



注意: [Timing Constraints] ウィンドウの制約がまだ適用されていない場合は、新しい制約を Tcl コンソールに入力しないでください。[Timing Constraints] ウィンドウでの制約の順序と、メモリでの制約の順序が異なるものになる可能性があります。混乱を避けるため、既存の制約を変更したら、必ずすべての制約を適用し直すようにしてください。

XDC テンプレート

XDC テンプレートにアクセスするには、[Tools] → [Language Templates] をクリックします。

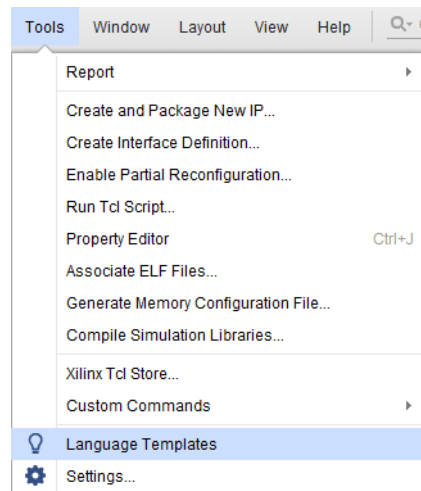


図 2-44: XDC テンプレート

XDC テンプレートの内容

XDC テンプレートには、次が含まれます。

- クロック定義、ジッター、入力/出力遅延、例外などの一般的なタイミング制約
- 物理制約
- コンフィギュレーション制約

XDC テンプレートの使用

XDC テンプレートを使用するには、次の手順に従います。

1. 使用するテンプレートを選択します。
2. [Preview] セクションに表示されるテキストをコピーします。
3. XDC ファイルにコピーしたテキストを貼り付けます。
4. 変数 (<> で囲まれた文字列) をデザインでの実際の名前または値に置き換えます。

アドバンス XDC テンプレート

システム同期およびソース同期 I/O 遅延制約などのアドバンス制約の場合、デザイン要件を表現するため Tcl 変数を設定する必要があるものがあります。Tcl 変数は、実際の `set_input_delay` および `set_output_delay` 制約で使用されます。

デフォルト値ではなく必要な値が代入されていることを確認してください。

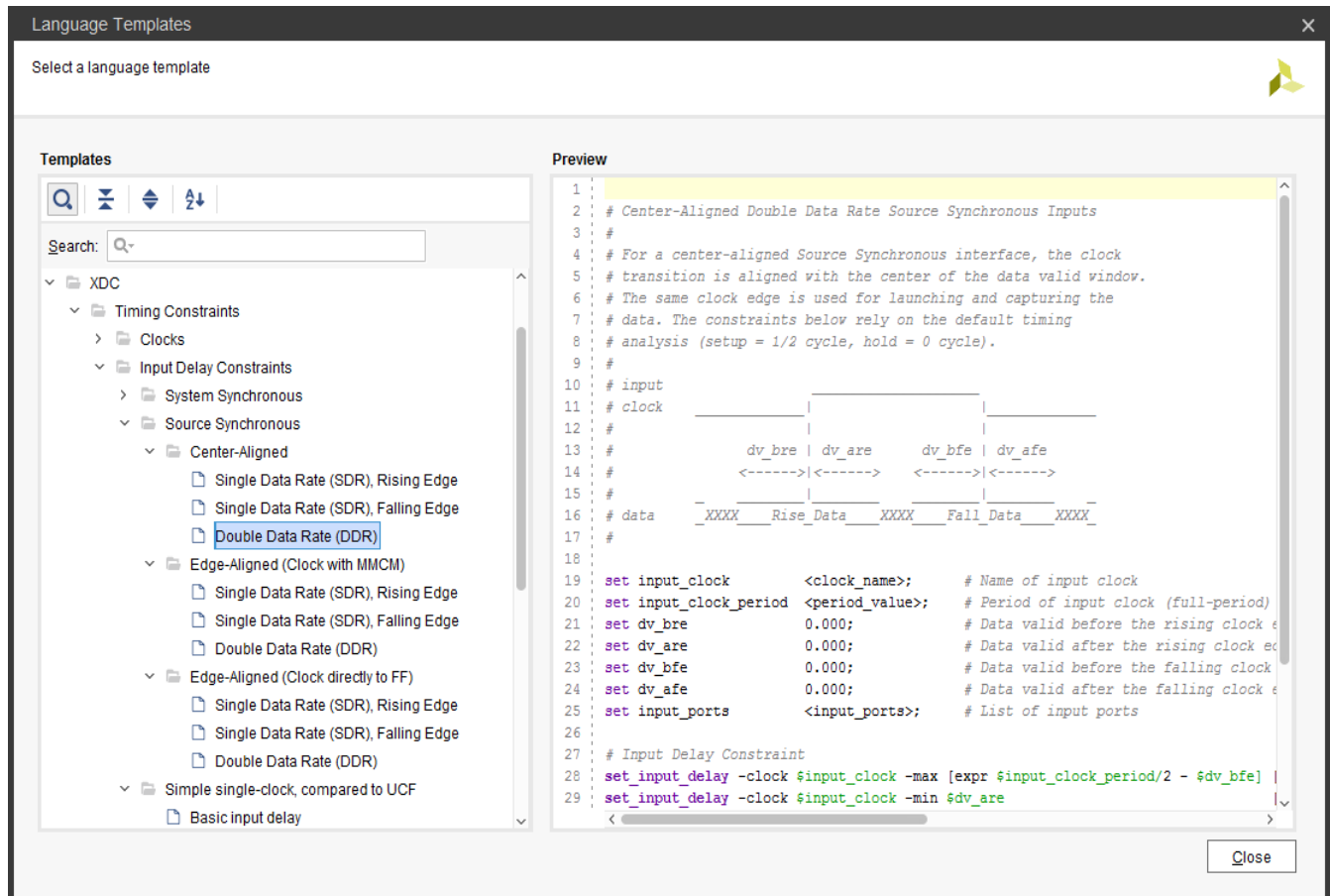


図 2-45: I/O 遅延制約のテンプレート

合成制約の作成

Vivado 合成では、デザインの RTL 記述が、テクノロジーにマップされたネットリストに変換されます。このプロセスは複数の段階で実行され、多数のタイミングドリブン最適化が実行されます。

ザイリンクス FPGA には、用途が多岐にわたるロジック機能が多数含まれています。インプリメンテーションの最後ですべてのデザイン要件が満たされるようにするため、制約を使用して合成エンジンに指示を与える必要があります。

Vivado IDE 合成の制約には、次の 4 種類があります。

- 「RTL 属性」
- 「タイミング制約」
- 「エラボレート済みデザイン制約」

RTL 属性

RTL 属性は RTL ファイルに記述する必要があります。RTL 属性では、通常、ロジックの特定の部分のマップ スタイル、レジスタやネットなどの保持、最終的なネットリストのデザイン階層の制御などを選択します。

詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) [\[参照 8\]](#) のこのセクションを参照してください。

重要: DONT_TOUCH 属性は USED_IN_SYNTHESIS および USED_IN_IMPLEMENTATION のプロパティには従いません。合成 XDC で DONT_TOUCH プロパティを使用すると、USED_IN_IMPLEMENTATION の値にかかわらず、インプリメンテーションに伝搬されます。

USED_IN_SYNTHESIS および USED_IN_IMPLEMENTATION の詳細は、[9 ページの「合成およびインプリメンテーション制約ファイル」](#)を参照してください。

DONT_TOUCH 属性の例:

```
set_property DONT_TOUCH true [get_cells fsm_reg]
```

タイミング制約

タイミング制約は、XDC ファイルを使用して合成エンジンに渡す必要があります。セットアップ解析に関する次の制約のみが、合成結果に実際に影響します。

- `create_clock`
- `create_generated_clock`
- `set_input_delay`
- `set_output_delay`
- `set_clock_groups`
- `set_false_path`
- `set_max_delay`
- `set_multicycle_path`

物理制約およびコンフィギュレーション制約

物理制約およびコンフィギュレーション制約は、合成アルゴリズムでは無視されます。

エラボレート済みデザイン制約

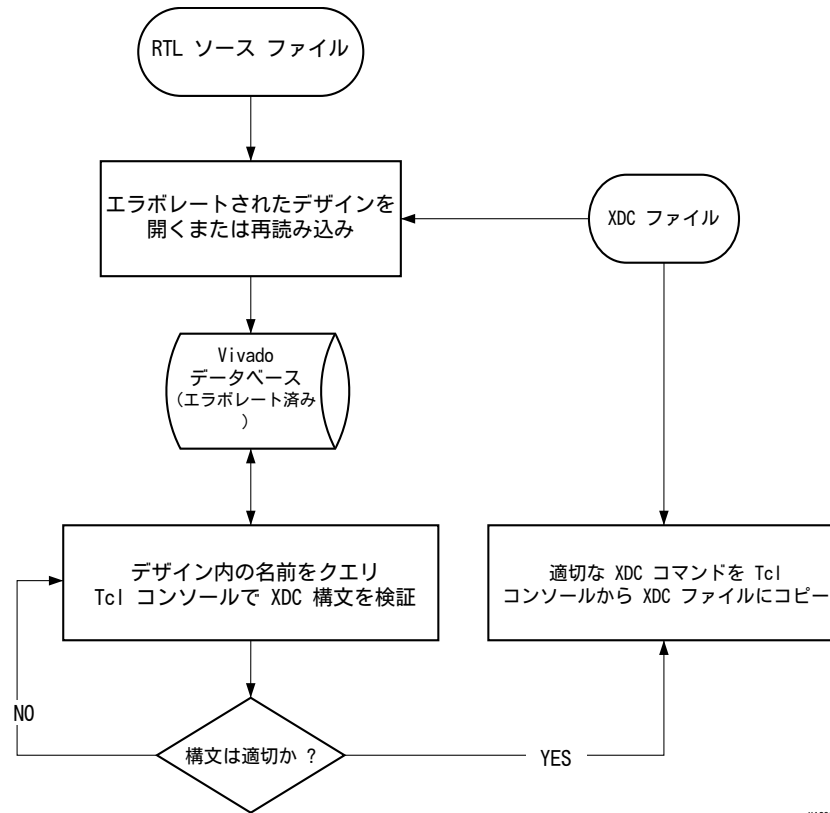


推奨: 合成 XDC の最初のバージョンを作成するときは、デザイン要件を記述する単純なタイミング制約を使用してください。

フローのこの段階では、ネット遅延のモデリングは正確ではありません。この時点での主な目的は、インプリメンテーションを開始する前に、タイミングを満たすか、タイミングが少しの差で満たされていない合成済みネットリストを得ることです。この状態のネットリストを得るには、多くの場合、XDC および RTL を何回か修正する必要があります。

[図 2-46](#) に、RTL ベースの XDC の作成手順を示します。エラボレートされたデザインで、合成用に制約するデザインのオブジェクト名を見つけます。

XDC ファイルを保存する前に、[Tcl Console] ウィンドウで XDC コマンドの構文を確認してください。エラボレート済みデザインでは、制約を作成し、クロックやデザイン オブジェクトをクエリできますが、タイミング レポート コマンドは実行できません。



X12982

図 2-46: エラボレートされたデザインでの制約の作成

合成用の制約を記述するときに安全に使用できるデザイン オブジェクトは、次のとおりです。

- 最上位ポート
- 手動でインスタンス化されたプリミティブ (セルおよびピン)

一部の RTL 名は、エラボレートされたデザインを作成するときに、変更されたり失われたりします。これがよく発生するのは、次の名前です。

- 「1 ビット レジスタの名前」
- 「複数ビット レジスタの名前」
- 「吸収されるレジスタおよびネット」
- 「階層名」

1 ビット レジスタの名前

デフォルトのレジスタ名は、RTL の信号名に「_reg」という接尾辞が付いた名前になります。

たとえば、VHDL および Verilog で次のように定義されている信号の場合、エラボレーション中に生成されるインスタンス名は wbDataForInputReg_reg になります。

```
VHDL: signal wbDataForInputReg : std_logic;
Verilog: reg wbDataForInputReg;
```


図 2-47 に、レジスタとそのピンの回路図を示します。レジスタ インスタンスまたはそのピンに制約を定義できます。

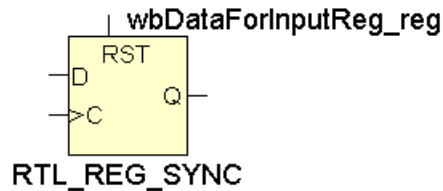


図 2-47: エラボレートされたデザインでの 1 ビット レジスタ

複数ビット レジスタの名前

デフォルトのレジスタ名は、RTL の信号名に「_reg」という接尾辞が付いた名前になります。XDC コマンドでは、複数ビット レジスタの個々のビットのみをクエリし、制約できます。

たとえば、VHDL および Verilog で次のように定義されている信号の場合、エラボレーション中に生成されるインスタンス名は loadState_reg[0]、loadState_reg[1]、および loadState_reg[2] になります。

```
VHDL: signal loadState: std_logic_vector(2 downto 0);
Verilog: reg [2:0] loadState;
```

図 2-48 に、レジスタの回路図を示します。複数ビット レジスタは、1 ビット レジスタのベクターとして表示されます。回路図では、ベクターは可能な限り小さく表示されます。各ビットを個別に表示することも可能です。

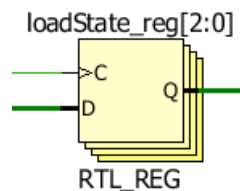


図 2-48: エラボレートされたデザインでの複数ビット レジスタ

各レジスタを個別に制約するか、次のパターンを使用してグループとして制約できます。

- レジスタ ビット 0 のみ
loadState_reg[0]
- すべてのレジスタ ビット
loadState_reg[*]



重要: loadState_reg[2:0] のようなパターンを使用して、マルチビット レジスタ (より広い意味ではマルチビット インスタンスすべて) をクエリすることはできません。

上記の名前は合成後のネットリストでの名前とも一致するので、これらに対して設定された制約は通常インプリメンテーションでも機能します。

吸収されるレジスタおよびネット

RTL ソースにあるレジスタやネットの一部が、さまざまな理由によりエラボレート済みデザイン (または合成済みデザイン) からなくなってしまうことがあります。たとえば、メモリ ブロック、DSP、シフトレジスタを推論する場合、複数のデザイン オブジェクトを 1 つのリソースに配置することが必要となります。制約を定義するのにこれらのオブジェクトを使用するのではなく、ほかの接続されているレジスタやネットで使用できるものがあるかを探してみてください。

階層名

Vivado 合成でデザインの階層を完全に保持するよう指定しない場合、合成中に一部またはすべての階層をフラットにできます。`-flatten_hierarchy`の詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) [\[参照 8\]](#)のこのセクションを参照してください。



推奨: すべての階層レベルが暗示的な一致を示す * 文字を使用せずにスラッシュ (/) 文字を使用して明示的に記述されている場合、合成制約で完全に展開された階層名を使用してください。階層が変更された場合でも、最終的なネットリスト名と一致する可能性が高くなります。

たとえば、デザインのサブレベルに次のようなレジスタがあるとします。

エラボレート済みデザインの例:

```
inst_A/inst_B/control_reg
```

合成中、このレジスタに特別な最適化は実行されないとすると、ツール オプションまたはデザイン構造によって、フラット名または階層名が得られます。

フラット ネットリストでのインスタンス名

```
inst_A/inst_B/control_reg      (F)
```

階層ネットリストでのインスタンス名

```
inst_A/inst_B/control_reg      (H)
```

フラット化された階層レベルを示すのにスラッシュ (/) が使用されるので、見た目には違いますが、メモリ内のオブジェクトをクエリすると違いがはっきりします。次のコマンドでは、F のネットリスト オブジェクトが返され、H のネットリスト オブジェクトは返されません。

```
% get_cells -hierarchical *inst_B/control_reg
% get_cells inst_A*control_reg
```

階層名に関連する問題を回避するため、次のようにすることをお勧めします。

- `get_*` コマンドを `-hierarchical` オプションを指定せずに使用します。
- エラボレートされたデザインに表示されるように、スラッシュ (/) を使用してすべての階層を区切ります。

階層オプションなしの例:

- 次のコマンドは、フラット ネットリストと階層ネットリストのどちらでも機能します。

```
% get_cells inst_A/inst_B/*_reg  
% get_cells inst_*/inst_B/control_reg
```

- 次のようにすることもできます。

```
% get_cells -hier -filter {NAME =~ inst_A/inst_B/*_reg}  
% get_cells -hier -filter {NAME =~ inst_*/inst_B/control_reg}
```



注意: 階層セルでも同様に、合成を実行するときに階層ピンに制約を設定しないでください。また、組み合わせロジックの演算子を接続するネットに制約を設定しないでください。これらは LUT に結合され、ネットリストからなくなる可能性があります。



推奨: 制約を変更したら XDC ファイルを保存し、エラボレートされたデザインを読み込み直して、メモリ内の制約と XDC ファイルの制約が一致するようにしてください。合成後に、メモリ内の同じ合成 XDC を使用して合成済みデザインを読み込み、[Report Timing Summary] を使用してタイミング解析を実行します。

合成によりデザインが変換されているため、合成前の制約の一部は正しく適用されない可能性があります。これらの問題を解決するには、次の手順に従います。

1. 合成済みネットリストに適用する新しい XDC 構文を検索します。
2. その制約を、インプリメンテーションのみに使用する新しい XDC ファイルに保存します。
3. 合成のみに使用する合成制約を別の XDC ファイルに移動します。

インプリメンテーション制約の作成

合成済みネットリストが生成されたら、インプリメンテーションに適用する XDC ファイルまたは Tcl スクリプトと共にメモリに読み込みます。XDC を読み込むとメッセージが表示されるので、そのメッセージに従って制約を確認し、適用できない制約を修正してください。

合成済みネットリストのオブジェクト名が、エラボレート済みデザインの名前と異なる場合があります。その場合は、正しい名前で作約を作成し直し、インプリメンテーションのみの XDC ファイルに保存します。

すべての XDC ファイルが正しく読み込まれたら、次の目的でタイミング解析を実行できます。

- 入力遅延および出力遅延などの不足している制約を追加。
- フォルス パス、マルチサイクル パス、最小/最大遅延制約などのタイミングの例外を追加。
- パスが長いことが原因で大きな違反が発生しているものを特定し、RTL 記述を修正。

合成に使用したのと同じ基本制約を使用し、インプリメンテーション専用の新しい制約を保存する 2 つ目の XDC ファイルを作成できます。物理制約およびコンフィギュレーション制約を別の XDC ファイルに保存することも可能です。

注記: プロジェクト モードでは、合成済みデザインを開くと、合成後の DCP からのネットリストがリンクされ、最上位階層ネットリストが構築されます。インプリメンテーション用にマークされた XDC ファイルもすべて自動的に読み込まれます。これにより、完全に合成済みのデザインでインプリメンテーション制約を確認できます。インプリメンテーション制約を変更すると、開いている合成済みデザインは最新の状態ではなくなりますが、合成済み run は最新の状態のままです。GUI のバナーにデザインを読み込み直すオプションが表示されます。

図 2-49 に、ネットリスト ベースの XDC の作成手順を示します。

合成ロジック複製のための制約の調整

合成中、デザイン パフォーマンスを向上するため一部のレジスタが複製されます。合成エンジンでは、複製されたセルを含めるためにユーザー XDC 制約は変更されません。Vivado 合成で複製されたオブジェクトにタイミング制約が設定されている場合、XDC 制約の記述方法によって、複製されたセルに常に XDC 制約が適用されるわけではなく、インプリメンテーションの結果の質 (QoR) に影響することがあります。

Vivado 合成を使用する際は、`get_cells` および `get_pins` コマンドを使用すると、複製されたオブジェクトが自動的に含まれるようになります。

たとえば `set_false_path -from [get_cells -hierarchical *rx_reg]` は、次のように記述し直すと、インプリメンテーション中に複製されたオブジェクトが含まれるようになります。

```
set_false_path -from [get_cells -hierarchical *rx_reg -include_replicated_objects]
```

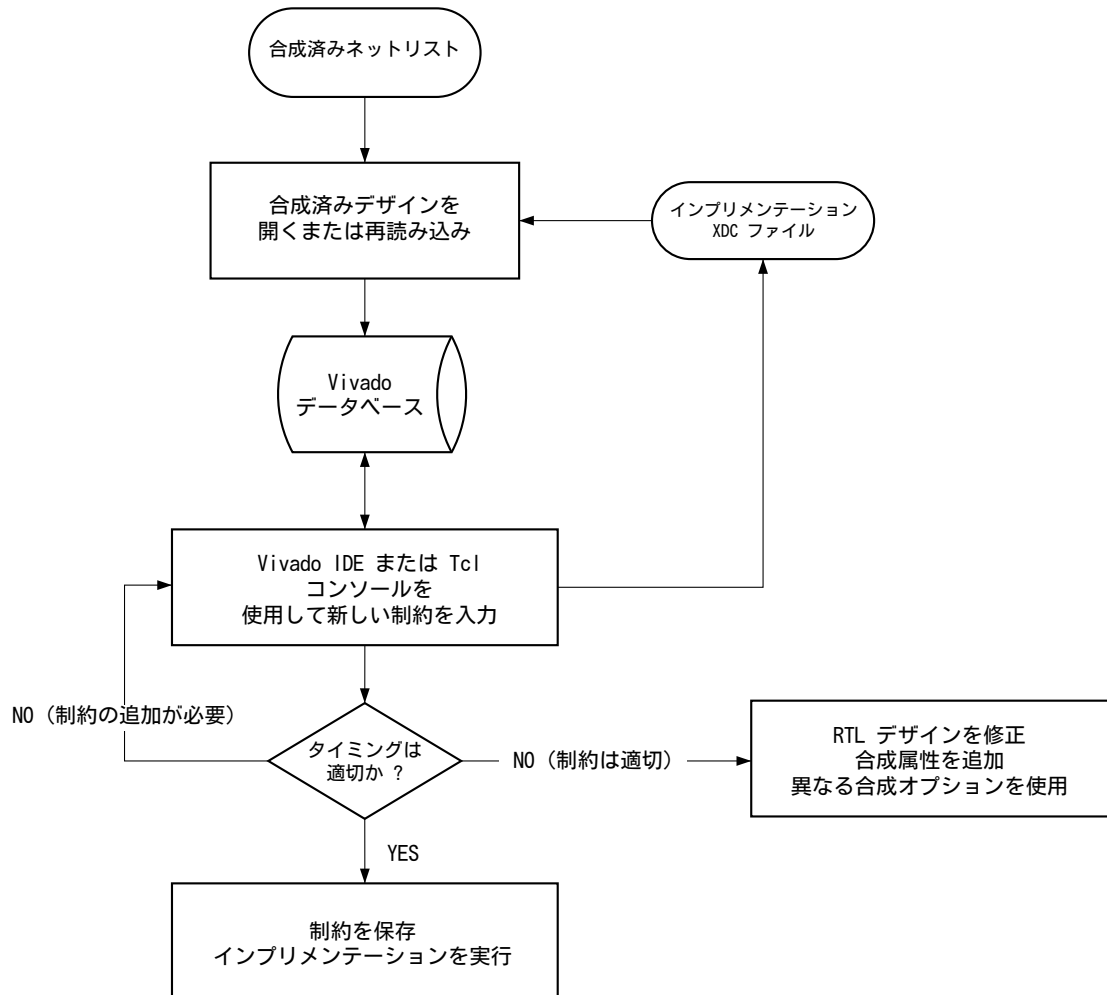
`-include_replicated_objects` コマンド オプションは、複製されたオブジェクトに設定された `ORIG_CELL_NAME` プロパティに依存します。次のクエリ コマンドでは、元のセルと複製されたセルが返されます。

```
get_cells -include_replicated_objects *rx_reg
get_cells -include_replicated_objects [get_cells -hier -filter {NAME =~ *rx_reg}]
get_cells -hierarchical -filter {NAME =~ *rx_reg || ORIG_CELL_NAME =~ *rx_reg}
```

`-filter` オプションは、常にオブジェクトのコレクションが構築されてから適用されます。フィルター式が `NAME` プロパティを参照する場合、`-filter` と `-include_replicated_objects` を一緒に使用しないようにしてください。このような場合、`NAME` で指定したパターンと一致しないと、複製オブジェクトは返されません。たとえば、次の構文では、`*reg_replica*` に一致する複製オブジェクトは返されません。

```
get_cells -include_replicated_objects -filter {NAME =~ *rx_reg}
```

設計手法チェック (`report_methodology`) を実行して XDCV-1 および XDCV-2 チェックのメッセージを確認し、`get_cells/get_pins -include_replicated_objects` オプションを使用して記述し直す必要がある制約を特定することをお勧めします。



X12981

図 2-49: 合成済みデザインでの制約の作成

インプリメンテーションに進む前に、大きなタイミング違反がないことを確認する必要があります。マイナーなタイミング違反であれば、たいていのものは配置配線で修正できますが、タイミング クロージャを不可能にするような根本的なデザインの問題は修正できません。



推奨: RTL を再確認して、違反のあるパスのロジック段数を削減したり、クロック ツリーを簡潔にして専用クロック リソースが使用されるようにして、関連するクロック間のスキューを最小限に抑えるようにしてください。合成属性を追加したり、別の合成オプションを使用することも可能です。

詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) [\[参照 8\]](#) のこのセクションまたは『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) [\[参照 9\]](#) のこのセクションを参照してください。

ブラックボックスを使用した合成用制約の調整

アウトオブコンテキスト (OOC) 合成モードを使用する場合、OOC モジュール (IP/BD/DFx/...) は最上位内のブラックボックスとして推論されます。つまり、OOC モジュール内のネットリストオブジェクトには最上位制約からはアクセスできません。これには、合成用の最上位制約がインプリメンテーション用の制約と異なっている必要があることがあります。プロジェクトモードでは、これは合成用の特定の XDC ファイルを作成し、`USED_IN_SYNTHESIS=TRUE` および `USED_IN_IMPLEMENTATION=FALSE` プロパティを設定すると実行できます。インプリメンテーション用の最上位 XDC は、`USE_IN_SYNTHESIS = FALSE` である必要があります。

ブラックボックスからアクセス可能なオブジェクトは、入力ポートと出力ポートのみです。このため、ブラックボックスを参照するときに最上位が指定できるタイミング制約のタイプが限られます。

OOC 合成による最上位制約の制限には、次のようなものがあります。

- OOC モジュール内で生成された自動派生クロックの名前は変更できません。
- OOC モジュール内で定義されたクロック名は参照できません。OOC モジュールの出力に伝搬されるクロックは、モジュールの XDC 内でクロックの名前が変更された場合でも、モジュール内の名前からではなく、モジュールのポートに接続されているネットに基づいて命名されます。
- 最上位制約が OOC モジュールから出力されるクロックを参照する必要がある場合は、`'get_clocks -of_objects [get_pins <MODULE_OOC_OUTPUT_CLOCK_PORT>']` などのクエリを使用する必要があります。

制約の適用範囲の設定

特定の XDC ファイルの制約は、必要に応じて、デザイン特定のモジュール、特定のセル、またはその両方に適用できます。これは、最上位の情報なしで、制約をサブレベル用に作成して適用するのに便利なだけでなく、ブロックレベル制約は、最上位制約とは別に作成する必要があります。さまざまなコンテキストで使用できるように、できるだけ一般的に記述する必要があります。また、これらはブロックの境界を超えるロジックには影響しないようにする必要があります。Vivado Design Suite プロジェクト内で生成された Vivado IP カタログの IP コアの場合は、メモリの制約はデフォルトでこの方法で読み込まれます。

XDC ファイルの適用範囲を設定するプロパティ

制約の適用範囲を限定するには、XDC ファイルで次のプロパティを指定します。

- `SCOPED_TO_REF`: モジュール (またはエンティティ) の名前を指定します。制約は、指定したモジュール (またはエンティティ) のすべてのインスタンスに適用されます。
- `SCOPED_TO_CELLS`: 階層セルの名前を指定します。制約は、各階層セルに個別に適用されます。
- `SCOPED_TO_REF + SCOPED_TO_CELLS`: 両方のプロパティが指定されている場合、`SCOPED_TO_REF` で指定されるモジュール (またはエンティティ) 内にある `SCOPED_TO_CELLS` で指定されている各セルに制約が適用されます。

これらのプロパティは、IP カタログを使用して RTL プロジェクトに追加されている IP コアに対しては自動的に設定されます。

XDC ファイルの適用範囲プロパティの設定例

図 2-50 に、2つの階層セル `uart_tx_ctl_i0` および `uart_baud_gen_tx_i0` を含む、`uart_tx` モジュールのインスタンスである `uart_tx_i0` セルを示します。

プロジェクトには、`uart_tx_ctl` モジュールを制約する XDC ファイル (`uart_tx_ctl.xdc`) が含まれます。

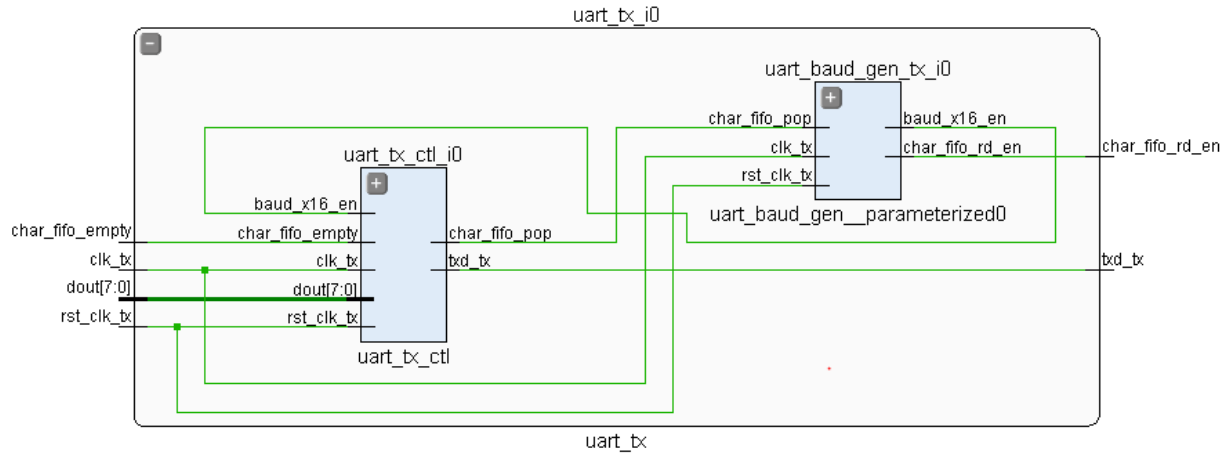


図 2-50: XDC ファイルの適用範囲プロパティの設定例

`uart_tx_ctl.xdc` に適用範囲プロパティを設定する同等の Tcl コマンドには、次の 3 つがあります。これらの値は、Vivado IDE で XDC ファイルの [Properties] ウィンドウでも設定できます。

```
# Using the reference module name only:
set_property SCOPED_TO_REF uart_tx_ctl [get_files uart_tx_ctl.xdc]

# Using the cell name only:
set_property SCOPED_TO_CELLS uart_tx_i0/uart_tx_ctl_i0 [get_files uart_tx_ctl.xdc]

# Using both the uart_tx reference module and uart_tx_ctl_i0 instance:
set_property SCOPED_TO_REF uart_tx [get_files uart_tx_ctl.xdc]
set_property SCOPED_TO_CELLS uart_tx_ctl_i0 [get_files uart_tx_ctl.xdc]
```

Vivado Design Suite を非プロジェクト モードで使用している場合、`read_xdc` コマンドに `-ref` および `-cells` オプションを使用すると、同じ結果を得ることができます。

```
# Using the reference module name only:
read_xdc -ref uart_tx_ctl uart_tx_ctl.xdc
# Using the cell name only:
read_xdc -cells uart_tx_i0/uart_tx_ctl_i0 uart_tx_ctl.xdc
# Using both the uart_tx reference module and uart_tx_ctl_i0 instance
read_xdc -ref uart_tx -cells uart_tx_ctl_i0 uart_tx_ctl.xdc
```

1つのモジュールがデザインに複数回インスタンス化されている場合、合成中にこれらが個別化され、合成後はRTLモジュールの各インスタンスのモジュール名は異なるものになります。元のRTLモジュールのすべてのインスタンスにXDC制約を適用するには、REF_NAMEプロパティではなくORIG_REF_NAMEを使用する必要があります。次に例を示します。

```
set_property SCOPED_TO_REF [get_cells -hierarchical -filter {ORIG_REF_NAME ==  
uart_tx_ctl}] [get_files uart_tx_ctl.xdc]  
read_xdc -ref [get_cells -hierarchical -filter {ORIG_REF_NAME == uart_tx_ctl}]  
uart_tx_ctl.xdc
```

XDCの適用範囲限定のメカニズム

制約の適用範囲の限定には、ポートを除き、SDC (Synopsys Design Constraint) 標準の一部であるcurrent_instanceが使用されます。current_instanceコマンドでデザインの下位階層を指定すると、オブジェクトクエリコマンドではその階層レベルとその下位階層に含まれるオブジェクトのみが返されます。

ただし、タイミングクロックオブジェクトおよびネットリストポートは例外です。

- タイミングクロックは、create_clockまたはcreate_generated_clockで定義されます。これらのクロックは、current_instanceの設定にかかわらず、デザイン全体でアクセスできます。get_clocksコマンドを使用すると、現在のインスタンスに含まれないクロックまたは現在のインスタンスを越えて伝搬されるクロックをクエリできます。適用範囲が限定された制約を作成する際、クロックが現在のインスタンスに完全に含まれていない場合は、クロックにタイミング例外を定義することはお勧めしません。クロックをXDCで参照できるようにするには、クロックが既に定義されている必要があるため、プロジェクトのXDCファイルの順序を変更する必要がある場合があります。
- get_portsコマンドを使用すると、current_instanceコマンドで下位インスタンスを設定していても最上位ポートを取得できますが、read_xdc -ref/-cellsコマンドで下位インスタンスに設定されたXDCファイルを読み込む場合や、SCOPED_TO_REF/SCOPED_TO_CELLSファイルプロパティを設定した後にデザインを読み込んだ場合、get_portsコマンドの動作は異なります。
 - get_portsで使用されるポート名は、最上位ポート名ではなく、指定されているインスタンスインターフェイスのポート名です。
 - 指定されているインスタンスポートがデザインの階層を介して最上位ポートに直接接続されている場合、get_portsコマンドで最上位ポートが返され、制約は最上位ポートに適用されます。
 - 指定されているインスタンスポートと最上位ポートの間にI/Oやクロックバッファなどの最下位セルがある場合は、get_portsコマンドはget_pinsコマンドになり、指定されている階層インスタンスピンが返されます。

Vivado Design Suite IPの制約ファイルを読み込む際には、常にXDCの制約適用範囲メカニズムが使用されます。

図2-51および図2-52に、IPレベルのXDCがこの方法で読み込まれる場合にget_portsコマンドがどのように処理されるかの例を示します。

図2-51では、I/OバッファがIP内にインスタンス化され、IPインターフェイスピンがその階層に関係なく最上位ポートに直接接続されています。IPのXDCが適用されると、get_portsコマンドの引数は自動的に最上位ポートに置き換えられます。

これにより、IPレベルにLOCまたはIOSTANDARDのような物理制約を設定でき、それが必要な最上位ポートに配置されるようになります。IPレベルで、デザインの最上位ポート名を指定する必要はありません。

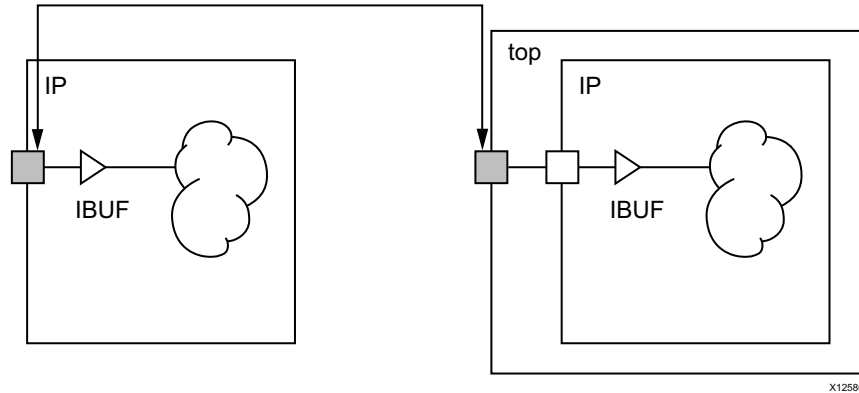


図 2-51: IP ポートの最上位ポートへの変換

図 2-52 では、IP に I/O バッファが含まれないので、合成エンジンで IP インターフェイス ピンと最上位ポート間に 1 つの I/O バッファが推論されます。その結果、XDC が適用されると、get_ports は IP インターフェイス ピン (階層ピンなど) の get_pins に変換されます。

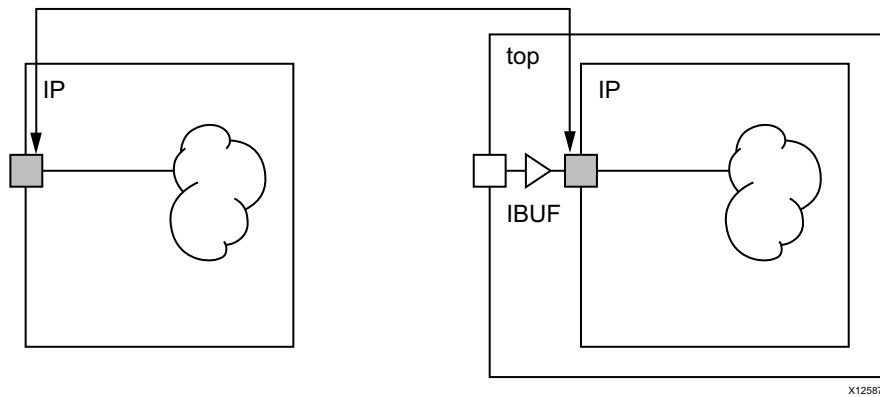


図 2-52: IP ポートの階層ピンへの変換

この機能は、IP のインターフェイスに制約を作成する際や、最上位デザインの名前を知らずにサブレベル モジュールを作成する際に有益です。

制約適用範囲が指定されている XDC ファイルに、最上位ポートにのみ適用可能な制約が含まれているのに、IP インスタンスが最上位ポートには直接接続されていない場合、Vivado Design Suite で XDC を読み込むときにエラーが発生します。たとえば、次の制約は最上位ポートのみに適用可能で、階層ピンには適用できません。

- set_input_delay/set_output_delay
- set_property IOSTANDARD

XDC を使用した IP およびサブモジュールの制約

IP を作成し、それを Vivado IP カタログから使用できるようにするために IP をパッケージする場合、XDC 制約も一緒にパッケージできます。Vivado Design Suite の IP はプラグアンドプレイであり、最上位デザイン制約を完了するのに、制約を切り貼りするための IP のサンプル プロジェクトは必要ありません。スタンドアロンの最上位デザインと同様に、その IP 用に開発された XDC ファイルと共に、IP をパッケージできます。IP カタログを使用して IP をプロジェクトにインスタンス化した場合、Vivado ツールで制約が適切に読み込まれます。

同様に、デザインのサブモジュール用に制約を作成し、プロジェクト フローでは

SCOPED_TO_REF/SCOPED_TO_CELLS XDC ファイル プロパティを設定、非プロジェクト モードでは `read_xdc -ref/-cells` コマンドを使用すると、IP コアと同じ制約適用範囲設定メカニズムを使用できます。

範囲を特定したクエリのガイドライン

このフローをスムーズに機能させるには、制約が IP またはサブモジュール インスタンスのみに適用されるように XDC 制約を記述する必要があります。Vivado ツールでは、62 ページの「制約の適用範囲の設定」で説明したように、クエリの範囲を特定の階層レベルに設定できます。IP またはサブモジュール用に制約を作成する際は、クエリコマンドの動作を理解しておく必要があります。

- セル/ネット/ピン オブジェクトのクエリは、指定されたインスタンスおよびそのサブモジュールに限定されます。
 - `get_cells/get_nets/get_pins <name pattern>`
 - オブジェクトの NAME プロパティは、適用範囲に指定されたインスタンスだけでなく、最上位に対するオブジェクトの完全階層パスを示します。NAME プロパティに対して `get_*` コマンドの `-filter` オプションを使用する場合は、glob 文字列一致演算子を使用し、* で開始するパターンを指定します。次に例を示します。

```
get_nets -hierarchical -filter {NAME =~ *clk}
```
- `get_ports` では、ブロック/IP のポートが直接最上位ポートに接続されている場合は、最上位ポートが戻されます。それ以外の場合、`get_ports` では階層ピンが戻されます。
- ネットリストのヘルパー コマンドも範囲が限定されます。
 - `all_ffs`、`all_latches`、`all_rams`、`all_registers`、`all_dsps`、`all_hsios` は、現在のインスタンスに含まれるインスタンスのみを返します。
- I/O ヘルパー コマンドは、範囲が限定された XDC では使用できません。
 - `all_inputs`、`all_outputs`
- クロック コマンドでは範囲は限定されず、デザインのすべてのタイミング クロックが返されます。
 - `get_clocks`、`all_clocks`
- 最上位およびローカル クロック オブジェクトは、ネットリストに対して `get_clocks -of_objects` を使用してクエリできます。
 - 現在のインスタンスに供給されるクロックは、`get_clocks -of_objects [get_ports <interfacePinName>]` を使用すると取得できます。
 - 現在のインスタンス内で自動生成されたクロックは、`get_clocks -of_objects [get_pins <instName/outPin>]` (`instName` はクロック ジェネレーター インスタンス) を使用すると取得できます。

- デザインにあるオブジェクトをクエリするには、`-of_objects` オプションを使用します。
 - 例: `get_pins -leaf -of_objects [get_nets local_net]`
- 現在のインスタンス インターフェイスのネットに接続されている最上位ポートをクエリできます。
 - `get_ports -of_objects [get_nets <scoped_instance_net>]`
- IP/サブモジュール インターフェイス ピンはクエリできません。
 - `get_pins clk` を実行すると、エラーが返されます。
- パストレース コマンドも範囲が限定されます。
 - `all_fanin/all_fanout` は指定された範囲でのみ実行され、その境界を越えては実行されません。
- 特定クロックに接続されているセルをすべてクエリするには、`all_registers` コマンドに `-clock` オプションを指定するのではなく、`get_cells/get_pins/get_nets` コマンドでできる限り具体的なパターンを指定します。制約する必要があるのが数個のオブジェクトであるのに、多数のオブジェクトを含むリストが返されると、実行時間が長くなる可能性があります。

範囲を限定したタイミング制約のガイドライン

最上位デザインに悪影響を与えないようにするには、IP またはサブモジュールに対して記述したタイミング制約が、一部のクロック定義を除き、その IP の境界を越えて伝搬されないようにする必要があります。

たとえば、IP の XDC で、IP に供給される 2 つのクロックの間にフォルス パス制約が定義されているとします。IP には非同期クロック境界に適切な回路が含まれていても、デザインの残りの部分には含まれていない場合があります。2 つのクロックに関連性があり、ハードウェアを正しく機能させるために、デザインの残りの部分でこれらのクロックのタイミングを解析する必要がある場合、これは問題となります。

また、第 7 章「XDC の優先順位」で説明しているように、IP の XDC ファイルで定義されたタイミング例外が最上位制約より優先されてしまう場合があります。このような状況を回避するため、IP のローカル ネットリスト オブジェクトに制約を適用することをお勧めします。2 つのグローバル クロック間にフォルス パスを設定する場合は、IP 内の始点セルのグループから、同じ IP 内の終点セルのグループに設定します。この方法は、グローバル例外に対し、ポイント ツー ポイント例外と呼ばれます。

IP/サブモジュール XDC に推奨される制約規則

ブロックレベルの制約は、次のような規則に従う必要があります。

1. クロックがデザインの最上位で作成されるようにする場合は、ブロックレベルの制約内でクロックを定義しないようにします。

ブロック内では、`get_clocks -of_objects` コマンドを使用してクエリします。このコマンドは、デザインの特定期間オブジェクトを介するクロックすべてを返します。

例:

```
set blockClock [get_clocks -of_objects [get_ports clkIn]]
```

クロックをブロック内で定義する必要がある場合は、インスタンスエート 済み入力/入出力バッファを駆動する入力/入出力ポートか、クロックを作成/変換するセルの出力 (MMCM/PLL またはタイミング ツールで自動的に処理される特別バッファは除く) に対して定義する必要があります。

例:

- 。 入力バッファの付いた入力クロック
 - 。 クロック分周器
 - 。 GT 復元クロック
2. ポートが最上位ポートに直接接続され、I/O バッファが IP 内にインスタンス化されている場合にのみ、入力および出力遅延を指定します。

例:

- 。 入力バッファの付いた入力データ ポート
 - 。 出力バッファの付いた出力データ ポート
3. IP 内に制限されていない 2 つのクロックにタイミング例外を定義しないようにします。
4. クロック名は最上位のクロック名に基づいて変わるので、クロックは名前指定しないようにしてください。ブロックが複数回インスタンス化される場合も名前指定しないでください。
5. ブロックが同じ最上位デザインで複数回インスタンス化できる場合、配置制約は追加しないでください。

制約の効率

制約の適用範囲の確認

タイミング制約を記述する際は、制約を簡潔なものにし、関連のネットリスト オブジェクトのみに設定することが重要です。非効率な制約を設定すると、実行時間が長くなり、メモリ消費量も大きくなります。また、タイミング例外が予測以上に多くのパスに適用されてほかの制約と競合し、デザインが不適切に制約される結果となることがあります。

タイミング制約を効率的なものにするには、目的のタイミング パスを正確に安全に制約するために指定するオブジェクト数をできるだけ少なくします。オブジェクトのリストはピンまたはセルの名前パターンから取得されることがほとんどなので、最大の効率を得ることはできませんが、タイミング例外用のオブジェクトをリストする際は、オブジェクト数をできるだけ少なくすることを目指してください。

Vivado では、タイミング例外を複数の方法でチェックできます。

- 設計手法チェック XDCB-1 (report_methodology): 多数 (> 1000) のオブジェクトを参照するタイミング制約をレポートします。
- [Report Exception] コマンド (report_exceptions): 定義されたタイミング例外の適用範囲および競合の情報をレポートします。

ザイリンクスでは、次のレポートを注意深く解析することをお勧めします。

- report_exceptions -coverage

このレポートは、各タイミング例外の論理パスの適用範囲を示します。タイミング例外に渡されるオブジェクトの数が、実際に制約が適用される始点と終点の数と比較されます。オブジェクトの数と始点/終点の数が大幅に異なる制約は、見直す必要があります。

- `report_exceptions -ignored`

このレポートは、ほかのタイミング制約が優先されたために無効になったタイミング制約をレポートします (`set_false_path` が `set_clock_group` により無効になるなど)。無効になった制約が正しいかを確認し、不要な制約は削除する必要があります。

- `report_exceptions -ignored_objects`

このレポートは、始点からのパスまたは終点へのパスが存在しないなどの理由で無視された始点および終点をリストします。

制約の実行時間の短縮

ピンのクエリの最適化

デザインのピン数はセル数の数倍なので、`get_cells` の代わりに `get_pins` を使用すると、実行時間に大きく影響する可能性があります。実行時間は、XDC 制約の処理時 (`open_checkpoint` の実行時間など) または Tcl スクリプトの実行時に長くなります。ピンオブジェクトとセルオブジェクトの関係を利用して、多数のピンをクエリする際の実行時間を短縮することをお勧めします。

デザインに含まれるすべてのピンから名前に基づいてピンを検索するよりも、まずピンのセルを検索し、返されたセルのピンをフィルターを使用して検索の方が効率的です。次にその例を示します。

推奨されるピンのクエリ方法

元のピンのクエリ:

```
get_pins -hier * -filter {NAME=~xx*/yy*}
```

推奨される効率的なピンのクエリ:

```
get_pins -filter {REF_PIN_NAME=~yy*} -of [get_cells -hier xx*]
```

推奨される別のピンのクエリ:

```
get_pins -filter {REF_PIN_NAME=~yy*} -of [get_cells -hier * -filter {NAME=~xx*}]
```

例

たとえば、次のような制約があるとします。

```
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
```

この制約を次のように記述し直すと、特に大型デザインで、クエリにかかる時間を大幅に短縮できます。

```
set_max_delay 15 -from [get_pins -of [get_cells -hier -filter
                                {NAME =~ *aclk_dpram_reg*/*/}] -filter {REF_PIN_NAME == CLK}] \
                    -to [get_cells -hier bclk_dout_reg*] \
                    -datapath_only
```

all_registers クエリの置換

次に、クエリに関するその他の推奨事項を示します。

- all_registers を使用したクエリは多数のオブジェクトを返すので、できるだけ使用しないようにしてください。このようなクエリは、適切な名前パターンを使用したセル/ピンのクエリに置き換える必要があります。
- all_registers をどうしても使用する必要があり、このクエリであるクロックドメインからのすべての順次エレメントを取得する場合は、all_registers -clock を使用する場合とクロック オブジェクトを直接使用する場合とで適用範囲が同じになることがあります。

たとえば、次の 2 つのコマンドの適用範囲は同じですが、get_clocks を使用する 2 つ目のコマンドは、マルチサイクルパス制約で膨大な数の順次エレメントではなく 1 つのクロック オブジェクトを参照しているので、より効率的です。

元の制約:

```
set_multicycle_path -from [all_inputs] -to [all_registers -clock clk1]
```

効率的な制約:

```
set_multicycle_path -from [all_inputs] -to [get_clocks clk1]
```



重要: Vivado Design Suite 2018.3 から、all_registers コマンドは、少なくとも 1 つのイネーブルにされたセットアップ/ホールド/リカバリ/リムーバブル タイミング アークおよび CLK->Q タイミング アークを含むプリミティブしか返さなくなりました。つまり、BUFGCE および BUFGCE_DIV などのバッファは all_registers コマンドでは返されなくなりました。

実行時間を短縮するための制約の順序

タイミング制約をメモリに読み込む際、タイミング エンジンで新しい各制約が検証され、発生する可能性のある問題がレポートされます。タイミング制約によっては、タイミング データベース (タイミング グラフとも呼ばれる) を部分的に無効にするものや、正しく適用するために最新のタイミング データベースが必要なものがあります。タイミング データベースが最新でなくなると、たとえば自動派生クロックをアップデートしたりデザインの特定のタイミング パスをディスエーブルにするために、タイミングのアップデートが必要となります。クロックをクエリする XDC コマンドやデザイン全体でネットリスト オブジェクトをクエリする XDC コマンドでは、最新のタイミング データベースが必要です。

タイミング データベースの状態に影響する制約とコマンドを交互に使用すると、タイミング情報が複数回アップデートされるため、実行時間が長くなる可能性があります。

実行時間を短縮するには、タイミング制約およびクエリの順序に注意する必要があります。次の表に、タイミンググラフに影響する XDC 制約およびコマンドを示します。

表 2-5: XDC 制約とそのタイミング グラフへの影響

タイミング グラフに影響する制約	タイミング グラフに影響しない制約	最新のタイミング グラフが必要な制約
create_clock	set_bus_skew	all_fanout
create_generated_clock	set_clock_groups	all_fanin
set_case_analysis	set_clock_latency	get_clocks
set_clock_sense	set_false_path	get_generated_clocks
set_clock_uncertainty	set_input_delay	all_clocks
set_disable_timing	set_input_jitter	-clock オプションを使用する制約
set_external_delay	set_min_delay	
set_propagated_clock	set_max_delay	
	set_max_time_borrow	
	set_multicycle_path	
	set_system_jitter	

実行時間が最も長くなる組み合わせの 1 つは、set_disable_timing と all_fanout または all_fanin です。このような組み合わせは使用しないようにしてください。次に例を示します。

```
set_disable_timing -from <pin> -to [all_fanout ...]
set_disable_timing -from [all_fanin ...] -to <pin>
```

実行時間を短縮するための表 2-5 に基づく最適な制約の順序は、次のとおりです。

1. XDC 制約 set_disable_timing、set_case_analysis、および set_external_delay。
2. タイミング グラフに影響する制約。
3. タイミング グラフのアップデートが不要な制約。



ヒント: 同じクエリを複数の場所で実行する場合は、クエリの結果を Tcl 変数として保存し、必要に応じてその Tcl 変数を参照するようにすることをお勧めします。

たとえば、次の制約順は最適なものではありません。

```
create_clock -name clk1
create_generated_clock -name genclk1 -master_clock [get_clocks -of [get_pins ...]]
set_disable_timing ...
create_clock -name clk2
set_false_path -from [get_clocks -of [get_pins ff1/C]]
set_case_analysis ...
create_clock -name clk3
set_max_delay -to [get_clocks -of [get_pins ff2/C]]
```

これを実行時間の面から効率的な順序にすると、次のようになります。

```
set_disable_timing ...
set_case_analysis ...
create_clock -name clk1
create_clock -name clk2
create_clock -name clk3
create_generated_clock -name genclk1 -master_clock [get_clocks -of [get_pins ...]]
set_false_path -from [get_clocks -of [get_pins ff1/C]]
set_max_delay -to [get_clocks -of [get_pins ff2/C]]
```


クロックの定義

クロックについて

デジタル デザインでは、レジスタからレジスタにデータを転送するための時間の基準となるのがクロックです。Vivado® 統合設計環境 (IDE) のタイミング エンジンには、クロックの特性を使用してタイミング パス要件を算出し、スラックを算出してデザインのタイミング マージンをレポートします。

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) [\[参照 4\]](#) のこのセクションを参照してください。

タイミング パスをできるだけ正確に、最大限に網羅するには、クロックを正しく定義する必要があります。クロックは次の特性により定義されます。

- そのツリー ルートのドライバー ピンまたはポート (起点) で定義されます。
- エッジは、周期と波形の特性で表現されます。
- 周期はナノ秒 (ns) で定義し、これは、波形が繰り返す間隔を表します。
- 波形は、クロック周期内の立ち上がりエッジおよび立ち下がりエッジの絶対時間 (ns) のリストです。波形のリストには偶数個の値を含める必要があります。最初の値は最初の立ち上がりエッジを示します。値を指定しない場合、デューティ サイクルはデフォルトで 50% になり、位相シフトは 0 ns になります。

[図 3-1](#) の例では、クロック Clk0 は周期が 10 ns、デューティ サイクルが 50%、位相シフトが 0 ns です。Clk1 クロックの周期は 8 ns、デューティ サイクルは 75% (8 ns 中 High の時間が 6 ns)、立ち上がりエッジの位相シフトは 2 ns です。

```
Clk0: period = 10, waveform = {0 5}
Clk1: period = 8, waveform = {2 8}
```

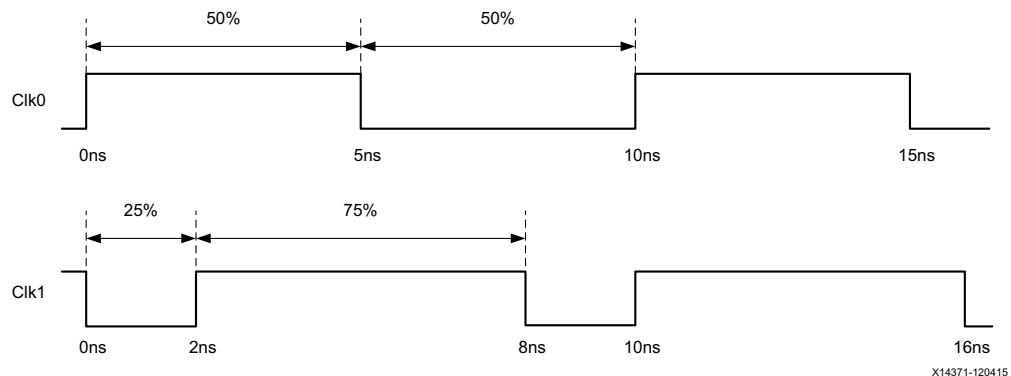


図 3-1: クロック波形の例

伝搬されたクロック

周期および波形は、クロックの理想的な特性を表します。クロックが FPGA に入力され、クロック ツリーを介して伝搬されると、クロック エッジに遅延が発生し、ノイズおよびハードウェアの動作により変動する可能性があります。これらはクロック ネットワーク レイテンシおよびクロックのばらつきと呼ばれます。

クロックのばらつきには、次のものが含まれます。

- クロック ジッター (「[クロック ジッター](#)」を参照)
- 位相エラー
- ユーザーが指定したその他のばらつき (「[その他のクロックのばらつき](#)」を参照)

Vivado IDE では、クロックはデフォルトでレイテンシおよびばらつきを含む伝搬されたクロックとして処理され、クロック ツリー挿入遅延およびばらつきを含む正確なスラック値が算出されます。

専用ハードウェア リソース

ザイリックス FPGA には専用ハードウェア リソースがあり、多数のデザイン クロックを効率的にサポートできます。これらのクロックは、通常ボード上の外部コンポーネントにより生成され、入力ポートからデバイスに供給されます。

クロックは、クロック調整ブロックと呼ばれる次のプリミティブでも生成できます。

- MMCM
- PLL
- BUFR

LUT やレジスタなどの通常のセルでクロックを変換することも可能です。

次のセクションで、クロックのタイプ別に最適なクロック定義方法を説明します。

プライマリ クロック

プライマリ クロックは、入力ポートまたはギガビット トランシーバー出力ピン (復元されたクロックなど) を介してデザインに入力されるボード クロックです。

プライマリ クロックは `create_clock` コマンドでしか定義できません。

注記: プライマリ クロックは、ザイリックス 7 シリーズ FPGA の場合にのみ、ギガビット トランシーバー出力に定義する必要があります。UltraScale および UltraScale+™ デバイスの場合、タイマーは GT 出力ポートでクロックを自動的に派生します。

プライマリ クロックは、ネットリスト オブジェクトに接続する必要があります。このネットリスト オブジェクトは、すべてのクロック エッジの起点を表し、ここからクロックがクロック ツリーのダウンストリームに伝搬されます。つまり、Vivado IDE でスラック値の算出に使用されるクロック レイテンシおよびばらつきを求めるときに、プライマリ クロックの起点が時間 0 になります。



重要: Vivado IDE では、プライマリ クロックが定義されている起点よりアップストリームにあるセルからのクロック ツリー遅延は無視されます。デザインの中央にあるピンにプライマリ クロックを定義すると、タイミング解析では一部のレイテンシしか使用されません。このクロックがデザインのほかの関連クロックと通信する場合、クロック 間のスキュー、そして結果的にスラックの値が不正確になる可能性があるため、これは問題です。

ほかのタイミング制約はプライマリ クロックを基準にすることが多いので、まずプライマリ クロックを定義する必要があります。

プライマリ クロックの例

図 3-2 では、ボード クロックはポート sysclk からデバイスに入力され、入力バッファおよびクロック バッファを介してパス レジスタに到達します。

- 周期: 10 ns
- デューティ サイクル: 50%
- 位相シフト: なし

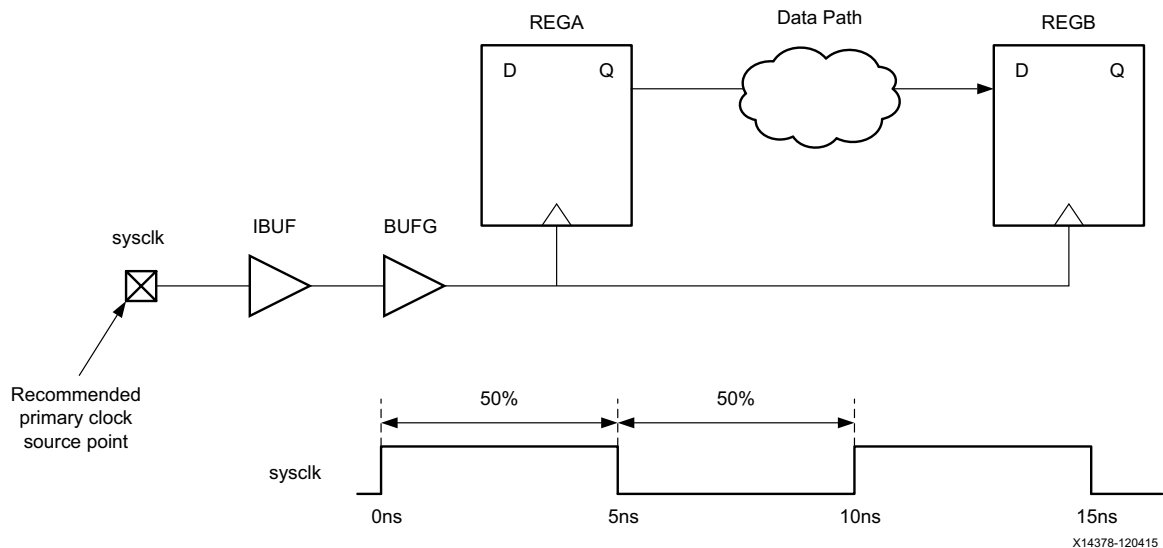


図 3-2: プライマリ クロックの例



推奨: ボード クロックは、クロック バッファの出力ではなく入力ポートに定義します。

対応するザイリンクス デザイン制約 (XDC)

```
create_clock -period 10 [get_ports sysclk]
```

sysclk と同様に、ボード クロック devclk は、ClkIn ポートからデバイスに入力されます。

- 周期: 10 ns
- デューティ サイクル: 25%
- 位相シフト: 90 度

対応する XDC:

```
create_clock -name devclk -period 10 -waveform {2.5 5} [get_ports ClkIn]
```

図 3-3 は、トランシーバー gt0 を示しています。このトランシーバーは、ボードの高速リンクからクロック rxclk をリカバリします。クロック rxclk の周期は 3.33 ns、デューティサイクルは 50% で、デザイン用に補正されたクロックを複数生成する MMCM に配線されています。

GT0 の出力ドライバー ピンに rxclk を定義すると、MMCM で駆動されるすべての生成クロックの起点は gt0/RXOUTCLK になります。これらの間のパスの Slack 算出には、正しいクロック レイテンシおよびばらつきの値が使用されます。

```
create_clock -name rxclk -period 3.33 [get_pins gt0/RXOUTCLK]
```

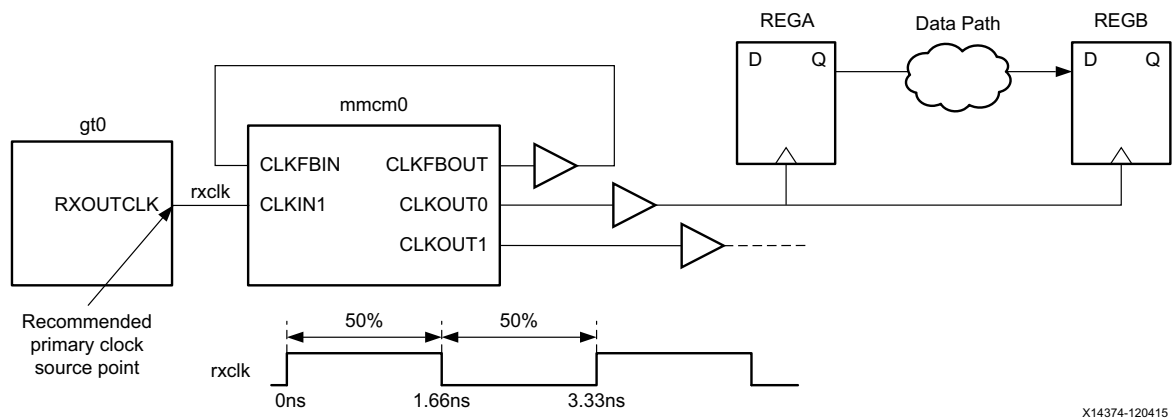


図 3-3: GT プライマリ クロックの例

図 3-4 に、差動バッファで PLL を駆動する例を示します。この場合、プライマリ クロックは差動バッファの正の入力にのみ作成します。バッファの正と負の両方の入力にそれぞれプライマリ クロックを作成すると、非現実的な CDC パスが作成されます。次に例を示します。

```
create_clock -name sysclk -period 3.33 [get_ports SYS_CLK_clk_p]
```

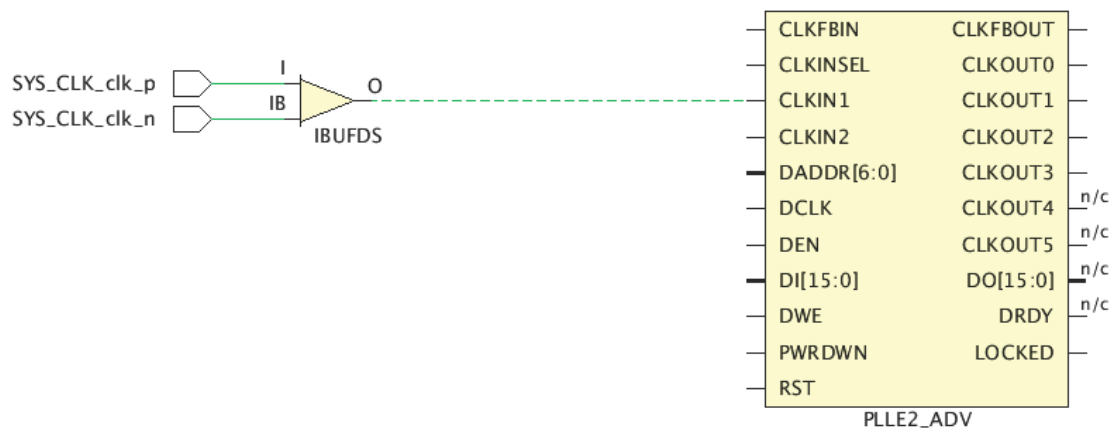


図 3-4: 差動バッファにプライマリ クロックを設定する例

仮想クロック

仮想クロックは、デザインのどのネットリスト エlement にも物理的に接続されていないクロックです。

仮想クロックを定義するには、`create_clock` コマンドをソースを指定せずに使用します。

一般的に、仮想クロックは次のような場合に、入力遅延および出力遅延を指定するのに使用されます。

- 外部デバイス I/O の基準クロックがデザイン クロックのいずれでもない。
- FPGA I/O パスが内部で生成されたクロックに関連付けられており、そのクロックが派生したボード クロックを基準にして適切にタイミング解析できない。

注記: この状況は 2 つの周期の比が整数ではない場合に発生し、タイミング パス要件が非常に厳しくなったり、非現実的なものになったりすることがあります。

- 内部クロックの特性を変更せずに、I/O 遅延制約に関連するクロックに異なるジッターおよびレイテンシを指定する場合。

たとえば、周期が 10 ns のクロック `clk_virt` があり、どのネットリスト オブジェクトにも接続されていないとします。この場合、[<objects>] 引数は指定されず、`-name` オプションが必須となります。

```
create_clock -name clk_virt -period 10
```

仮想クロックは、入力および出力遅延制約で使用する前に定義する必要があります。

生成クロック

このセクションでは、生成クロックについて説明します。

- 「生成クロックについて」
- 「ユーザー定義の生成クロック」
- 「自動派生クロック」
- 「自動生成クロックの名前の変更」

生成クロックについて

生成クロックは、MMCM などのクロック調整ブロックと呼ばれる特別なセルまたはユーザー ロジックにより駆動されます。

生成クロックは、マスター クロックに関連付けられています。create_generated_clock コマンドには、マスター クロックの始点を指定します。マスター クロックには、プライマリ クロックまたはほかの生成クロックを指定できます。

生成クロックのプロパティは、マスター クロックから直接派生します。周期または波形を指定するのではなく、調整回路がマスター クロックをどのように変換するかを記述する必要があります。

マスター クロックと生成クロックの関係は、次のいずれかで定義できます。

- 単純な周波数の分周
- 単純な周波数の通倍
- 周波数の分周と通倍の組み合わせで整数以外の比を生成 (通常 MMCM または PLL を使用)
- 位相シフトまたは波形の反転
- デューティ サイクルの変換
- 上記すべての組み合わせ



推奨: まず、すべてのプライマリ クロックを定義してください。生成クロックを定義するには、プライマリ クロックが必要です。

注記: 生成クロックのレイテンシを算出するため、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 ピンの間のシーケンシャル パスおよび組み合わせパスがトレースされます。生成クロックのレイテンシを算出するのに、組み合わせパスのみをトレースの方が望ましいこともあります。その場合は、-combinational オプションを使用します。

ユーザー定義の生成クロック

ユーザー定義の生成クロックは、次のようなクロックです。

- `create_generated_clock` コマンドで定義されている。
- ネットリスト オブジェクト (理想的にはクロック ツリーのルート ピン) に接続されている。

`-source` オプションを使用してマスター クロックを指定します。このオプションには、マスター クロックが伝搬されるピンまたはポートを指定します。一般的には、マスター クロックの起点または生成クロックのソース セルの入力クロック ピンが使用されます。



重要: `-source` オプションには、ピンまたはポートのネットリスト オブジェクトのみを指定します。クロック オブジェクトは指定できません。

例 1: 単純な 2 分周

プライマリ クロック `clkin` の周期は `10 ns` です。このクロックはレジスタ `REGA` で 2 分周され、ほかのレジスタのクロック ピンを駆動します。対応する生成クロックは `clkdiv2` と呼ばれます。

この生成クロックを指定する 2 つの例を次に示します。

```
create_clock -name clkin -period 10 [get_ports clkin]

# Option 1: master clock source is the primary clock source point
create_generated_clock -name clkdiv2 -source [get_ports clkin] -divide_by 2 \
[get_pins REGA/Q]

# Option 2: master clock source is the REGA clock pin
create_generated_clock -name clkdiv2 -source [get_pins REGA/C] -divide_by 2 \
[get_pins REGA/Q]
```

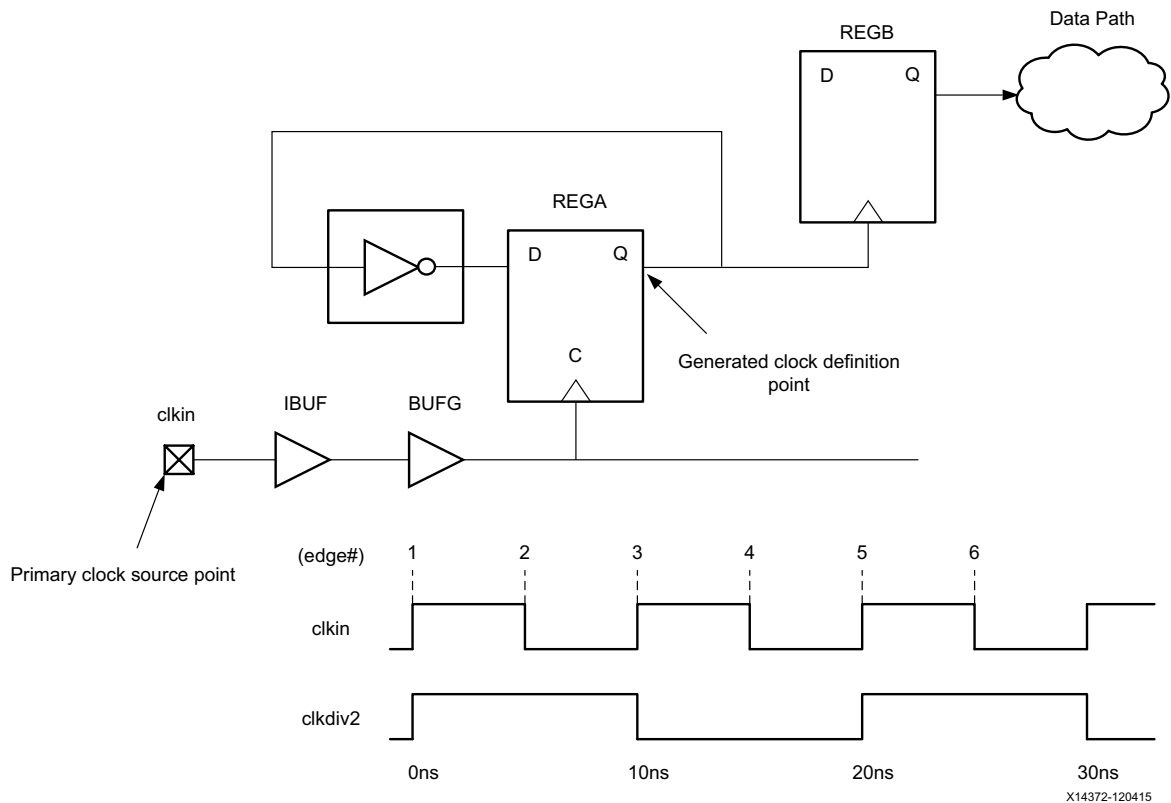


図 3-5: 生成クロックの例 1

例 2: -edges オプションを使用した 2 分周

-divide_by オプションの代わりに -edges オプションを使用すると、マスター クロックのエッジに基づいて生成クロックの波形を定義できます。引数はマスター クロック エッジのインデックスのリストで、生成クロックのエッジの時間位置を立ち上がりクロック エッジから順に定義します。

次の例では、「例 1: 単純な 2 分周」で定義した生成クロックと同じものを定義しています。

```
# waveform specified with -edges instead of -divide_by
create_generated_clock -name clkdiv2 -source [get_pins REGA/C] -edges {1 3 5} \
[get_pins REGA/Q]
```

例 3: -edges および -edge_shift オプションを使用したデューティ サイクルおよび位相シフトの指定

生成クロックの波形の各エッジは、-edge_shift で正または負の値を指定することにより個別にシフトさせることもできます。位相シフトが必要な場合は、このオプションを使用します。

-edge_shift オプションは、次のオプションと同時に使用することはできません。

- -divide_by
- -multiply_by
- -invert

たとえば、clkkin というマスター クロックがあり、周期は 10 ns、デューティ サイクルは 50% であるとします。このクロックはセル mmcm0 に入力され、このセルによりデューティ サイクルが 25% で位相が 90 度シフトされたクロックが生成されます。生成クロックの定義には、マスター クロックのエッジ 1、2、および 3 が使用されています。これらのエッジは、それぞれ 0 ns、5 ns、および 10 ns で発生します。適切な波形を得るには、1 番目と 3 番目のエッジを 2.5 ns シフトします。

```
create_clock -name clkkin -period 10 [get_ports clkkin]
create_generated_clock -name clkshift -source [get_pins mmcm0/CLKIN] -edges {1 2 3} \
    -edge_shift {2.5 0 2.5} [get_pins mmcm0/CLKOUT]
# First rising edge: 0ns + 2.5ns = 2.5ns
# Falling edge: 5ns + 0ns = 5ns
# Second rising edge: 10ns + 2.5ns = 12.5ns
```

注記: -edge_shift には、正の値または負の値を指定できます。

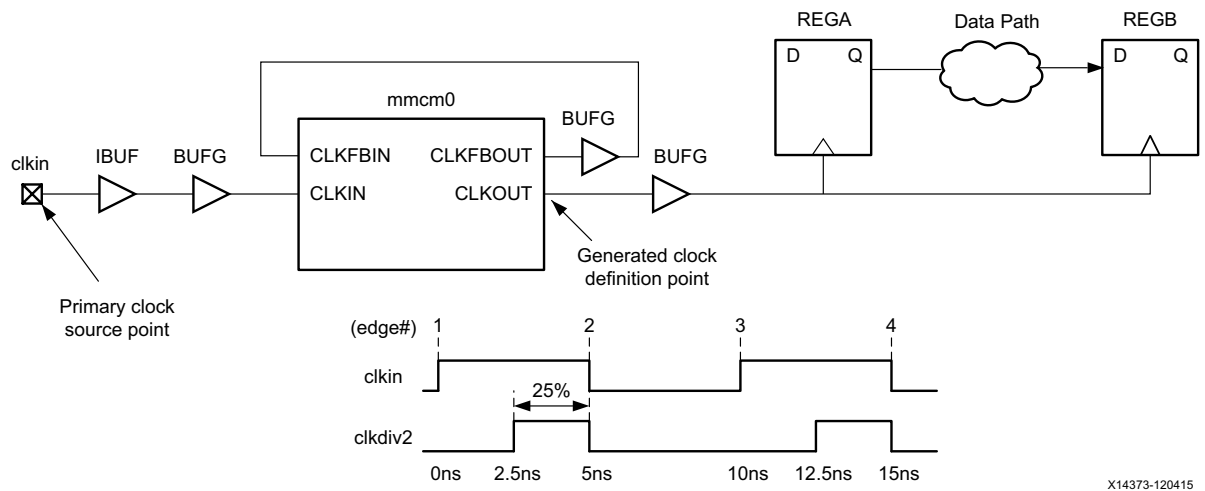


図 3-6: 生成クロックの例 3

例 4: -divide_by と -multiply_by オプションを同時に使用

Vivado IDE では、-divide_by と -multiply_by を同時に指定できます。これは、標準 SDC (Synopsys Design Constraints) のサポートを拡張したものです。これらのオプションを同時に指定すると、MMCM または PLL インスタンスで生成されたクロックを手動で定義するときに便利ですが、これらの制約はツールで自動的に生成されるようにすることを推奨します。

詳細は、「自動派生クロック」を参照してください。

たとえば、「例 3: -edges および -edge_shift オプションを使用したデューティ サイクルおよび位相シフトの指定」の mmcm0 セルでマスター クロックの周波数が 4/3 で通倍されるとすると、この生成クロックの定義は次のようになります。

```
create_generated_clock -name clk43 -source [get_pins mmcm0/CLKIN] -multiply_by 4 \
    -divide_by 3 [get_pins mmcm0/CLKOUT]
```

MMCM/PLL の出力に生成クロック制約を作成する場合は、波形の定義が MMCM/PLL の設定に一致していることを確認してください。

例 5: マスター クロックを組み合わせアークのみを介してトレース

この例では、マスター クロックは、2 分周クロックを生成するレジスタ分周器と、マスター クロックがレジスタ クロック分周器からの 2 分周クロックのいずれかを選択するクロック マルチプレクサーを駆動しています。マスター クロックからのパスには、シーケンシャル アークを介するものと組み合わせアークを介するものの 2 つがあります。マルチプレクサー出力に、マスター クロックからマルチプレクサーを介する組み合わせパスのレイテンシを考慮した生成クロックを作成するには、-combinational オプションを使用します。

```
create_generated_clock -name clkout -source [get_pins mmcm0/CLKIN] -combinational
[get_pins MUX/O]
```

例 6: ODDR で駆動されるフォワード クロック

この例では、ODDR セルで駆動される出力ポートにフォワード クロックを作成しています。このフォワード クロックは ODDR/CLKDIV ピンを駆動するマスター クロックを参照しており、周期はマスター クロックと同じです (-divide_by 1)。

```
create_generated_clock -name ck_vsf_clk_2 \
-source [get_pins ODDR1_vsfclk2_inst/CLKDIV] -divide_by 1 [get_ports vsf_clk_2]
```

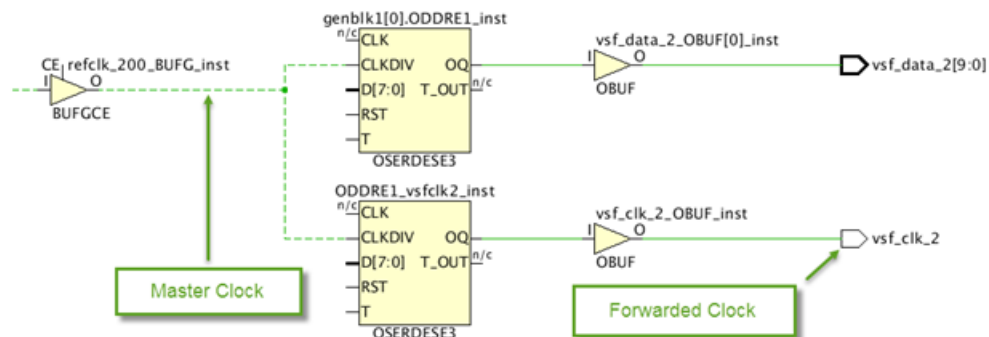


図 3-7: フォワード クロックの例

自動派生クロック

自動派生クロックは、自動生成クロックとも呼ばれます。Vivado IDE では、関連付けられているマスター クロックが既に定義されている場合、クロック調整ブロック (CMB) の出力ピンに自動的に制約が作成されます。

ザイリンクス 7 シリーズ デバイス ファミリの CMB は次のとおりです。

- MMCM*/PLL*
- BUFR
- PHASER*

ザイリンクス UltraScale™ デバイス ファミリの CMB は次のとおりです。

- MMCM*/PLL*
- BUFG_GT/BUFGCE_DIV
- GT*_COMMON/GT*_CHANNEL/IBUFDS_GTE3
- BITSlice_CONTROL/RX*_BITSlice
- ISERDESE3

同じネットリストオブジェクト (同じ定義点 (ネットまたはピン)) に既にユーザー定義クロック (プライマリまたは生成クロック) が定義されている場合は、クロックは自動生成されません。自動派生クロックには、定義点に接続されているネットの最上位セグメント名に基づいた名前が付けられます。

自動派生クロックの例

次に、MMCME2 により生成されたクロックの例を示します。

マスター クロック `clkIn` は、MMCME2 インスタンス `clkip/mmcm0` の `CLKIN` 入力を駆動します。自動生成クロックの名前は `cpuClk` で、その定義点は `clkip/mmcm0/CLKOUT` です。

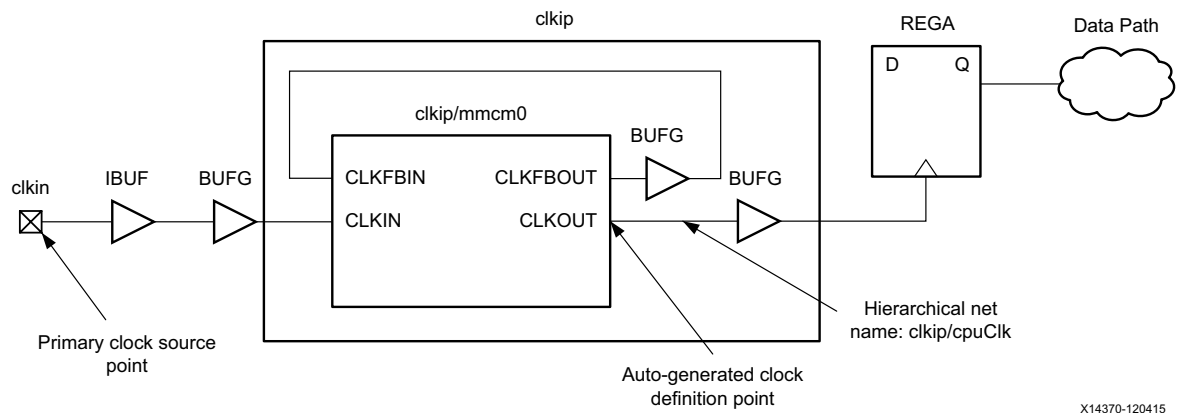


図 3-8: 自動生成クロックの例



ヒント: 自動生成クロックの名前がわからない場合は、`get_clocks -of_objects <pin/port/net>` コマンドを使用してクエリします。このコマンドは、クロック名の変更ににかかわらず、制約またはスクリプトで使用できます。

ローカル ネット名

CMB インスタンスがデザインの階層内に配置されている場合、生成クロックの名前にローカル ネット名 (親セル名を含まない名前) が使用されます。

たとえば、`clkip/cpuClk` という階層ネットの場合は次のようになります。

- 親セル名は `clkip` です。
- 生成クロック名は `cpuClk` です。

名前の競合

2つの自動生成クロックの名前が競合する場合、Vivado IDE で次のように名前を区別する接尾辞が付けられます。

- `usrclk`
- `usrclk_1`
- `usrclk_2`
- ...

生成クロックの名前を指定するには、次のいずれかの方法を使用します。

- RTL で重複しない、わかりやすいネット名を選択します。
- 生成クロックの名前を指定するには、`create_generated_clock` を使用します。

自動生成クロックの名前の変更

ツールで自動生成されたクロックの名前は変更できます。これには、`create_generated_clock` コマンドでいくつかのオプションを指定します。

```
create_generated_clock -name new_name [-source master_pin] [-master_clock
master_clk] source_object
```

必須の引数は、新しい生成クロックの名前と生成クロックのソース オブジェクトです。生成クロックのソース オブジェクトは、自動派生クロックが作成されたオブジェクト (CMB 出力ピン、UltraScale の場合は GT 出力ピンなど) です。`-source` および `-master` オプションは、ソース ピンを伝搬するクロックが複数ある場合に、明確にするために指定する必要があります。



重要: `-edges`、`-edge_shift`、`-divide_by`、`-multiply_by`、`-combinational`、`-duty_cycle`、`-invert` のいずれかのオプションを `create_generated_clock` コマンドに使用すると、生成クロックの名前は変更されません。指定の特性を持つクロックが新たに生成されます。



重要: モジュール (IP/BD/DFx/...) がアウト オブ コンテキストで合成されると、そのモジュールは最上位が合成される際にブラック ボックとして推論され、モジュールの内部ピンおよびクロック名にアクセスできなくなります。この場合、合成に使用される最上位 XDC 制約は、クロック名を参照したり、モジュール内で生成される自動派生クロックの名前を変更したりすることはできません。OOC 合成では、最上位タイミング制約は、これらのクロックを伝搬するモジュールポートを介して OOC クロックに指定する必要があります。これは、`'get_clocks -of_objects [get_pins <OOC_MODULE_OUTPUT_CLOCK_PORT>]` のようなクエリを使用すると実行できます。インプリメンテーションに使用される XDC 制約の場合、XDC 制約が適用される前にデザイン全体が再構築されるため、この制限はありません。

制限

- 自動派生クロックの名前は、PLL や MMCM などのクロック調整ブロックの出力など、自動派生クロックの起点となるピンでのみ変更できます。たとえば、自動派生クロックが BUFG を介して伝搬される場合でも、その BUFG の出力で名前を変更することはできません。
- プライマリ クロックおよびユーザー定義の生成クロックの名前は変更できません。この方法で変更できるのは、自動派生クロックの名前のみです。
- `source_object` には、自動派生クロックが作成されたオブジェクトを指定する必要があります。

生成クロックの名前を変更できない場合、エラーが返されます。また、生成クロックの名前を変更するときに、マスター クロックが存在していることも必要です。自動派生クロックは、別のタイミング制約で参照された後であっても、いつでも XDC 内で変更できます。

次に、MMCM の出力ピンでの生成クロックに対して report_clocks を実行した場合のレポート例を示します。

```
=====
Generated Clocks
=====

Generated Clock   : clkfbout_clk_core
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Multiply By      : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT}

Generated Clock   : clk_rx_clk_core
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Multiply By      : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0}

Generated Clock   : clk_tx_clk_core
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Edges            : {1 2 3}
Edge Shifts      : {0.000 0.500 1.000}
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1}
```

次の3つのコマンドは、MMCM の出力で自動的に派生された3つのクロックの名前を変更するために指定する必要のあるコマンド ライン オプションを示しています。

```
create_generated_clock -name clk_rx [get_pins
clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0]
create_generated_clock -name clk_tx [get_pins
clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1]
create_generated_clock -name clkfbout [get_pins
clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT]
```

名前を変更した後の report_clocks の出力は、次のようになります。

```
=====
Generated Clocks
=====

Generated Clock   : clkfbout
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Multiply By      : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKFBOUT}

Generated Clock   : clk_rx
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Multiply By      : 1
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT0}

Generated Clock   : clk_tx
Master Source    : clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKIN1
Master Clock     : clk_pin_p
Edges            : {1 2 3}
Edge Shifts      : {0.000 0.500 1.000}
Generated Sources : {clk_gen_i0/clk_core_i0/inst/mmcm_adv_inst/CLKOUT1}
```

クロック グループ

このセクションでは、クロック グループについて説明します。次の内容が含まれます。

- 「クロック グループについて」
- 「クロックのカテゴリ」
- 「非同期クロック グループ」
- 「排他的なクロック グループ」

クロック グループについて

Vivado IDE では、クロック グループ制約またはフォルス パス制約が指定されていない場合は、デフォルトで、デザインに含まれるすべてのクロック間のパスのタイミングが解析されます。set_clock_groups コマンドを使用すると、クロック グループ間のタイミング解析がディスエーブルになります。同じグループ内のクロック間のタイミング解析はディスエーブルになりません。set_false_path 制約とは異なり、タイミングはクロック間の両方向で無視されます。

複数のクロック グループを指定するには、-group オプションを複数回使用します。グループ内のクロックがどれもデザインに含まれない場合、そのグループは空になります。set_clock_groups 制約は、少なくとも2つのグループが有効で、空ではない場合にのみ有効です。1つのグループだけが有効でその他すべてのグループが空の場合は、set_clock_groups 制約は適用されず、エラー メッセージが表示されます。

[Schematic] ウィンドウまたは [Report Clock Networks] コマンドを使用してクロック ツリーのトポロジを表示し、どのクロック間のタイミング解析をディスエーブルにする必要があるかを確認してください。また、[Report Clock Interaction] コマンドを使用して、2つのクロック間の既存の制約を確認したり、同じプライマリ クロックを共有しているか (既知の位相関係がある) を判断したり、共通周期のないクロックを特定できます。



注意: 2つのクロック間のタイミング解析を無視しても、それらのクロック間のパスがハードウェアで正しく機能するとは限りません。メタステーブル状態にならないようにするため、これらのパスに再同期化回路または非同期データ転送プロトコルがあることを確認してください。

クロックのカテゴリ

このセクションでは、同期、非同期、共通周期のないクロックについて説明します。

同期クロック

2つのクロックの位相関係が予測可能である場合、これらのクロックは同期しています。クロック ツリーがネットリストの同じルートから出発しており、共通周期がある場合、2つのクロックは通常同期しています。

たとえば、周期の比が2の生成クロックとそのマスター クロックは、生成クロックの起点までは同じネットリストリソースを伝搬されるので同期しており、共通周期は2サイクルです。これらのクロックは、問題なくタイミング解析できます。

非同期クロック

2つのクロックの位相関係を判断できない場合、これらのクロックは非同期です。

たとえば、2つのクロックがボード上の異なるオシレーターで生成され、異なる入力ポートから供給される場合、位相関係は不明なので、これらのクロックは非同期として扱う必要があります。2つのクロックがボード上の同じオシレーターで生成されている場合は、非同期ではありません。

ほとんどの場合、プライマリ クロックどうしは非同期として扱うことができます。プライマリ クロックとそれを基に生成されたクロックをまとめて、非同期クロック グループを構成できます。

共通周期のないクロック

タイミング エンジンで 1000 サイクル以上、2つのクロックの共通周期を検出できなかった場合、これらのクロックは共通周期なしとして処理されます。この場合、1000 サイクルでのワースト セットアップ関係がタイミング解析で使用されますが、これが本当にワースト ケースであるかどうかは、タイミング エンジンでは確認できません。

これは通常、2つのクロックの周期比が半端な分数になる場合に発生します。たとえば、同じプライマリ クロックを共有する2つの MMCM で生成された `clk0` および `clk1` という2つのクロックがあるとします。

- `clk0` の周期は 5.125 ns です。
- `clk1` の周期は 6.666 ns です。

これらのクロックの立ち上がりエッジは 1000 サイクル以内では揃いません。タイミング エンジンでは、これらの2つのクロック間のタイミング パスに、0.01 ns というセットアップ パス要件が使用されます。これら2つのクロックにはクロック ツリールートでは既知の位相関係がありますが、それらの波形のためタイミング解析を安全に実行できません。

非同期クロックと同様、スラックの算出は通常どおりであるように見えますが、値は信用できるものではありません。そのため、共通周期のないクロックは通常、非同期クロックと同様に処理されます。非同期クロックおよび共通周期のないクロックは、制約の設定およびクロック乗せ換え回路では同様に扱う必要があります。

非同期クロック グループ

非同期クロックおよび共通周期のないクロックのタイミングは安全に解析することはできません。これらのクロック間のタイミング パスは、`set_clock_groups` コマンドを使用して解析で無視されるようにできます。



重要: `set_clock_groups` コマンドは、通常のタイミング例外よりも優先されます。非同期クロック間の一部のパスに制約を設定し、それをレポートする必要がある場合は、タイミング例外のみを使用し、`set_clock_groups` は使用しないでください。

非同期クロック グループの例

- プライマリ クロック `clk0` は、入力ポートに定義され、MMCM に入力されて `usrclk` および `itfclk` の2つのクロックが生成されます。
- 2つ目のプライマリ クロック `clk1` は、GTP インスタンスの出力に定義されているリカバリ クロックで、2つ目の MMCM に入力されて `gtclkrx` および `gtclktx` の2つのクロックが生成されます。

非同期クロック グループの作成

-asynchronous オプションを使用して、非同期グループを作成します。

```
set_clock_groups -name async_clk0_clk1 -asynchronous -group {clk0 usrclk itfclk} \
-group {clk1 gtclk rx gtclktx}
```

生成クロックの名前が前もって予測できない場合は、`get_clocks -include_generated_clocks` を使用してダイナミックにそれらを取得します。-include_generated_clocks は SDC の拡張オプションです。

上記の例は、次のようにも記述できます。

```
set_clock_groups -name async_clk0_clk1 -asynchronous \
-group [get_clocks -include_generated_clocks clk0] \
-group [get_clocks -include_generated_clocks clk1]
```

排他的なクロック グループ

デザインによっては、異なるクロックを使用するいくつかの動作モードを持つものがあります。この場合、通常クロックは BUFGMUX や BUFGCTRL などのクロック マルチプレクサーまたは LUT を使用して選択されます。



推奨: クロック ツリーでは LUT をできるだけ使用しないでください。

これらのセルは組み合わせセルであるため、すべての入力クロックは出力に伝搬されます。Vivado IDE では、1 つのクロック ツリーに複数のタイミング クロックを同時に存在させることができ、すべての動作モードに対するレポートを同時に生成できるので便利です。ただし、これはハードウェアではできません。

このようなクロックは排他的クロックと呼ばれ、`set_clock_groups` の次のオプションを使用して制約を設定できます。

- -logically_exclusive
- -physically_exclusive

排他的なクロック グループの例

MMCM インスタンスで `clk0` および `clk1` が生成され、BUFGMUX インスタンス `clkmux` に接続されます。`clkmux` の出力はデザインのクロック ツリーを駆動します。

デフォルトでは、`clk0` および `clk1` クロックが同じクロック ツリーを共有していて、同時に存在できなくても、これら 2 つのクロック間のパスが解析されます。

これらのクロック間のパスの解析をディスエーブルにするには、次の制約を入力します。

```
set_clock_groups -name exclusive_clk0_clk1 -physically_exclusive \
-group clk0 -group clk1
```

次のオプションは、ザイリンクス FPGA では同等です。

- -physically_exclusive
- -logically_exclusive

physically および logically は、ASIC テクノロジーでのさまざまなシグナル インテグリティ解析 (クロストーク) を指しており、ザイリンクス FPGA では必要ありません。

クロック レイテンシ、ジッター、ばらつき

クロック波形の定義に加え、動作条件や環境に関連した、予測可能でランダムな変動も指定する必要があります。

クロック レイテンシ

クロック エッジは、ボードおよびFPGA を伝搬した後、ある遅延でデスティネーションに到達します。この遅延は、通常次のもので表されます。

- ソース レイテンシ (クロック 起点の前、通常デバイス外の遅延)
- ネットワーク レイテンシ

ネットワーク レイテンシによる遅延 (挿入遅延) は、自動的に見積もられるか (配線前)、正確に算出 (配線後) されます。

ザイリンクス以外のタイミング エンジンを使用している場合は通常、クロック ツリーの伝搬遅延が算出されるようにするため、`set_propagated_clock` という SDC コマンドを使用する必要があります。Vivado ツールではこのコマンドは必要ありません。クロック伝搬遅延はデフォルトで算出されます。

- すべてのクロックが伝搬クロックとして処理されます。
- 生成クロックのレイテンシには、マスター クロックの挿入遅延と、その生成クロックのネットワーク レイテンシが含まれます。

ザイリンクス FPGA では、`set_clock_latency` コマンドを使用して、デバイス外のクロック レイテンシを指定してください。

set_clock_latency の例

```
# Minimum source latency value for clock sysClk (for both Slow and Fast corners)
set_clock_latency -source -early 0.2 [get_clocks sysClk]
# Maximum source latency value for clock sysClk (for both Slow and Fast corners)
set_clock_latency -source -late 0.5 [get_clocks sysClk]
```

クロックのばらつき

クロック ジッター

ASIC ではクロック ジッターは通常クロックのばらつき特性で表されますが、ザイリンクス FPGA ではジッター特性は予測可能で、タイミング解析エンジンにより自動的に算出できるほか、別に指定することもできます。

入力ジッター

入力ジッターとは、標準また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。入力ジッターは絶対値で、クロック エッジの各側の変動を表します。

各ブライマリ クロックの入力ジッターを個別に設定するには、`set_input_jitter` コマンドを使用します。生成クロックに直接入力ジッターを設定することはできません。Vivado IDE タイミング エンジンでは、生成クロックがマスター クロックから継承したジッターが自動的に算出されます。

- 生成クロックが MMCM または PLL で駆動される場合は、入力ジッターは算出されたディスクリート ジッターに置き換えられます。
- 生成クロックが組み合わせセルまたはシーケンシャル セルで駆動される場合は、生成クロックのジッターはマスター クロック ジッターと同じになります。

システム ジッター

システム ジッターは、電源ノイズ、ボード ノイズ、またはシステムのその他のジッターによる全体的なジッターです。

デザイン全体、つまりすべてのクロックに対して 1 つの値のみを設定するには、`set_system_jitter` コマンドを使用します。

次のコマンドは、入力ポート `clkin` を介して伝搬されるプライマリ クロックに ± 100 ps のジッターを設定します。

```
set_input_jitter [get_clocks -of_objects [get_ports clkin]] 0.1
```

注記: 全体的なクロックのばらつきの算出における入力ジッターおよびシステム ジッターの影響は、些細なものではなく、1 つの式に従いません。クロックのばらつきの算出はパスによって異なり、クロック トポロジ、パスに関連するクロック ペア、クロック ツリーに MMCM/PLL が存在するかどうか、およびその他の考慮事項に依存します。
[Report Timing] コマンドのテキストおよび GUI レポートには、各タイミング パスのクロックのばらつきの内訳が示されます。

その他のクロックのばらつき

異なるコーナー、遅延、特定のクロック関係に対して追加のクロックのばらつきを定義するには、`set_clock_uncertainty` コマンドを使用します。これは、タイミングの観点から、デザインの部分にゆとりをもたせる便利な方法です。

制約の順序にかかわらず、クロック間のばらつきが単純なクロックのばらつきよりも優先されます。次の例では、クロック `clk1` にクロックのばらつき `1.0 ns` が後に設定されていますが、クロック `clk1` から `clk2` へのタイミングパスにクロックのばらつき `2.0 ns` が設定されています。

```
set_clock_uncertainty 2.0 -from [get_clocks clk1] -to [get_clocks clk2]
set_clock_uncertainty 1.0 [get_clocks clk1]
```

クロック間のばらつきを 2 つのクロック ドメイン間で定義する際は、次のようにクロック ドメインで可能性のあるすべての相互関係を含めるようにしてください。

- `clk1` から `clk2`
- `clk2` から `clk1`

I/O 遅延の制約

I/O 遅延の制約について

デザインの外部タイミングを正確に記述するには、入力ポートおよび出力ポートのタイミング情報を指定する必要があります。Vivado® 統合設計環境 (IDE) では FPGA 内のタイミング情報のみが認識されるので、デバイス外部に存在する遅延値は次のコマンドを使用して指定する必要があります。

- `set_input_delay`
- `set_output_delay`

入力遅延

`set_input_delay` コマンドは、デザインのインターフェイスでのクロック エッジに対して、入力ポートに入力パス遅延を指定します。



ビデオ: 入力遅延については、[Vivado Design Suite QuickTake ビデオ: 入力遅延の設定](#)をご覧ください。

アプリケーション ボードでは、入力遅延は次のものの位相差を表します。

- 外部チップからボードを介して FPGA の入力パッケージ ピンに伝搬されるデータ
- 相対基準ボード クロック

このため、入力遅延は、デバイスのインターフェイスでのクロックおよびデータの位相によって、正または負の値になります。

注記: 入力遅延は、STARTUPE3/DATA_IN[0:3] (UltraScale+™ デバイス) などの内部データ ピンにも設定できます。

入力遅延オプションの使用

`-clock` は、SDC 標準ではオプションですが、Vivado IDE では必須です。相対クロックは、デザイン クロックまたは仮想クロックのいずれかにできます。



推奨: 仮想クロックを使用する場合は、デザイン内の入力ポートに接続されるデザイン クロックと同じ波形を使用してください。このようにすると、タイミング パス要件が現実的なものになります。仮想クロックを使用すると、デザイン クロックを変更せずに、さまざまなジッターやソース レイテンシのパターンを記述できます。

入力遅延コマンドには、次のオプションがあります。

- 「-min および -max オプション」
- 「-clock_fall オプション」
- 「-add_delay オプション」

-min および -max オプション

-min および -max オプションは、次の解析用の値を指定します。

- 最小遅延解析 (ホールド/リムーバル)
- 最大遅延解析 (セットアップ/リカバリ)

どちらのオプションも使用しない場合、入力遅延値は最小値および最大値の両方に適用されます。

-clock_fall オプション

-clock_fall オプションを使用すると、相対クロックの立ち下がりクロック エッジで送信されるタイミング パスに遅延制約が適用されます。このオプションを使用しない場合、相対クロックの立ち上がりエッジのみが考慮されます。

-clock_fall を、-rise および -fall オプションと混同しないでください。-rise および -fall は、クロック エッジではなくデータ エッジに関するオプションです。

-add_delay オプション

-add_delay オプションは、次の場合に使用する必要があります。

- 最大 (または最小) 入力遅延制約が存在する。
- 同じポートに 2 番目の最大 (または最小) 入力遅延制約を指定する必要がある。

このオプションは通常、DDR インターフェイスのように、複数のクロック エッジに対して入力ポートに遅延制約を設定する場合に使用されます。

入力遅延制約は、入力ポートまたは双方向ポートに適用できます。ただし、クロック入力ポートには適用できず、これらは自動的に無視されます。内部ピンには適用できません。

set_input_delay コマンドのオプションの使用

次の例では、set_input_delay コマンドのオプションの典型的な使用方法を示します。入力遅延制約の設計手法に関する追加情報は、『UltraFast 設計手法 (Vivado Design Suite 用)』(UG949) [\[参照 5\]](#) のこのセクションを参照してください。

入力遅延の例 1

次の例では、定義済みの sysClk に対して、最小遅延解析および最大遅延解析の両方の解析に使用する入力遅延を定義しています。

```
> create_clock -name sysClk -period 10 [get_ports CLK0]
> set_input_delay -clock sysClk 2 [get_ports DIN]
```

入力遅延の例 2

次の例では、前に定義した仮想クロックに対して入力遅延を定義します。

```
> create_clock -name clk_port_virt -period 10
> set_input_delay -clock clk_port_virt 2 [get_ports DIN]
```

入力遅延の例 3

次の例では、sysClk に対して min 解析と max 解析用に異なる入力遅延を定義します。

```
> create_clock -name sysClk -period 10 [get_ports CLK0]
> set_input_delay -clock sysClk -max 4 [get_ports DIN]
> set_input_delay -clock sysClk -min 1 [get_ports DIN]
```

入力遅延の例 4

I/O ポート間の完全な組み合わせパスに制約を設定するには、定義済みの仮想クロックに対して I/O ポートに入力遅延および出力遅延を定義します。

次の例では、DIN ポートと DOUT ポート間の組み合わせパスに 5 ns (10 ns - 4 ns - 1 ns) の制約を設定しています。

```
> create_clock -name sysClk -period 10 [get_ports CLK0]
> set_input_delay -clock sysClk 4 [get_ports DIN]
> set_output_delay -clock sysClk 1 [get_ports DOUT]
```

Timing Constraints ウィザードを使用して組み合わせパスに制約を設定する方法については、[37 ページの「\[Combinatorial Delays\] ページ」](#)を参照してください。

入力遅延の例 5

次の例では、DDR クロックに対して入力遅延値を定義します。

```
> create_clock -name clk_ddr -period 6 [get_ports DDR_CLK_IN]
> set_input_delay -clock clk_ddr -max 2.1 [get_ports DDR_IN]
> set_input_delay -clock clk_ddr -max 1.9 [get_ports DDR_IN] -clock_fall -add_delay
> set_input_delay -clock clk_ddr -min 0.9 [get_ports DDR_IN]
> set_input_delay -clock clk_ddr -min 1.1 [get_ports DDR_IN] -clock_fall -add_delay
```

この例では、デバイス外にある clk_ddr 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制約が作成されます。

入力遅延の例 6

この例では、STARTUPE3 からファブリックまでのパスのタイミングを解析するため、STARTUPE3 の内部ピン (UltraScale+ デバイス) にクロックと入力遅延を指定しています。

```
> create_generated_clock -name clk_sck -source [get_pins -hierarchical
*axi_quad_spi_0/ext_spi_clk] [get_pins STARTUP/CCLK] -edges {3 5 7}
> set_input_delay -clock clk_sck -max 7 [get_pins STARTUP/DATA_IN[*]] -clock_fall
> set_input_delay -clock clk_sck -min 1 [get_pins STARTUP/DATA_IN[*]] -clock_fall
```

STARTUPE3 のタイミング制約の詳細は、『AXI Quad SPI LogiCORE IP 製品ガイド』(PG153) [\[参照 6\]](#) を参照してください。

出力遅延

`set_output_delay` コマンドは、デザインのインターフェイスでのクロック エッジに対して、出力ポートに出力パス遅延を指定します。



ビデオ: 出力遅延については、[Vivado Design Suite QuickTake ビデオ: 出力遅延の設定](#)をご覧ください。

アプリケーション ボードでは、この遅延は次のものの位相差を表します。

- a. FPGA の出力パッケージ ピンからボードを介して別のデバイスに伝搬されるデータ
- b. 相対基準ボード クロック

出力遅延は、FPGA 外部のクロックおよびデータの相対位相によって、正または負の値になります。

注記: 出力遅延は、STARTUPE3/DATA_OUT[0:3] (UltraScale+ デバイス) などの内部データ ピンにも設定できます。

出力遅延オプションの使用

`-clock` は、SDC 標準ではオプションですが、Vivado Design Suite では必須です。

相対クロックは、デザイン クロックまたは仮想クロックのいずれかにできます。



推奨: 仮想クロックを使用する場合は、デザイン内の出力ポートに接続されるデザイン クロックと同じ波形を使用してください。このようにすると、タイミング パス要件が現実的なものになります。仮想クロックを使用すると、デザイン クロックを調整しなくても、ジッターまたはソース レイテンシのパターンを記述できます。

出力遅延コマンドには、次のオプションがあります。

- 「`-min` および `-max` オプション」
- 「`-clock_fall` オプション」
- 「`-add_delay` オプション」

`-min` および `-max` オプション

`-min` および `-max` は、最小遅延解析 (ホールド/リムーバル) および最大遅延解析 (セットアップ/リカバリ) に対して、それぞれ異なる値を指定します。どちらのオプションも使用しない場合、出力遅延値は最小値および最大値の両方に適用されます。

`-clock_fall` オプション

`-clock_fall` オプションを使用すると、相対クロックの立ち下がりクロック エッジで受信されるタイミング パスに出力遅延制約が適用されます。このオプションを使用しない場合、デバイス外部の相対クロックの立ち上がりエッジのみが考慮されます。

`-clock_fall` を、`-rise` および `-fall` オプションと混同しないでください。`-rise` および `-fall` は、クロック エッジではなくデータ エッジに関するオプションです。

-add_delay オプション

-add_delay オプションは、次の場合に使用する必要があります。

- 最大出力遅延制約が既に存在する。
- 同じポートに 2 番目の最大出力遅延制約を指定する必要がある。

最小出力遅延制約の場合も同様に使用する必要があります。このオプションは通常、DDR インターフェイスの立ち上がりおよび立ち下がりエッジや、異なるクロックを使用する複数のデバイスに出力ポートが接続されている場合など、複数のクロック エッジに対して出力ポートを制約する場合に使用されます。



重要: 出力遅延制約は、出力ポートまたは双方向ポートにのみ適用できます。内部ピンには適用できません。

set_output_delay コマンドのオプションの使用

次の例では、set_output_delay コマンドのオプションの典型的な使用方法を示します。出力遅延制約の設計手法に関する追加情報は、『UltraFast 設計手法 (Vivado Design Suite 用)』(UG949) [\[参照 5\]](#) のこのセクションを参照してください。

出力遅延の例 1

次の例では、定義済みの sysClk に対して、最小遅延解析および最大遅延解析の両方の解析に使用する出力遅延を定義しています。

```
> create_clock -name sysClk -period 10 [get_ports CLK0]
> set_output_delay -clock sysClk 6 [get_ports DOUT]
```

出力遅延の例 2

次の例では、定義済みの仮想クロックに対して出力遅延を定義します。

```
> create_clock -name clk_port_virt -period 10
> set_output_delay -clock clk_port_virt 6 [get_ports DOUT]
```

出力遅延の例 3

次の例では、DDR クロックに対して、出力遅延を最小遅延 (ホールド) および最大遅延 (セットアップ) 解析で異なる値に指定します。

```
> create_clock -name clk_ddr -period 6 [get_ports DDR_CLK_IN]
> set_output_delay -clock clk_ddr -max 2.1 [get_ports DDR_OUT]
> set_output_delay -clock clk_ddr -max 1.9 [get_ports DDR_OUT] -clock_fall
-add_delay
> set_output_delay -clock clk_ddr -min 0.9 [get_ports DDR_OUT]
> set_output_delay -clock clk_ddr -min 1.1 [get_ports DDR_OUT] -clock_fall
-add_delay
```

この例では、デバイス外部からの clk_ddr クロックの立ち上がり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、立ち上がりおよび立ち下がりの両方のクロック 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制約が作成されます。

出力遅延の例 4

この例では、ファブリックから STARTUPE3 までのパスのタイミングを解析するため、STARTUPE3 の内部ピン (UltraScale+ デバイス) にクロックと出力遅延を指定しています。

```
> create_generated_clock -name clk_sck -source [get_pins -hierarchical  
*axi_quad_spi_0/ext_spi_clk] [get_pins STARTUP/CCLK] -edges {3 5 7}  
> set_output_delay -clock clk_sck -max 6 [get_pins STARTUP/DATA_OUT[*]]  
> set_output_delay -clock clk_sck -min 1 [get_pins STARTUP/DATA_OUT[*]]
```

STARTUPE3 のタイミング制約の詳細は、『AXI Quad SPI LogiCORE IP 製品ガイド』(PG153) [\[参照 6\]](#) を参照してください。

タイミング例外

タイミング例外について

ロジックがデフォルトのままでは正しくタイミングを解析できない場合、タイミング例外を指定する必要があります。2 クロック サイクルごとに結果を取り込むロジックなど、タイミングを別に処理する必要がある場合は、タイミング例外コマンドを使用します。

Vivado® 統合設計環境 (IDE) では、表 5-1 に示すタイミング例外コマンドがサポートされています。

表 5-1: タイミング例外コマンド

コマンド	機能
set_multicycle_path	パスの始点から終点までデータを伝搬させるのに必要なクロック サイクル数を指定します。
set_false_path	デザインに含まれているロジック パスで、解析から除外すべきものを指定します。
set_max_delay set_min_delay	最小パス遅延または最大パス遅延の値を指定します。このコマンドを使用すると、デフォルトのセットアップおよびホールド制約ではなく、ユーザーが指定した最大/最小遅延値が使用されます。

注記: Vivado ツールでは、実行時間を考慮し、タイミング例外の競合はダイナミックには解析されません。タイミング例外の解析およびレポートには、report_exceptions コマンドを実行します。

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 テクニク』(UG906) [参照 4] を参照してください。

マルチサイクルパス

マルチサイクルパス制約を使用すると、デザイン クロック波形に基づいてタイマーにより決定されるセットアップ関係およびホールド関係を変更できます。デフォルトでは、シングル サイクル解析が実行されます。ただし、この解析が厳しすぎ、一部のロジック パスには不適切な場合があります。

たとえば、データが終点で安定するまでに複数クロック サイクル必要な論理パスなどです。パスの始点と終点の制御回路で許容される場合は、マルチサイクルパス制約を使用してセットアップ要件を緩和することをお勧めします。

目的にもよりますが、ホールド要件では元の時間関係が維持される場合もあります。これにより、タイミングドリブン アルゴリズムでタイミング要件の厳しいパスが重点的に処理され、実行時間の短縮にもつながります。

パスの乗数およびクロック エッジの設定

`set_multicycle_path` コマンドは、セットアップ解析、ホールド解析、またはその両方用に、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、パスの乗数を変更するのに使用します。

`set_multicycle_path` の構文

基本的なオプションを使用した `set_multicycle_path` コマンドの構文は、次のとおりです。

```
set_multicycle_path <path_multiplier> [-setup|-hold] [-start|-end]
                    [-from <startpoints>] [-to <endpoints>] [-through <pins|cells|nets>]
```

`<path_multiplier>` は必ず指定する必要があります。デフォルト値は次のとおりです。

- セットアップ (またはリカバリ) 解析では 1
- ホールド (またはリムーバル) 解析では 0

ホールド関係は、セットアップ関係と直接関連しています。最も一般的なケースのホールド サイクル数を得るには、次の式を使用します。

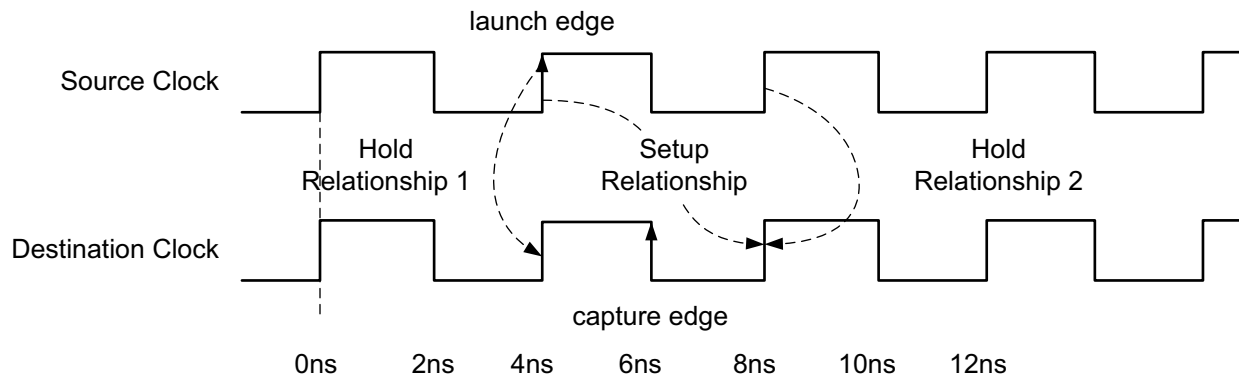
$$\text{Hold cycles} = \text{<setup path multiplier>} - 1 - \text{<hold path multiplier>}$$

- 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` オプションを使用します。
- 同様に、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` オプションを使用します。

注記: 関連用語の定義は、『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) [参照 4] のこのセクションを参照してください。



重要: 各セットアップ関係に対し、2 つのホールド関係があります。(1) 1 つ目のホールド関係は、セットアップ ソース エッジが、アクティブなデスティネーション エッジの前に到着するエッジで取り込まれないようにします。(2) もう 1 つのホールド関係は、アクティブなソース エッジの後のエッジが、アクティブなデスティネーション エッジで取り込まれないようにします。タイミング解析ツールでは両方のホールド関係が算出されますが、解析およびレポートでは厳しい方のホールド関係のみが使用されます。図 5-1 を参照してください。



X14346-061015

図 5-1: パスのセットアップ関係およびホールド関係の例



重要: 同じクロックまたは同一の 2 つのクロック (位相シフトがあるなしにかかわらず同じ波形) 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は、`-start` および `-end` オプションを指定しても効果はありません。

表 5-2 は、`-end` および `-start` オプションがアクティブなソースエッジとデスティネーションエッジに与える影響をまとめています。

表 5-2: アクティブソースエッジおよびデスティネーションエッジ

	ソースクロック (<code>-start</code>) ソースエッジの移動方向	デスティネーションクロック (<code>-end</code>) デスティネーションエッジの移動方向
セットアップ	<---- (前)	----> (先) (デフォルト)
ホールド	----> (先) (デフォルト)	----> (先) (デフォルト)



重要: `set_multicycle_path` コマンドの `-setup` オプションは、セットアップ関係を変更するだけでなく、ホールド関係にも影響します。ホールド関係は、セットアップ関係と直接関連しています。ホールド関係を元の位置に戻すには、別の `set_multicycle_path` 制約を `-hold` を使用して指定する必要があります。

注記: マルチサイクル制約は、1 つのパス、複数のパス、または 2 つのクロック間に設定できます。

次のセクションでは、マルチサイクルパスを使用する一般的な状況と、セットアップおよびホールドの乗数と `-start` および `-end` オプションのタイミングパス要件への影響を示します。

1つのクロックドメインでのマルチサイクル制約

同じクロックドメイン内または同じ波形の2つのクロック間に定義されるマルチサイクル制約は、同じように機能します。図 5-2 を参照してください。

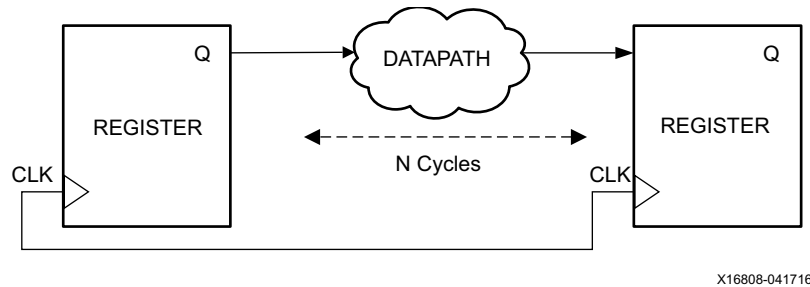


図 5-2: 1つのクロックドメインでのマルチサイクル制約

図 5-3 に、スタティック タイミング解析 (STA) ツールで使用するデフォルトのセットアップ関係とホールド関係を示します。

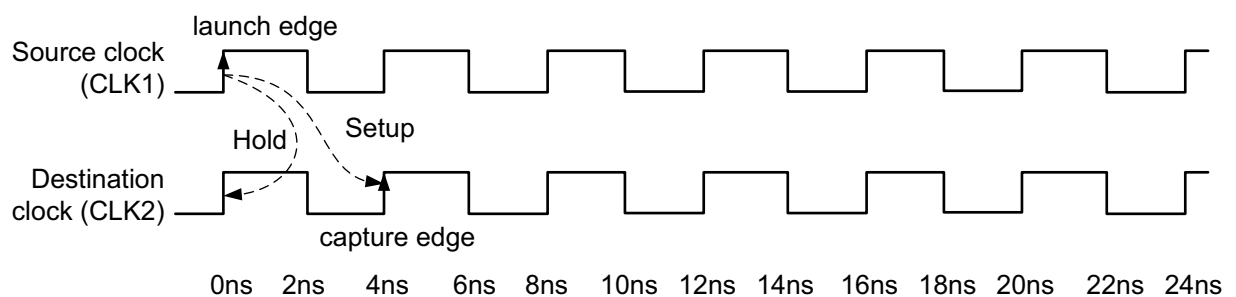


図 5-3: デフォルトのセットアップおよびホールド関係

セットアップおよびホールド タイミング要件は、次のとおりです。

- セットアップ チェック

$$T_{\text{Datapath(max)}} < T_{\text{CLK}(t=\text{Period})} - T_{\text{Setup}}$$
- ホールド チェック

$$T_{\text{Datapath(min)}} > T_{\text{CLK}(t=0)} + T_{\text{Hold}}$$

ホールドを維持しながらセットアップを緩和

図 5-4 に、2 サイクルごとにイネーブルになる 2 つのフリップフロップ間のパスを示します。このパスにマルチサイクルパス制約を定義して、デスティネーションクロックの最初のエッジはアクティブでなく、2 つ目のエッジのみが新しいデータを受信することを示すのが安全です。

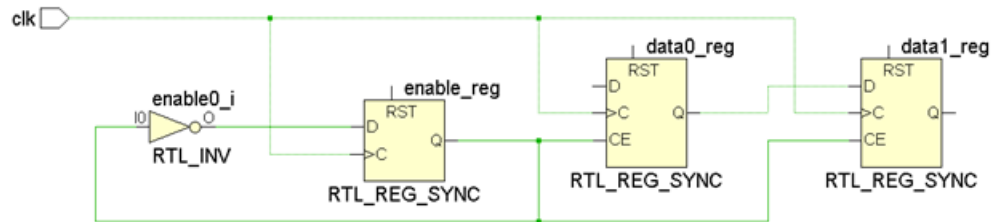


図 5-4: 2 サイクルごとにイネーブルになるレジスタ

次の制約により、新しいセットアップ関係を定義します。

```
set_multicycle_path 2 -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

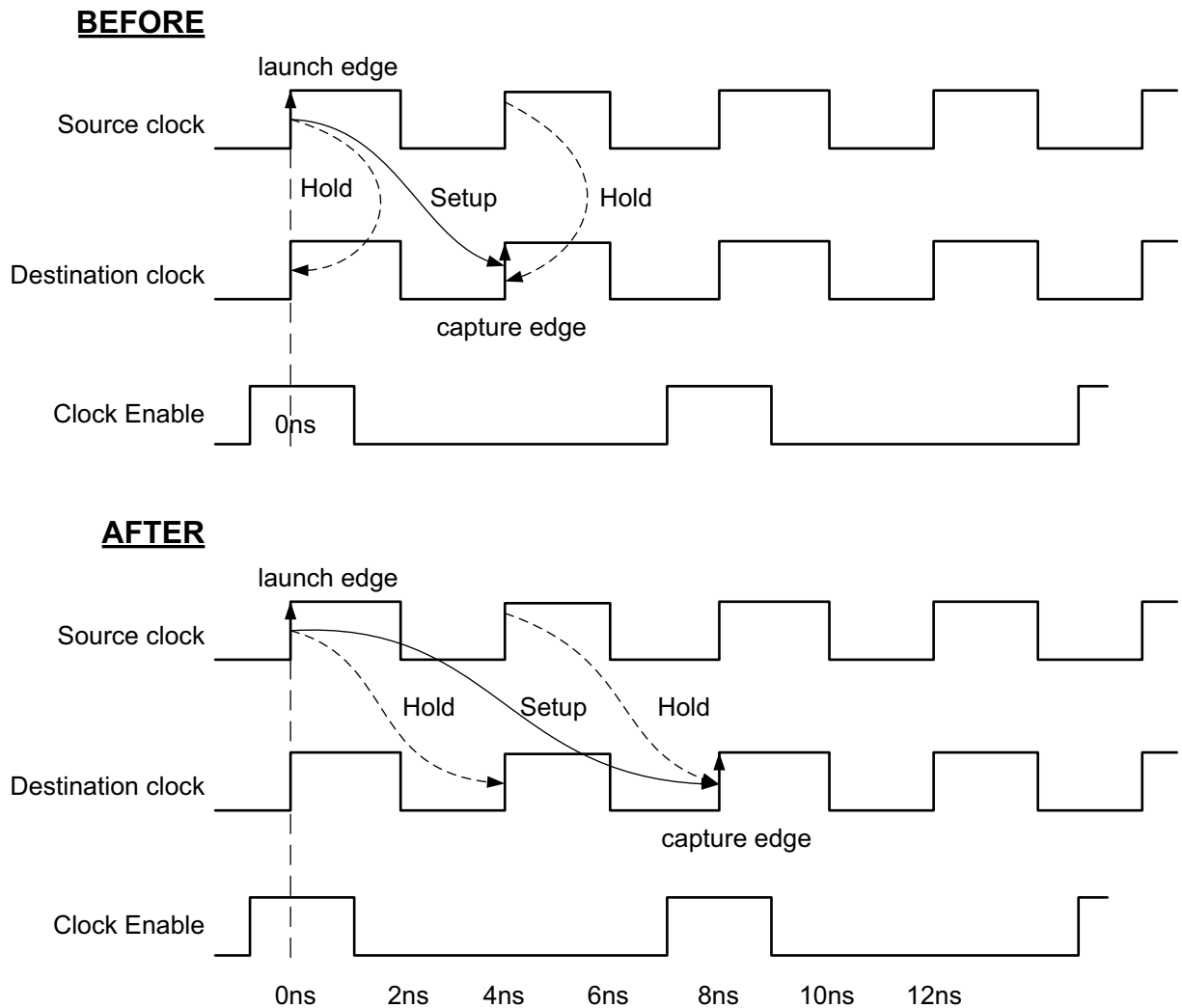
『Vivado Design Suite ユーザー ガイド: デザイン解析およびクロージャ テクニック』(UG906) [参照 4] のこのセクションでは、ホールドの時間関係がどのようにセットアップの時間関係から派生しているかが説明されています。セットアップ関係を変更すると、セットアップのソース エッジおよびデスティネーション エッジの変更に伴い、ホールド関係も変更されます。



重要: 新しいホールド要件が厳しくなりすぎると、タイミング クロージャが困難になります。デザインで可能な場合にホールド要件を緩和するのは、ユーザーが実行する必要があります。

図 5-4 と同じ例で、セットアップチェックを 2 番目のデスティネーション エッジに移動すると、ホールドチェックが自動的に最初のデスティネーション エッジ (セットアップチェックの 1 クロック周期前) に移動します。

図 5-5 に、マルチサイクルパス制約でセットアップ パスの乗数のみを指定した場合に、セットアップ関係とホールド関係の両方がどのように変更されるかを示します。



X14377-120415

図 5-5: マルチサイクルパス: セットアップのみを緩和

このパスでは、クロック イネーブルがあるので、data0_reg でデータを 1 クロック間保持する必要はありません。この場合、ホールド関係を元に戻すことをお勧めします (同じソース エッジとデスティネーション エッジの間)。元に戻すには、2 つ目のマルチサイクルパス制約を追加し、ホールド チェックのみを変更します。

```
set_multicycle_path 1 -hold -end -from [get_pins data0_reg/C] \
-to [get_pins data1_reg/D]
```

set_multicycle -hold コマンドで -end オプションが使用されているのは、デスティネーション クロックのエッジを前に移動する必要があるからです。

注記: ソース クロックとデスティネーション クロックの波形は同じなので、-end の使用はオプションです。デスティネーション エッジを前に移動すると、ソース エッジを先に移動した場合と同じホールド関係が得られます。コマンドをシンプルにするため、-end は次の 2 つの例からは削除されています。

図 5-6 に、両方のマルチサイクルパス制約を適用した後のセットアップ関係とホールド関係を示します。

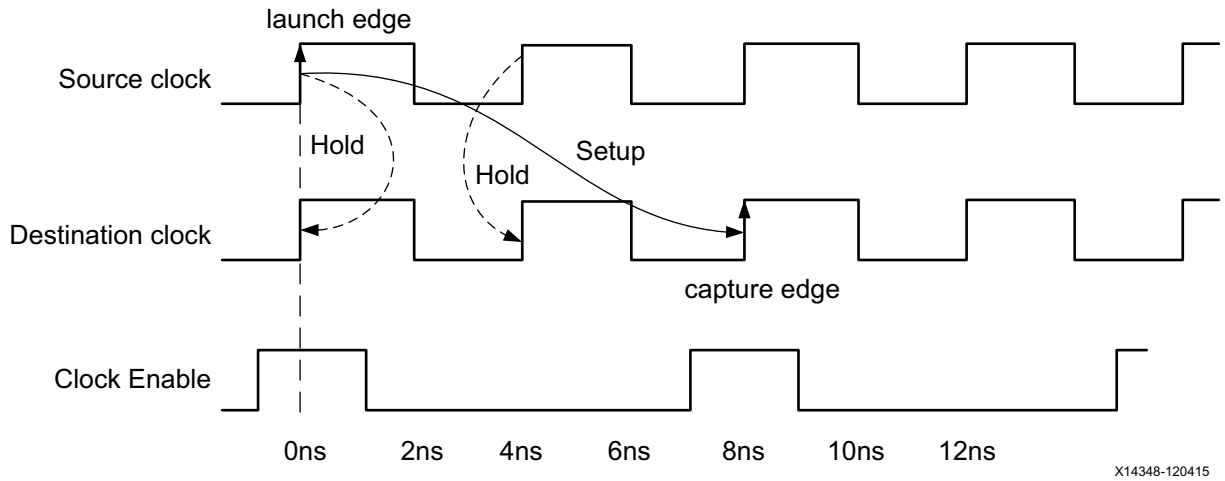


図 5-6: マルチサイクルパス: セットアップとホールドの両方を緩和

この例をまとめると、data0_reg/C と data1_reg/D の間の 2 のマルチサイクルパスを適切に定義するには、次の制約が必要です。

```
set_multicycle_path 2 -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
set_multicycle_path 1 -hold -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

セットアップの乗数が 4 のマルチサイクルパスでは、制約は次のようになります。

```
set_multicycle_path 4 -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
set_multicycle_path 3 -hold -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

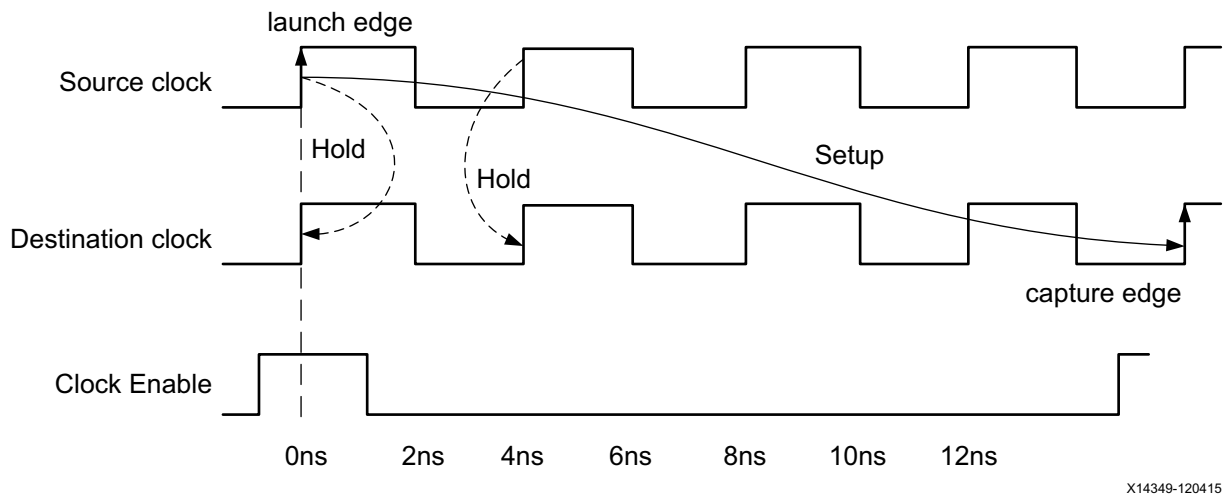


図 5-7: セットアップの乗数が 4 のマルチサイクルパス

セットアップの移動

次に、セットアップを移動した例を示します。

- 「例 1: セットアップの乗数を 5 に設定し、ホールドをそれに応じて移動」
- 「例 2: セットアップの乗数を 5、ホールドの乗数を 4 に設定」

例 1: セットアップの乗数を 5 に設定し、ホールドをそれに応じて移動

セットアップ パスの乗数を 5 に設定するとします。ホールド パスの乗数は指定されていないので、ホールド関係はセットアップのソース エッジおよびデスティネーション エッジから算出されます。

```
set_multicycle_path 5 -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

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 クロックに対して定義されるので、デスティネーション クロックのエッジが先に移動します。セットアップのデスティネーション エッジは、5 クロック周期後になります。ホールドの乗数は指定されていないので、ホールド チェックのデスティネーション クロックのエッジは、セットアップ チェックに使用されるアクティブ エッジの 1 サイクル前に到着するエッジのままになります。

セットアップ関係およびホールド関係の両方に対し、ソース クロックのエッジは変更されません。

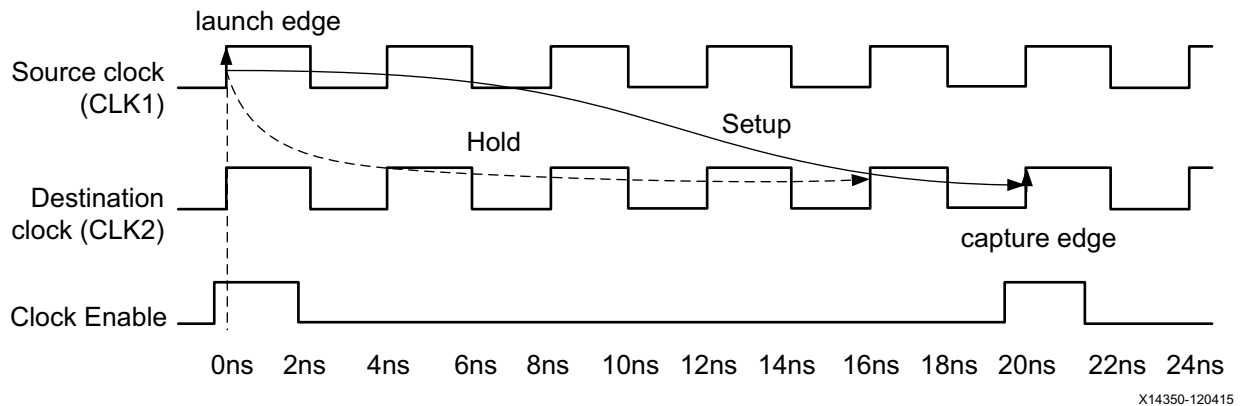


図 5-8: セットアップの乗数を 5 に設定し、ホールドをそれに応じて移動

4 サイクルのホールド要件では、タイミングドリブンのインプリメンテーション ツールで通常、スローおよびファーストの両方のタイミング コーナーでホールド タイミングを満たすため、データパスに大きな遅延を挿入する必要があります。これにより、不要なエリアおよび消費電力が発生します。そのため、可能な場合はホールド要件を緩和することが重要です。

このサンプル デザインでは、クロック イネーブル信号があるので、data0_reg でデータを 4 サイクル間保持しなくてもメタステーブル状態になることはありません。「例 2: セットアップの乗数を 5、ホールドの乗数を 4 に設定」では、ホールド要件の緩和方法を説明します。

例 2: セットアップの乗数を 5、ホールドの乗数を 4 に設定

この例では、次のように定義されています。

- セットアップの乗数を 5
- ホールドの乗数を 4 (5 - 1)

これは、2 つのシーケンシャル セル間で新しいデータが 5 サイクルごとに送信および受信されるということです。

```
set_multicycle_path 5 -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
set_multicycle_path 4 -hold -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

デフォルトでは、セットアップの乗数はデスティネーション クロックに適用され、この例ではデスティネーション エッジが 5 サイクル先に移動します。ホールド チェックは、デフォルトではセットアップ チェックに応じて変更されます。

2 つ目のコマンドでは、ホールドの乗数はソース クロックに適用され、この例ではソース エッジが 4 サイクル先に移動します。

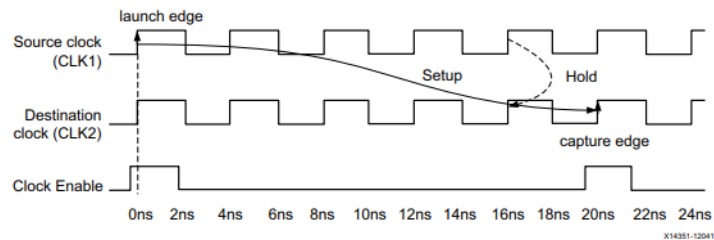


図 5-9: セットアップの乗数を 5、ホールドの乗数を 4 に設定

ソース クロックおよびデスティネーション クロックの両方の波形が同じで、位相も一致しているため、図 5-9 は図 5-10 と同じになります。

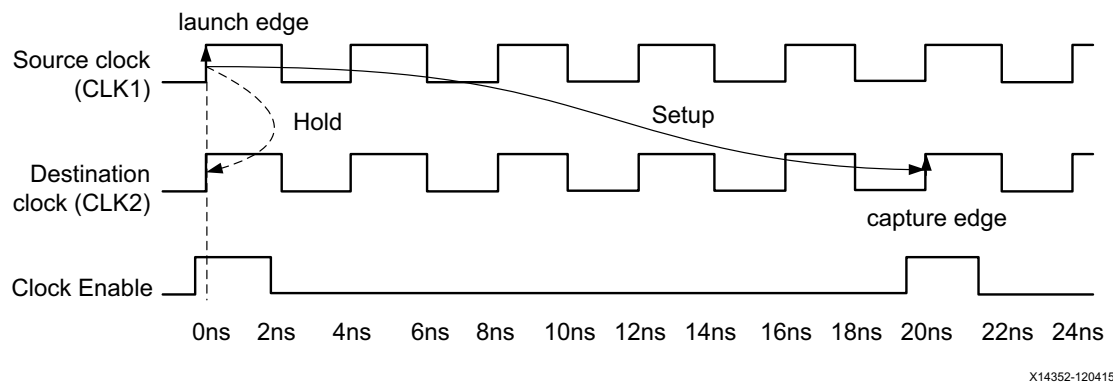


図 5-10: セットアップの乗数を 5、ホールドの乗数を 4 に設定



重要: 通常、同じクロック ドメイン内または同じ波形を持つ 2 つのクロック間では、次に示すように、セットアップの乗数が N の場合、ホールドの乗数は N-1 (最も一般的なケース) に定義します。

```
set_multicycle_path N -setup -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
set_multicycle_path N-1 -hold -from [get_pins data0_reg/C] -to [get_pins data1_reg/D]
```

マルチサイクルパスおよびクロック位相シフト

クロック周期が同じで位相シフトのある 2 つのクロックドメイン間で、タイミング制約を定義する必要がある場合があります。その場合、タイミングエンジンで使用されるデフォルトのセットアップ関係およびホールド関係を理解しておくことが重要です。注意深く調整しないと、2 つのクロック間の位相シフトにより、これらのクロックドメイン間のロジックが過剰に制約される可能性があります。

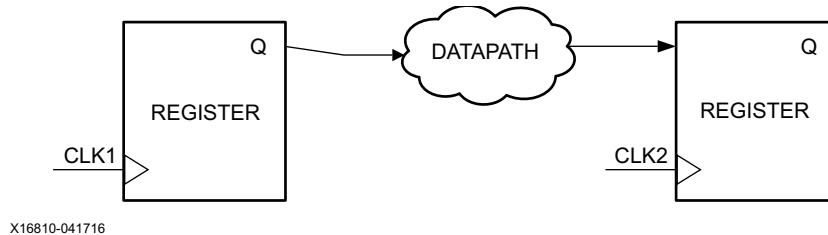


図 5-11: マルチサイクルパスおよびクロック位相シフト

次のような 2 つのクロックがあるとします。

- 2 つのクロック CLK1 と CLK2 の波形は同じ。
- CLK2 は +0.3 ns シフトされている。

タイミングエンジンでは、セットアップ関係を算出するため、両方の波形のすべてのエッジが検出され、最も厳しい制約となるソースクロックとデスティネーションクロックの 2 つのエッジが選択されます。

クロックの位相シフトのため、タイミングエンジンで使用されるセットアップ関係とホールド関係が予測と異なる場合があります。図 5-12 を参照してください。

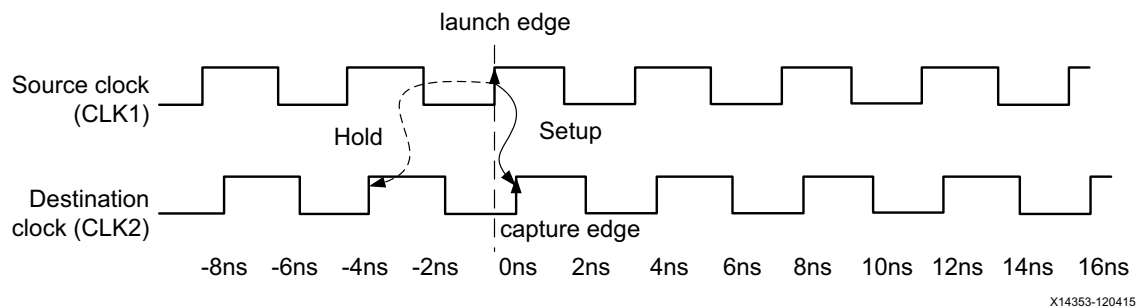


図 5-12: 位相シフトクロックでマルチサイクルパス制約を設定しない場合の例

この例では、位相シフトのためセットアップ制約は 0.3 ns になります。これでは、タイミングクロージャを達成することはほぼ不可能です。一方、ホールドチェックは -3.7 ns であり、これはゆるすぎます。

意図どおりの動作を得るため、セットアップエッジとホールドエッジを調整する必要があります。これには、セットアップの乗数を 2 に設定するマルチサイクルパス制約を追加します。

```
set_multicycle_path 2 -setup -from [get_clocks CLK1] -to [get_clocks CLK2]
```

これにより、セットアップ要件のデスティネーションエッジが 1 サイクル先に移動します。ホールドのデフォルトのエッジは、セットアップ要件から導出されたものになるので、指定する必要はありません。

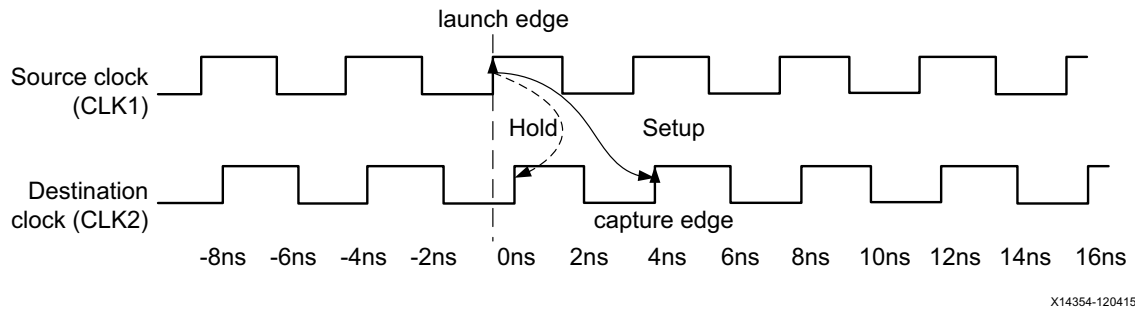


図 5-13: 正の位相シフトの例: セットアップの乗数を 2 (-end) に設定し、ホールドをそれに応じて移動

2 つのクロック ドメイン間の位相シフトが負の場合 (図 5-14)、セットアップ チェックおよびホールド チェックに使用されるソース エッジとデスティネーション エッジは、1 つのクロック ドメインで位相シフトのない場合と似ています。

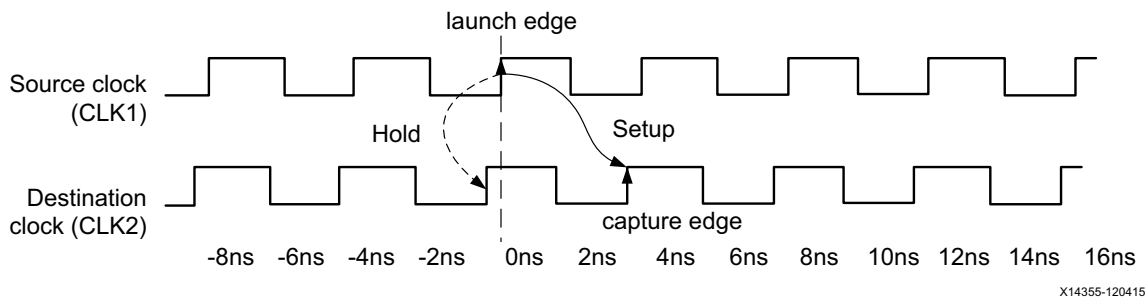


図 5-14: 負の位相シフトの例

負の位相シフトでは、位相シフトの影響を相殺するためマルチサイクルパス制約を設定する必要は通常ありませんが、位相シフトが大きい場合は、セットアップおよびホールド要件を現実的なものにするためにクロックのソース エッジまたはデスティネーション エッジを調整する必要がある場合もあります。

低速クロックから高速クロックのマルチサイクル

ソース クロック CLK1 が低速で、デスティネーション クロック CLK2 が高速である場合を考えます。図 5-15 を参照してください。

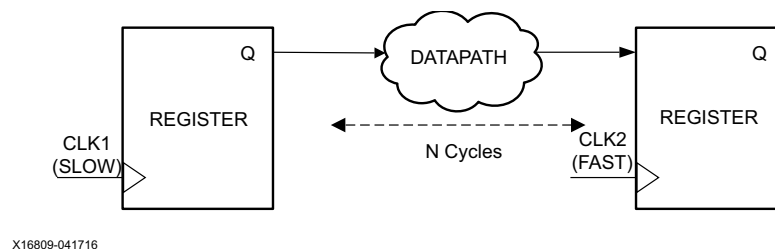


図 5-15: 低速クロックから高速クロックのマルチサイクル

次のような 2 つのクロックがあるとして。

- CLK2 の周波数は CLK1 の周波数の 3 倍
- 受信レジスタにクロック イネーブル信号があり、2 つのクロックの間にマルチサイクル パス制約を設定可能。
図 5-16 を参照してください。

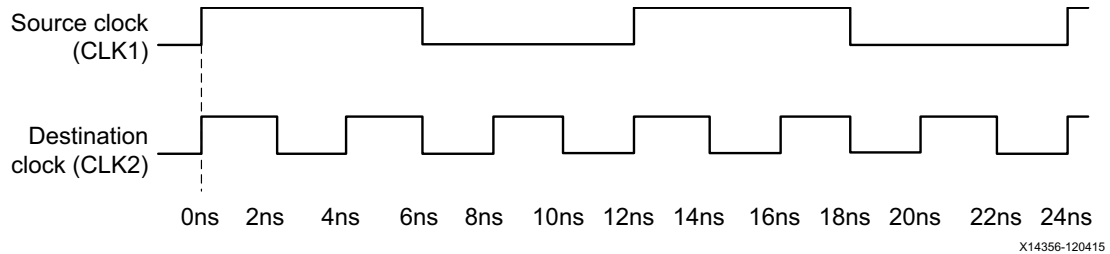


図 5-16: 低速クロックから高速クロックのマルチサイクル

図 5-17 に、マルチサイクルが設定されていない場合に STA ツールで使用するセットアップ関係とホールド関係を示します。

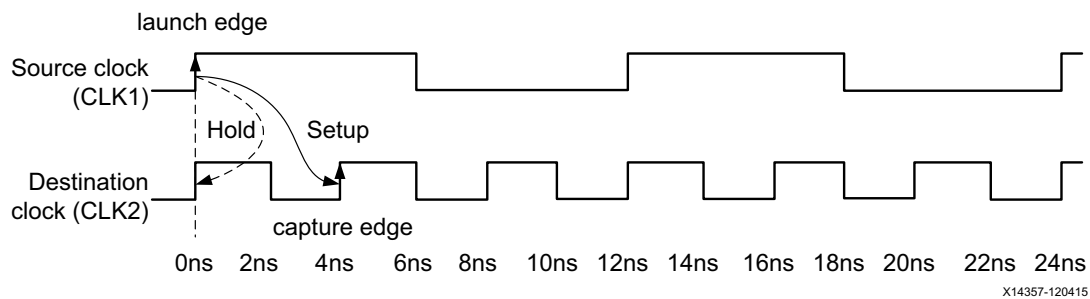


図 5-17: デフォルトのセットアップおよびホールド関係

例 1: セットアップの乗数を 3 に設定し、ホールドをそれに応じて移動

次のようなセットアップの乗数が 3 のマルチサイクルパス制約を定義するとします。

```
set_multicycle_path 3 -setup -from [get_clocks CLK1] -to [get_clocks CLK2]
```

セットアップに対してマルチサイクルパス制約を設定すると、セットアップチェックに使用されるデスティネーションクロックのエッジが 2 (3 - 1) サイクル先に移動します。ホールドパスの乗数は指定されていないので、ホールド関係はセットアップのソースエッジおよびデスティネーションエッジから導出されます。ソースクロックのアクティブエッジはマルチサイクルパス制約により変更されません。

図 5-18 に、マルチサイクルが設定された後のセットアップ関係とホールド関係を示します。

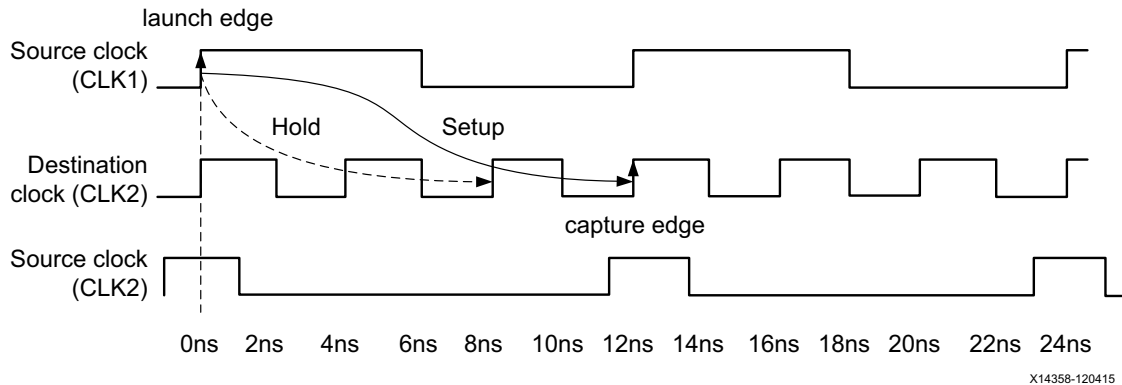


図 5-18: セットアップの乗数を 3 にし、ホールドをそれに応じて移動

このパスが機能するためには、送信レジスタにデータを CLK2 の 1 サイクル間保持する必要はありません。そのようにすると不要なロジックを追加することになり、エリアおよび消費電力が増加します。

受信レジスタにはクロック ネーブル信号があるので、メタステーブル状態になることなくホールド要件を緩和できます。

例 2: セットアップの乗数を 3、ホールドの乗数を 2 (-end) に設定

前の例でホールド要件を緩和するには、ホールド関係のデスティネーション クロック エッジを 2 クロック サイクル前に移動する必要があります。これには、`set_multicycle_path -hold -end` コマンドを `-end` オプションを指定して使用します。

```
set_multicycle_path 3 -setup -from [get_clocks CLK1] -to [get_clocks CLK2]
set_multicycle_path 2 -hold -end -from [get_clocks CLK1] -to [get_clocks CLK2]
```



ヒント: `set_multicycle_path -hold` コマンドで `-end` オプションを指定しない場合、ソース クロック エッジが先に移動します。これでは意図したホールド要件が得られません。

「例 1: セットアップの乗数を 3 に設定し、ホールドをそれに応じて移動」にあるように、セットアップの乗数は、セットアップ チェックに使用されるデスティネーション クロックのエッジを 2 (3 - 1) サイクル先に移動します。

図 5-19 に、2 つのマルチサイクルパス制約が設定された後のセットアップ関係とホールド関係を示します。

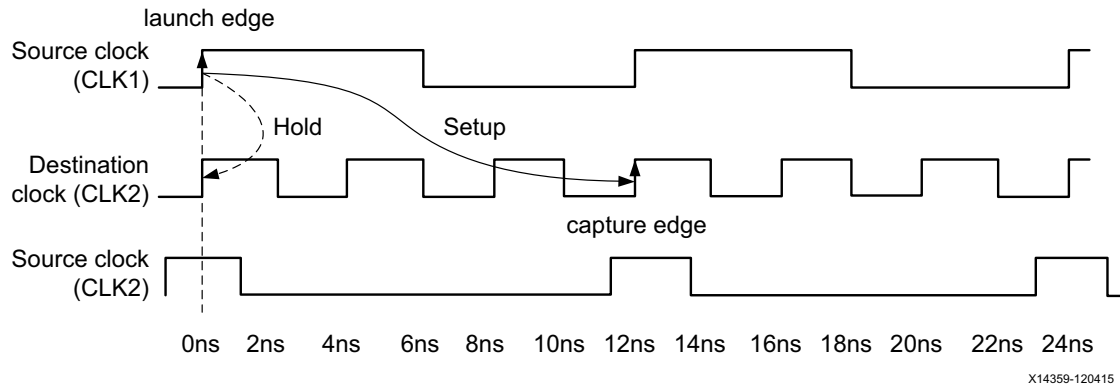


図 5-19: セットアップの乗数を 3、ホールドの乗数を 2 (-end) に設定



重要: 低速クロックから高速クロックへのクロック乗せ換えでは、次のコード例に示すように、セットアップの乗数を N に定義した場合、デスティネーションクロック (-end) に対してホールドの乗数を N-1 に定義します (最も一般的なケース)。

```
set_multicycle_path N -setup -from [get_clocks CLK1] -to [get_clocks CLK2]
set_multicycle_path N-1 -hold -end -from [get_clocks CLK1] -to [get_clocks CLK2]
```

高速クロックから低速クロックのマルチサイクル

ソースクロック CLK1 が高速で、デスティネーションクロック CLK2 が低速である場合を考えます。図 5-20 を参照してください。

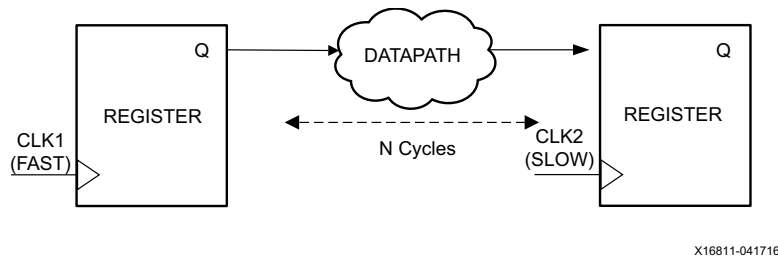


図 5-20: 高速クロックから低速クロックのマルチサイクル

この例では、ソース クロック CLK1 が高速クロックで、デスティネーション クロック CLK2 が低速クロックです。CLK1 の周波数が CLK2 の周波数の 3 倍であるとして、図 5-21 を参照してください。

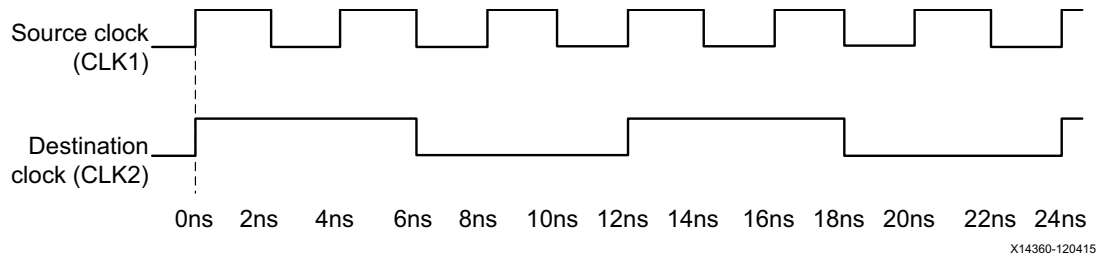


図 5-21: 高速クロックから低速クロックのマルチサイクル

図 5-22 に、マルチサイクルが設定されていない場合に STA ツールで使用するセットアップ関係とホールド関係を示します。

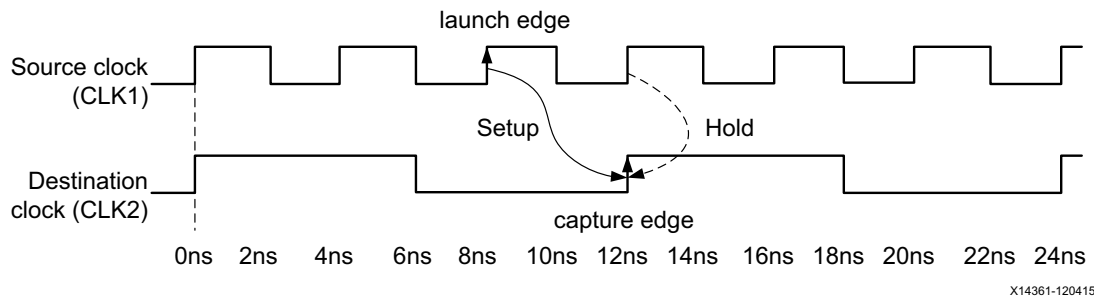


図 5-22: デフォルトのセットアップおよびホールド関係

例: セットアップの乗数を 3 (-start)、ホールドの乗数を 2 に設定

この例では、次が想定されます。

- セットアップの乗数 3 がソース クロックに対して定義されます (-start)。
- ホールドの乗数 1 が定義されます。

例:

```
set_multicycle_path 3 -setup -start -from [get_clocks CLK1] -to [get_clocks CLK2]
set_multicycle_path 2 -hold -from [get_clocks CLK1] -to [get_clocks CLK2]
```

ソース クロック (-start) に対してセットアップの乗数を設定すると、セットアップチェックに使用されるソース クロックのエッジが 2 (3 - 1) サイクル前に移動します。ただし、ホールドの乗数はソース クロックに対して設定されるので (デフォルトの -start オプションと -hold を使用)、ホールド関係に使用されるソース クロックのエッジは 2 サイクル先に移動します。

セットアップ チェックおよびホールド チェックの両方で、デスティネーション クロックのエッジは変更されません。次の図を参照してください。

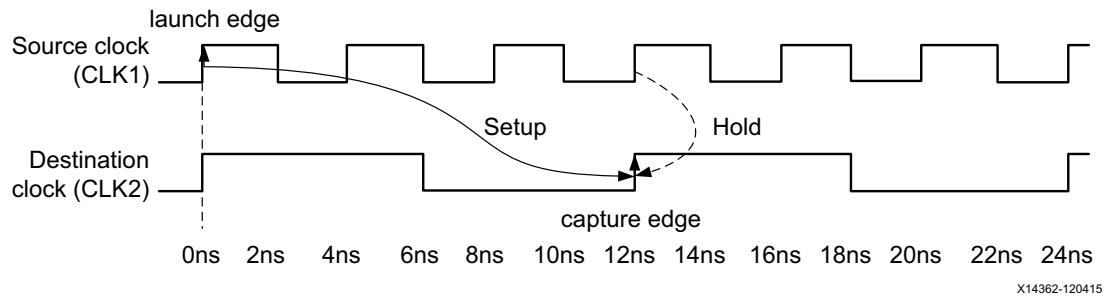


図 5-23: セットアップの乗数を 3 (-start)、ホールドの乗数を 2 に設定



重要: 高速クロックから低速クロックへのクロック乗せ換えでは、セットアップの乗数をソース クロック (-start) に対して N に設定し、ホールドの乗数を N-1 に設定します (最も一般的なケース)。次に例を示します。

```
set_multicycle_path N -setup -start -from [get_clocks CLK1] -to [get_clocks CLK2]
set_multicycle_path N-1 -hold -from [get_clocks CLK1] -to [get_clocks CLK2]
```

表 5-3 に、上記の例をまとめます。

表 5-3: セットアップ N を使用してマルチサイクルパスを定義する方法

状況	マルチサイクルパス制約
同じクロック ドメインまたは同じ周期で位相シフトのない同期クロック ドメイン間	<pre>set_multicycle_path N -setup -from CLK1 -to CLK2 set_multicycle_path N-1 -hold -from CLK1 -to CLK2</pre>
同期した低速クロック ドメインから高速クロック ドメイン	<pre>set_multicycle_path N -setup -from CLK1 -to CLK2 set_multicycle_path N-1 -hold -end -from CLK1 -to CLK2</pre>
同期した高速クロック ドメインから低速クロック ドメイン	<pre>set_multicycle_path N -setup -start -from CLK1 -to CLK2 set_multicycle_path N-1 -hold -from CLK1 -to CLK2</pre>

注記: コマンドをシンプルにするため、表 5-3 では get_clocks コマンドは省略されています。

フォルス パス

フォルス パスは、デザインには存在するが、機能はなく、タイミング解析する必要のないパスです。そのため、フォルス パスはタイミング解析で無視する必要があります。

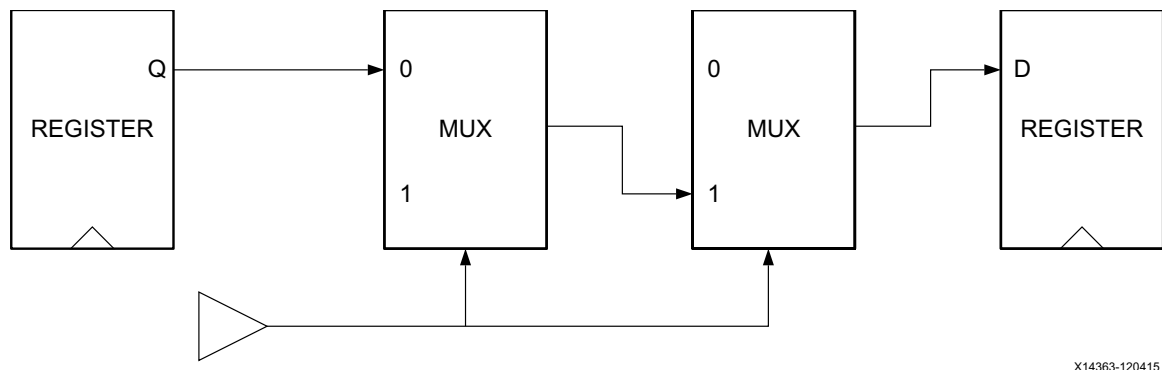


ビデオ: フォルス パスなどの高度なタイミング例外については、[Vivado Design Suite QuickTake ビデオ: 高度なタイミング例外 - フォルス パス、最小/最大遅延、ケース解析](#)を参照してください。

次に、フォルス パスの例を示します。

- ダブル シンクロナイザー ロジックが追加されたクロック ドメインをまたがるパス
- 電源投入時に一度だけ書き込まれるレジスタ
- リセットまたはテスト ロジック
- 非同期分散 RAM の書き込みクロックと非同期読み出しクロックの間のパスを無視 (該当する場合)

図 5-24 に、機能しないパスの例を示します。2 つのマルチプレクサーはどちらも同じセレクト信号で駆動されるので、Q から D へのパスは存在せず、フォルス パスとして定義する必要があります。



X14363-120415

図 5-24: 機能しないパスの例



ヒント: 同期パスのタイミング要件を緩和することだけが目的で、パスをタイミング解析、検証、最適化する必要がある場合は、フォルス パス制約ではなくマルチサイクルパス制約を使用してください。

タイミング解析からフォルス パスを除外するには、次のような理由があります。

- **実行時間の短縮**

タイミング解析からフォルス パスを除外すると、機能しないパスのタイミングをツールで解析したり最適化する必要はありません。機能しないパスをタイミング エンジンおよび最適化エンジンで処理する必要があると、実行時間が大幅に増加します。

- **QoR (結果の品質) の向上**

フォルス パスを除外すると、QoR が大幅に向上する可能性があります。合成、配置、最適化されたデザインの質は、ツールが解決しようとするタイミング問題に大きく影響されます。

機能しないパスにタイミング違反がある場合、実際に機能するパスではなく、機能しないパスの問題を修正するために時間が費やされる可能性があります。ロジックが複製されるなど、デザインのサイズが不必要に大きくなるだけでなく、機能しないパスの違反が大きいため本物の違反が修正されないこともあります。ベストな結果は、現実的な制約を設定しないと達成できません。

フォールス パスは、XDC コマンドの `set_false_path` を使用してツール内で定義されます。

```
set_false_path [-setup] [-hold] [-from <node_list>] [-to <node_list>] \
  [-through <node_list>]
```

これ以外にも、パスをより詳細に指定する次のコマンド オプションを使用できます。サポートされるすべてのコマンド ライン オプションの詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 10] を参照してください。

- `-from` オプションでは、有効な始点のノード リストを指定する必要があります。有効な始点は、クロック オブジェクト、順次エレメントのクロック ピン、入力 (または入出力) プライマリ ポートです。複数のエレメントを指定できます。
- `-to` オプションでは、有効な終点のノード リストを指定する必要があります。有効な終点は、クロック オブジェクト、出力 (または入力) プライマリ ポート、順次エレメントの入力データ ピンです。複数のエレメントを指定できます。
- `-through` オプションでは、有効なピン、ポート、またはネットのノード リストを指定する必要があります。複数のエレメントを指定できます。



注意: `-from` および `-to` オプションを使用せずに `-through` オプションを使用する場合は、指定したピンまたはポートを追加するすべてのパスがタイミング解析から除外されるので、注意が必要です。IP またはサブブロック用にタイミング制約を定義し、別のコンテキストまたは大型のプロジェクトで使用する場合は、特に注意してください。`-through` のみを使用すると、予測以上の数のパスが削除される可能性があります。

`-through` オプションの順序も重要です。次に例を示します。

次の 2 つは、異なる制約を設定します。

```
set_false_path -through cell1/pin1 -through cell2/pin2
set_false_path -through cell2/pin2 -through cell1/pin1
```

次の例では、`reset` ポートからすべてのレジスタへのタイミング パスを除外しています。

```
set_false_path -from [get_port reset] -to [all_registers]
```

次の例では、2 つの非同期クロック ドメイン間 (クロック `CLKA` から `CLKB`) のタイミング パスを除外しています。

```
set_false_path -from [get_clocks CLKA] -to [get_clocks CLKB]
```

上記の例では、クロック `CLKA` からクロック `CLKB` のパスは除外されますが、クロック `CLKB` からクロック `CLKA` へのパスは除外されません。2 つのクロック ドメイン間の両方向のパスをすべてディスエーブルにするには、次のように 2 つの `set_false_path` コマンドが必要です。

```
set_false_path -from [get_clocks CLKA] -to [get_clocks CLKB]
set_false_path -from [get_clocks CLKB] -to [get_clocks CLKA]
```



重要: 上記 2 つの `set_false_path` 例で意図した機能が実行されますが、2 つ以上のクロック ドメインが非同期で、それらのクロック ドメイン間のパスをいずれかの方向でディスエーブルにする必要がある場合は、`set_clock_groups` コマンドを使用することを推奨します。
`set_clock_groups -group CLKA -group CLKB`

図 5-24 の機能していないパスの例では、フォルス パスが `-from` や `-to` オプションではなく、`-through` オプションを使用して設定されています。図 5-25 を参照してください。

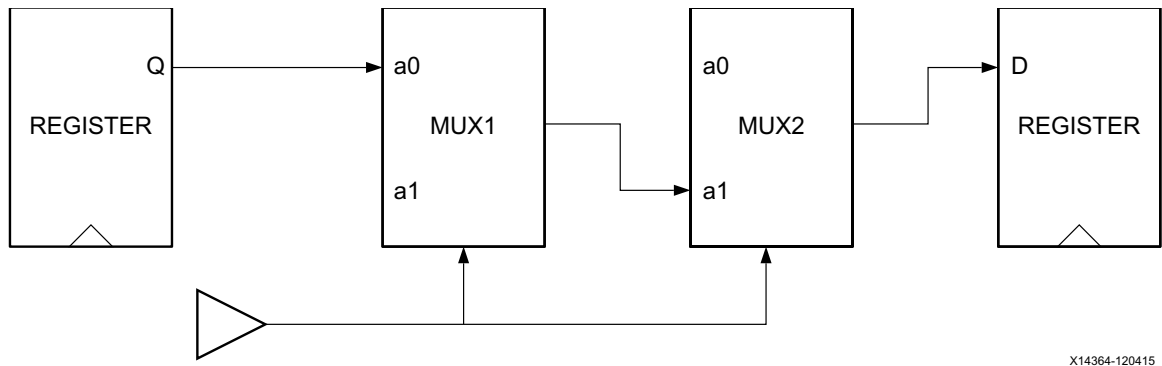


図 5-25: 機能しないパスの例

これにより、始点および終点の特定のパターンを見つける必要なく、上図に示すパスを通過するすべてのパスを選択できます。

```
set_false_path -through [get_pins MUX1/a0] -through [get_pins MUX2/a1]
```

注記: `-through` オプションの順序も重要です。上記の例の順序では、フォルス パスはまず MUX1/a0 ピンを通過してから MUX2/a1 ピンを通過します。

別の一般的な例として、非同期デュアルポート分散 RAM を使用する場合があります。書き込みはクロック RAM に同期していますが、読み出しはデザインで許容される場合は非同期にすることもできます。この場合、書き込みクロックと読み出しクロックの間のタイミング パスをフォルス パスとして設定しても問題ありません。

この設定には次の 2 つの方法があります。

- RAM の前にある書き込みレジスタから読み出しクロックを受信する RAM の後にあるレジスタまでのパスを、フォルス パスとして定義します。

```
set_false_path -from [get_cells <write_registers>] -to [get_cells <read_registers>]
```

Vivado Design Suite のサンプルプロジェクト wavegen (HDL):

```
set_false_path -from [get_cells -hier -filter {NAME =~ *gntv_or_sync_fifo.gl0.wr*reg[*]}] -to [get_cells -hier -filter {NAME =~ *gntv_or_sync_fifo.mem*gpr1.dout_i_reg[*]}]
```

- RAM の WE ピンから開始するパスをフォルス パスとして定義します。

```
set_false_path -from [get_cells -hier -filter {REF_NAME =~ RAM* && IS_SEQUENTIAL && NAME =~ <PATTERN_FOR_DISTRIBUTED_RAMS>}]
```

Vivado Design Suite のサンプルプロジェクト wavegen (HDL):

```
set_false_path -from [get_cells -hier -filter {REF_NAME =~ RAM* && IS_SEQUENTIAL && NAME =~ *char_fifo*}]
```

図 5-26 に、WAVE (HDL) サンプルプロジェクトで分散 RAM がどのように駆動されるかを示します。

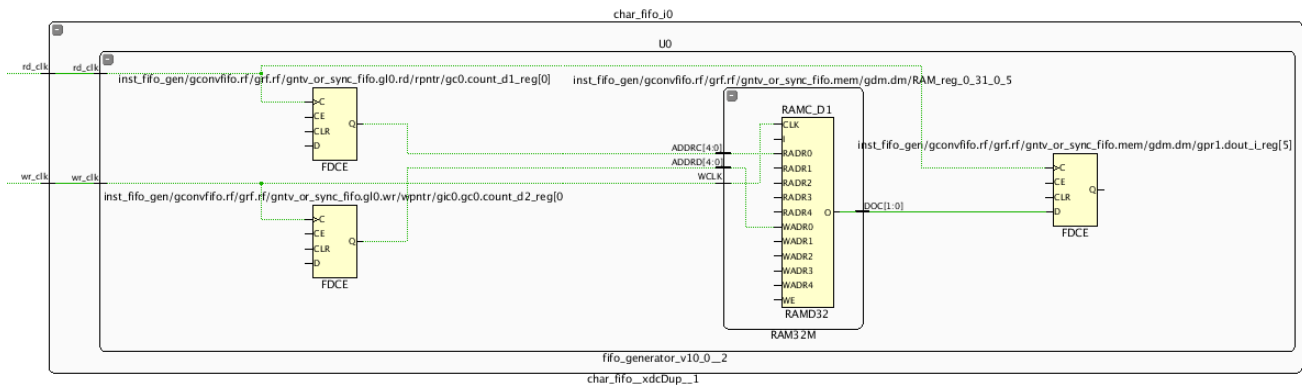


図 5-26: WAVE サンプルプロジェクトで駆動される分散 RAM

最小/最大遅延

パスの最大遅延または最小遅延を変更できます。

- パスのデフォルトのセットアップ (リカバリ) 要件を変更するには最大遅延制約を使用します。
- パスのデフォルトのホールド (リムーバル) 要件を変更するには最小遅延制約を使用します。



ビデオ: 最小/最大遅延などの高度なタイミング例外については、[Vivado Design Suite QuickTake ビデオ: 高度なタイミング例外 - フォルス パス、最小/最大遅延、ケース解析](#)を参照してください。

最大遅延制約および最小遅延制約の設定

最大遅延制約および最小遅延制約は、2 つの異なる XDC コマンドで設定します。どちらのコマンドにも、同様のオプションがあります。

最大遅延制約の構文

```
set_max_delay <delay> [-datapath_only] [-from <node_list>]
                        [-to <node_list>] [-through <node_list>]
```

最小遅延制約の構文

```
set_min_delay <delay> [-from <node_list>]
                      [-to <node_list>] [-through <node_list>]
```

これ以外にも、パスをより詳細に指定するコマンド オプションがあります。サポートされるすべてのコマンド ライン オプションの詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [\[参照 10\]](#)を参照してください。

-from オプションのノード リスト

- -from オプションでは、有効な始点のノード リストを指定する必要があります。有効な始点は、クロック、入力または入出力ポート、レジスタや RAM などの順次エレメントのクロック ピンです。有効でない始点のノードを使用すると、パスが分割されます。パスの分割については、次のセクションで説明します。
- 複数のエレメントを指定できます。

-to オプションのノード リスト

- -to オプションでは、有効な終点のノード リストを指定する必要があります。有効な終点は、クロック、出力または入出力ポート、シーケンシャル セルのデータ ピンです。
- 有効でない終点のノードを使用すると、パスが分割されます。詳細は、[119 ページの「パス分割」](#)を参照してください。
- 複数のエレメントを指定できます。

-through オプションのノード リスト

- -through オプションでは、有効なピン、ポート、またはネットのノード リストを指定する必要があります。
- 複数のエレメントを指定できます。

デフォルトでは、スラックの算出にクロック スキューが含まれます。

スラックの計算からクロック スキューを除外するには、-datapath_only オプションを使用できます。-datapath_only オプションは、set_max_delay コマンドでのみサポートされ、-from オプションと共に使用する必要があります。

[表 5-4](#) に、set_max_delay 制約に -datapath_only を使用する際の影響をまとめます。

-datapath_only オプションを使用した場合と使用しない場合の set_max_delay のパス遅延算出の共通点は、次のとおりです。

- 入力遅延は、パスが入力ポートで開始し、set_input_delay がそのポートに指定されている場合に、パス遅延算出に含まれます。
- 出力遅延は、パスが出力ポートで終了し、set_output_delay がそのポートに指定されている場合に、パス遅延算出に含まれます。
- データ ピンのセットアップ タイムは、パスが順次エレメントのデータ ピンで終了する場合に、パス遅延算出に含まれます。

表 5-4: -datapath_only を使用した場合としない場合の最大遅延制約の違い

	set_max_delay	set_max_delay -datapath_only
パス遅延の算出	制約が順次エレメントのクロック ピンで開始し、順次エレメントのデータ ピンで終了する場合にスキューが含まれる。	スキューは含まれない。
ホールド要件	変化なし	フォルス パス
-from オプション	オプション	必須

パスに最大遅延制約および最小遅延制約を設定した場合の影響

-datapath_only オプションが使用されていない場合、パスに最大遅延制約を設定しても、そのパスの最小要件は変更されません。そのパスのホールド (リムーバル) チェックはデフォルトのままです。

注記: set_max_delay に -datapath_only オプションを使用すると、そのパスでホールド要件が無視されます (set_false_path -hold 内部制約がいくつか生成されます)。

同様に、パスに最小遅延制約を設定しても、デフォルトのセットアップ (リカバリ) チェックは変更されません。

たとえば、あるパスに最大遅延要件だけがある場合は、そのパスには set_max_delay と set_false_path コマンドを組み合わせて制約を設定できます。次に例を示します。

```
set_max_delay 5 -from [get_pins FD1/C] -to [get_pins FD2/D]  
set_false_path -hold -from [get_pins FD1/C] -to [get_pins FD2/D]
```

上記の例では、始点が FD1/C で終点が FD2/D のパスに、5 ns のセットアップ要件が設定されます。set_false_path コマンドがあるので、最小要件はありません。

入力または出力ロジックの制約

set_max_delay コマンドおよび set_min_delay コマンドは、通常、入力または出力ロジックに制約を設定するためには使用されません。入力ポートと 1 段目のレジスタとの間の入力ロジックには、一般的に set_input_delay コマンドを使用して制約を設定します。このコマンドには、クロックと入力ポートを関連付けるオプションがあります。

同じ理由で、最終段のレジスタと出力ポートの間の出力ロジックには、一般的に、set_output_delay コマンドを使用して制約を設定します。ただし、プライマリ入力ポートとプライマリ出力ポートの間の組み合わせパス (in-to-out I/O パス) を制約する場合は、通常 set_max_delay コマンドおよび set_min_delay コマンドを使用します。

非同期信号の制約

set_max_delay コマンドは、クロック関係がないが、最大遅延が必要な非同期信号を制約するためにも使用できます。

たとえば、2 つの非同期クロックドメインの間のタイミング パスは、set_clock_groups コマンド (推奨)、または set_false_path コマンド (推奨されない) を使用して、ディスエーブルにできます。これは、クロックドメイン間がダブルレジスタシンクロナイザーや FIFO などを使用して適切に設計されていることを想定しています。ただし、2 つのクロックドメイン間のパス遅延が不必要に大きくないことを確認する必要があります。

複数ビットの CDC の場合、ビット間のスキューが特定の要件内に収まるようにする必要があります。スキューはパス スキュー制約 (set_bus_skew) で設定することはできますが、2 つのクロックドメイン間のパス遅延が不必要に大きくないことを確認する必要があります。これには、関連するパスのソースの XDC ファイル内で set_false_path または set_clock_groups 制約を set_max_delay -datapath_only に置換します。CDC パスの制約の詳細は、第 6 章「CDC 制約」を参照してください。

注記: パスには最大遅延でタイミング制約が付くので、フォールス パス制約と最大遅延制約でランタイムが異なるようになります。

2つのクロックドメイン間のいくつかのパス、またはすべてのパスに対し、最大遅延を設定する必要がある場合は、`set_max_delay -datapath_only` コマンドを使用してください。この場合、`set_max_delay` よりも `set_clock_groups` が優先されるため、2つのクロックドメインを非同期として定義するのに `set_clock_groups` は使用できません。ほかのクロック乗せ換えパスを `set_false_path` と `set_max_delay` 制約を組み合わせる必要があります。

次に例を示します。

```
set_max_delay <delay> -datapath_only -from <startpoints_source_clock_domain> \
-to <endpoints_destination_clock_domain>
```

パス分割

`set_max_delay` および `set_min_delay` コマンドでは、ほかの XDC 制約とは異なり、`-from` および `-to` オプションを使用する場合に、それぞれに無効の始点および終点のリストを使用できます。

無効な始点を指定した場合、そのノードが有効な始点となるようにするため、タイミングはそのノードを介して伝搬されなくなります。

次の例では、有効な始点は FD1/C のみです。

```
set_max_delay 5 -from [get_pins FD1/C]
```

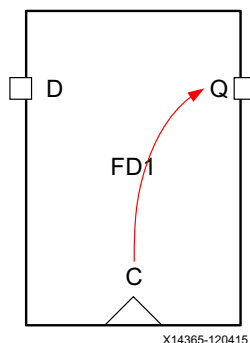


図 5-27: 元のタイミング アーク

FD1/Q に制約が適用されると、Q ピンを有効な始点にするため、タイミングはアーク c->Q を介して伝搬されなくなります。

```
set_max_delay 5 -from [get_pins FD1/Q]
```

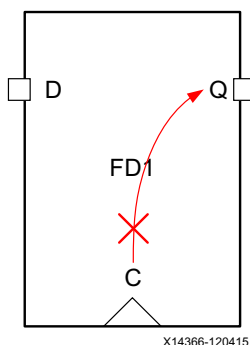
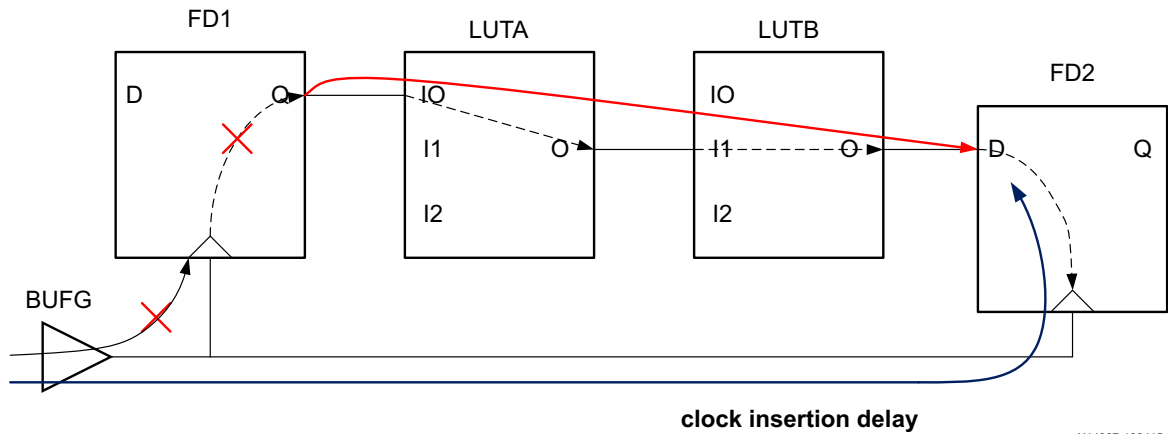


図 5-28: パス分割後タイミングは伝搬されない

有効な始点を作成するためにタイミングの伝搬を停止するプロセスは、パス分割と呼ばれます。パス分割は、最大および最小遅延解析の両方に影響します。パス分割は、それらのノード (FD1/C および FD1/Q) を介するタイミング制約にも影響します。

注記: パス分割により、FD1/Q からのパスのソース クロックにクロック挿入遅延は使用されません。そのため、終点のクロック スキューは考慮されるので、スキューが大きくなる可能性があります。図 5-29 を参照してください。



X14367-120415

図 5-29: パス分割の結果スキューが大きくなる



注意: パス分割により、予期しない結果を招く可能性があります。パス分割を避けるか、十分に注意して使用してください。

パスが分割された場合、パスにデフォルトのホールド要件はありません。-datapath_only オプションが指定されていない場合は、必要であれば set_min_delay コマンドを使用してパスにホールド要件を設定します。

こうしたリスクがあるため、パス分割が発生するとクリティカル警告メッセージが表示されます。

クロック スキューを考慮しないようにするため出力 FD1/Q を始点とする場合は、-datapath_only オプションを使用することをお勧めします。次に例を示します。

```
set_max_delay 5 -from [get_pins FD1/C] -datapath_only
```

同様に、有効でない終点を指定した場合、そのノードが有効な終点となるように、タイミングはそのノードの後には伝搬されなくなります。

次の例では、無効な終点である LUT A/O に最大遅延が指定されています。

```
set_max_delay 5 -from [get_pins LUT A/O]
```

これを次の図に示します。

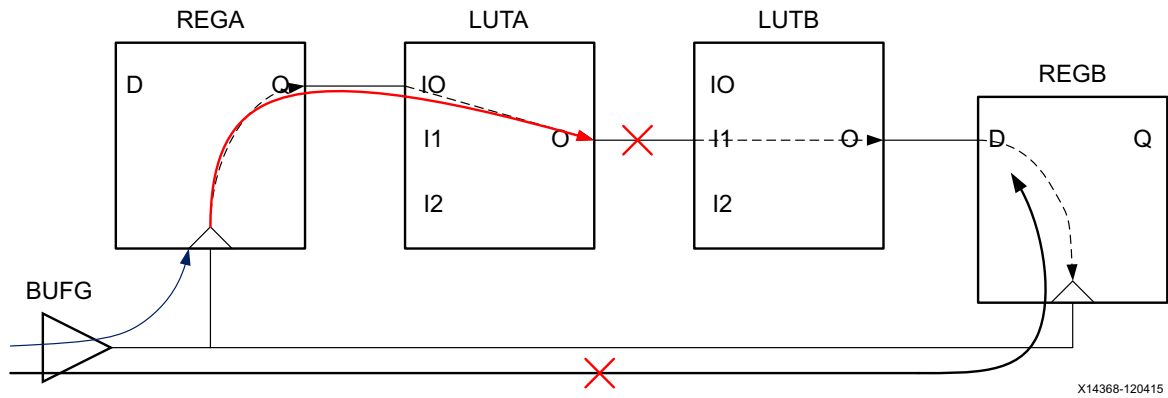


図 5-30: 有効でない終点が指定された場合のパス分割

LUTA/O を有効な終点にするため、タイミングは LUTA/O の後には伝搬されなくなります。この結果、LUTA/O を通過するすべてのタイミングパスのセットアップおよびホールドが影響を受けます。REGA/C から LUTA/O までのパスに対しては、ソースクロックの挿入遅延のみが考慮されるので、スキューが大きくなる可能性があります。

パス分割によりタイミングがタイミングアークを介して伝搬されなくなるので、予期しない結果を招く可能性があります。これらのノードを通過するすべてのタイミングパスが影響を受けます。

次の例では、LUTA/O と REGB/D との間に最大遅延が設定されています。

```
set_max_delay 6 -from [get_pins LUTA/O] -to [get_pins REGB/D]
```

これを次の図に示します。

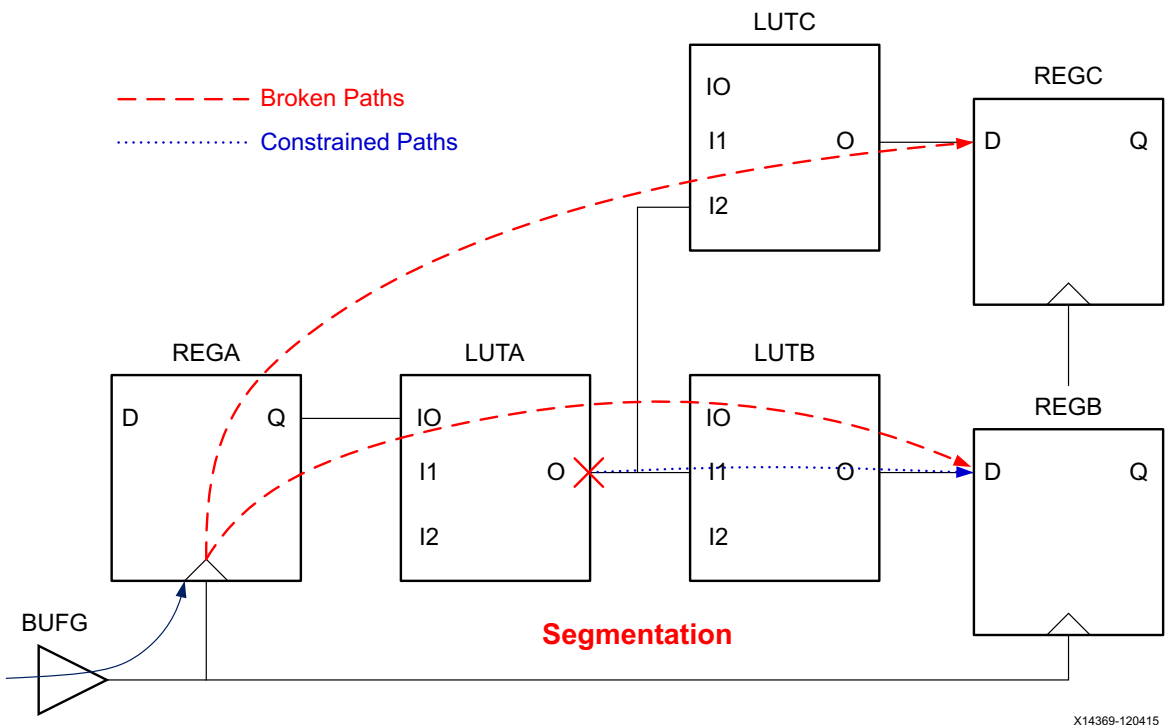


図 5-31: パス分割により複数のパスが分割される

LUTA/O は有効な始点ではないため、パスが分割され、LUTA/I* および LUTA/OLUTA/O からのタイミング アークが切断されます。set_max_delay 制約は LUTA/O と REGB/D の間のみ設定されていますが、REGA/C と REGC/D との間のパスなど、ほかのパスも分割されます。

パス分割とタイミング例外

パス分割によりタイミング例外間の優先順位が変更されると思われるかもしれませんが、実際にはそうではありません。

set_clock_groups 制約が set_max_delay 制約より優先されるかどうかは、ケース バイ ケースです。次の 2 つの状況を考えます。

状況 1

```
set_max_delay <ns> -datapath_only -from <instance> -to <instance>
```

この場合、-from/-to でインスタンス名が指定されます。インスタンスが指定されると、Vivado により常に有効な始点を選択されるので、set_clock_groups -asynchronous により常に set_max_delay は無効になります。

状況 2

```
set_max_delay <ns> -datapath_only -from <pin> -to <pin | instance>
```

この場合、-from に指定されたピン名によりパスが分割されると、set_max_delay 制約が set_clock_groups -asynchronous により無効になることはありません。これは、パス分割により、ピン名のパス始点が最初のクロックドメインから開始すると考慮されなくなり、このパスに set_clock_groups 制約が適用されず、set_max_delay 制約が適用されるからです。

ケース解析

デザインによっては、一部の信号が特定のモードで定数になります。たとえば、動作モードでは、テスト信号はトグルしないので、そのアクティブ レベルによって VDD または VSS に接続されます。デザインに電源が投入された後にトグルしない信号もこの対象になります。また、多くのデザインには複数の動作モードがあり、ある動作モードではアクティブな信号が別のモードでは非アクティブである場合があります。

解析空間、実行時間、およびメモリ消費量を削減するため、信号が定数値となることを指定しておくことが重要です。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。

信号が非アクティブであることを宣言するには、set_case_analysis コマンドを使用します。このコマンドは、ピンまたはポートに適用できます。

注記: ピンにケース解析を設定すると、そのピンに関連するタイミング アークがすべてディスエーブルになり、通過するパスはレポートされなくなります。



ビデオ: set_case_analysis などの高度なタイミング例外については、[Vivado Design Suite QuickTake ビデオ: 高度なタイミング例外 - フォルス パス、最小/最大遅延、ケース解析](#)を参照してください。

set_case_analysis コマンドの構文は、次のとおりです。

```
set_case_analysis <value> <pins or ports objects>
```

<value> は、次のいずれかになります。

0, 1, zero, one, rise, rising, fall, or falling

rise、rising、fall、または 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。

ケース値は、ポート、最下位セルのピン、階層モジュールのピンに設定できます。

次の例では、マルチプレクサー 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、セレクト ピン S に定数値が設定されているため、clk_2 のみが出力ピンを介して伝搬されます。

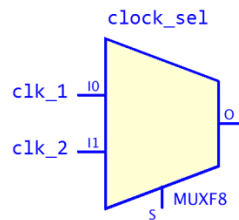


図 5-32: クロック例

```
create_clock -name clk_1 -period 10.0 [get_pins clock_sel/I0]
create_clock -name clk_2 -period 15.0 [get_pins clock_sel/I1]
set_case_analysis 1 [get_pins clock_sel/S]
```

次の例では、BUFG_GT のダイナミック クロック分周ピン DIV[2:0] は、VCC/GND に接続されているのではなく、ロジックにより駆動されます。

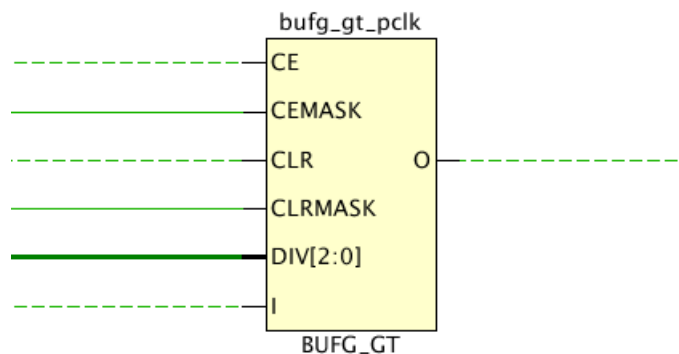


図 5-33: BUFG_GT/DIV の例

この場合、ツールで出力クロック (1 で分周) のワースト ケースが想定され、入力クロックがバッファ出力に伝搬されます。クロック分周 1 が使用されない場合、このワースト ケースの状況により見積もりが不必要に悪くなり、デザインの制約が厳しすぎるものになる可能性があります。DIV[2:0] バスに set_case_analysis 制約を設定すると、BUFG_GT 出力ピンに自動生成されるクロックを制御できます。

たとえば、ワースト ケースのクロック分周が 3 の場合、BUFG_GT に次のケース解析を適用する必要があります。

```
set_case_analysis 0 [get_pins bufg_gt_pclk/DIV[0] ]
set_case_analysis 1 [get_pins bufg_gt_pclk/DIV[1] ]
set_case_analysis 0 [get_pins bufg_gt_pclk/DIV[2] ]
```

注記: UltraScale™ および UltraScale+™ デバイスでは、GT_CHANNEL に複数の内部マルチプレクサーを介して GT_CHANNEL の出力に伝搬される複数の入力クロック (TXOUTCLK など) があります。この場合、ケース解析は GT_CHANNEL のクロック多重化制御信号 (TXSYSCLKSEL、TXOUTCLKSEL など) に同様の方法で使用し、GT_CHANNEL の出力に伝搬する入力または内部クロックを選択します。

タイミング アークのディスエーブル

set_disable_timing コマンドを使用すると、セル内のタイミング アークをディスエーブルにできます。セルの入力ポートから出力ポートまでのタイミング アークのみをディスエーブルにできます。

注記: set_disable_timing コマンドは、ポートまたはワイヤからのタイミング アークをディスエーブルにするためにも使用できます。この場合、-from および -to コマンド ライン オプションは使用せず、ポート オブジェクトまたはタイミング アーク オブジェクトのみを指定します。

一部のタイミング アークは、特定の状況进行处理するため、自動的にディスエーブルになります。たとえば、組み合わせフィードバック ループは推奨されず、正しくタイミング解析できません。このようなループは、ループ内のタイミング アークの 1 つをディスエーブルにすることにより分割されます。

別の例として、マルチプレクサーに設定されたケース解析があります。デフォルトでは、マルチプレクサーのすべてのデータ入力は出力ポートに伝搬されますが、信号にケース解析が設定されていると、1 つのデータ入力ポートのみが出力ポートに伝搬されます。この場合は、タイマーにより、それ以外のデータ入力ポートから出力ポートへのタイミング アークが分割されます。

set_disable_timing コマンドを使用すると、デザイン内のセルのタイミング アークを手動で分割できます。たとえば、組み合わせフィードバック ループを分割するためにディスエーブルにするタイミング アークをツールに判断させるのではなく、ユーザーが指定できます。

また、LUT 入力に複数のクロックが供給されているが、LUT 出力ポートには 1 つのクロックだけを伝搬する必要がある場合、伝搬しないクロックに関連するタイミング アークを分割することにより対処します。

LUTRAM が関係する状況もよくあります。LUTRAM 内には書き込みクロックと読み出しクロックの間に WCLK ピンから出力 O ピンへの物理的なパスがありますが、LUTRAM ベースの非同期 FIFO は、WCLK から O への CDC パスが発生しないように設計されています。それでも、このタイミング アークはイネーブルになっており、この WCLK から O へのタイミング アークを介するパスがレポートされ、このアークが原因で TIMING-10 DRC 違反が発生する可能性があります。その場合、WCLK から O のアークをディスエーブルにし、これらのパスのタイミング解析が実行されて無効な DRC 違反が発生するのを回避する必要があります。このタイミング アークは、ザイリンクス LUTRAM ベースの FIFO の現在のインプリメンテーションでは自動的にディスエーブルになっています。

注記: タイミング アークをディスエーブルにすると、そのアークを介するタイミング パスはレポートされなくなります。有効なタイミング アークをディスエーブルにしないように注意してください。ハードウェアでデザイン エラーとなるタイミング違反やタイミング問題を見逃す結果になる可能性があります。

set_disable_arc コマンドの構文は、次のとおりです。

```
set_disable_timing [-from <arg>] [-to <arg>] [-quiet] [-verbose] <objects>
```

-from および -to オプションには、Vivado オブジェクトではなく、ピン名のみを指定します。また、ピン名には、デザイン ピン名ではなく、ライブラリ セルからのピン名を使用する必要があります。次に例を示します。

```
set_disable_timing -from WCLK -to 0 [get_cells inst_fifo_gen/ gdm.dm/gpr1.dout_i_reg[*]]
```

上記のコマンドは、LUTRAM ベースの非同期 FIFO inst_fifo_gen/ gdm.dm/gpr1.dout_i_reg[*] すべてに対し、タイミング アーク WCLK->0 をディスエーブルにします。

-from および -to の指定はオプションです。-from を指定しない場合、-to で指定したピンで終了する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。同様に、-to を指定しない場合、-from で指定したピンから開始する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 も -to も指定されていない場合、コマンドで指定したセルのタイミング アークすべてがディスエーブルになります。

report_disable_timing コマンドを使用すると、ツールにより自動的にディスエーブルにされているタイミング アークとユーザーが手動でディスエーブルにしたタイミング アークがレポートされます。このリストが長い場合があるので注意してください。-file コマンド ライン オプションを使用すると、ファイルに結果が保存できます。

注記: -cells を使用して report_disable_timing を実行する階層モジュールを指定できます。

CDC 制約

CDC 制約について

クロック乗せ換え (CDC) 制約は、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 パスに適用されます。ソース クロックとデスティネーション クロックの関係および CDC パスに設定されたタイミング例外によって、同期 CDC と非同期 CDC があります。たとえば、同期クロック間の CDC パスにフォルス パス制約が設定されている場合はタイミング解析は実行されず、非同期 CDC と同様に処理されます。

非同期 CDC パスには、安全なものと危険なものがあります。非同期 CDC に関して使用される「安全」および「危険」という用語は、クロック間のタイミング解析に使用されるものとは異なります (`report_clock_interaction` を参照)。非同期 CDC パスは、同期回路を使用してデスティネーション シーケンシャル セルがメタステーブル状態になるのを回避している場合は、安全とみなされます。

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 テクニク』(UG906) [\[参照 4\]](#) のこのセクションを参照してください。

CDC パスのタイミング解析は、`set_false_path` または `set_clock_groups` 制約を使用すると完全に無視でき、`set_max_delay -datapath_only` を使用すると部分的に解析できます。また、複数ビット CDC パスのキャプチャ時間スプレッドは `set_bus_skew` 制約を使用して制約できます。

バス スキュー制約

バス スキュー制約について

バス スキュー制約は、複数の非同期 CDC パス間の最大スキュー要件を設定するために使用します。バス スキューは、タイミング パスに関連付けられている従来のクロック スキューとは異なり、同じ `set_bus_skew` 制約が適用されているすべてのパス間におけるキャプチャ時間の最大差に対応します。バス スキュー要件はファースト コーナーおよびスロー コーナーの両方に適用されますが、コーナーを超えては解析されません。

バス スキュー制約の目的は、データを送信可能で、1 つのデスティネーション クロック エッジで受信可能なソース クロック エッジの数を制限することです。許容誤差は、制約が設定されるパスに使用されている CDC 同期機構によって異なります。通常、バス スキュー制約は次の CDC トポロジに使用されます。

- 非同期 FIFO などを使用されるグレイ コード バス転送
- CE、MUX、または MUX Hold 回路でインプリメントされる複数ビット CDC
- コンフィギュレーション レジスタ

set_bus_skew コマンドを使用すると、安全にタイミング解析可能な同期 CDC にもバス スキュー制約を設定できてしまいますが、このような制約は必要ありません。セットアップおよびホールド チェックで、安全にタイミング解析可能な 2 つの同期 CDC パス間で安全にデータが転送されるよう確認されます。

バス スキュー制約を使用する CDC は、次のとおりです。

- set_clock_groups が適用されている非同期 CDC
- set_false_path または set_max_delay -datapath_only、あるいはその両方が全体に適用されている非同期 CDC
- set_false_path または set_max_delay -datapath_only、あるいはその両方が適用されている同期 CDC パス

バス スキュー制約は、タイミング例外ではなくタイミング アサーションなので、タイミング例外 (set_clock_group、set_false_path、set_max_delay、set_max_delay -datapath_only、および set_multicycle_path) およびそれより優先される制約には影響しません。

バス スキュー制約は、route_design コマンドでのみ最適化されます。set_bus_skew 制約をレポートするには、コマンド ラインで report_bus_skew コマンドを使用するか、GUI で [Reports] → [Timing] → [Report Bus Skew] をクリックします。バス スキュー制約は、タイミング サマリ レポート (report_timing_summary) ではレポートされません。

set_bus_skew コマンドの構文

基本的なオプションを使用した set_bus_skew コマンドの構文は、次のとおりです。

```
set_bus_skew [-from <args>] [-to <args>] [-through <args>] <value>
```

-from オプションでは、有効な始点のオブジェクト リストを指定する必要があります。set_bus_skew の有効な始点は、クロックおよびレジスタや RAM などの順次エレメントのクロック ピンです。入力ポート (または入出力ポート) は set_bus_skew ではサポートされません。

-to オプションでは、有効な終点のノード リストを指定する必要があります。set_bus_skew の有効な終点は、クロックおよびレジスタや RAM などの順次エレメントのクロック ピンです。出力ポート (または入出力ポート) は set_bus_skew ではサポートされません。

-through オプションでは、有効なピンまたはネットのリストを指定する必要があります。

-from および -to オプションではクロックを参照することもできますが、より詳細な設定にし、制約ごとに限定された始点および終点のリストを指定することをお勧めします。これにより、各制約が適用されるパスの数が多くなりすぎるのを回避でき、各制約が適度に満たされるようになります。

注記: バス スキュー制約を指定する場合、-from および -to オプションの両方を指定する必要があります。

注記: バス スキュー制約は、ファンアウトのないパスに設定することをお勧めします。また、各バス スキュー制約で少なくとも 2 つの始点と 2 つの終点を指定してください。

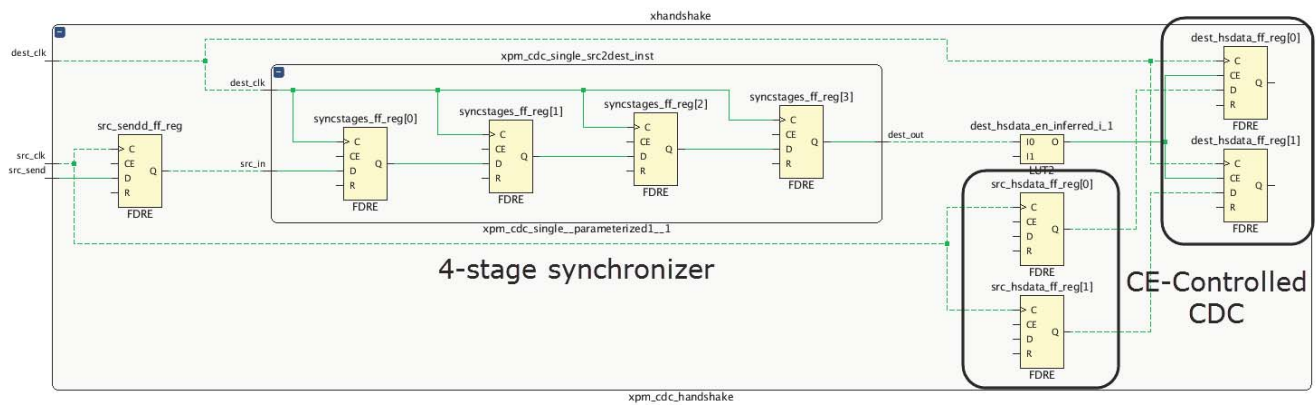
バス スキューは、現実的で適度な値にする必要があります。ソース クロックとデスティネーション クロックの最小周期の 1/2 より大きい値にすることをお勧めします。バス スキューに推奨される値は、次の例に示すように、CDC トポロジによっても異なります。

set_bus_skew の例 1

この例では、CDC はハンドシェイク メカニズムの一部です。データがサンプリングできるようになると、ソース クロック ドメインで send 信号が生成されます。デスティネーション クロック ドメインでは、この send 信号に 4 段のシンクロナイザーが使用されます。4 段のシンクロナイザーの後、この信号は CDC デスティネーション レジスタのクロック イネーブルピンを駆動します。このようなクロック イネーブル制御 CDC 構造では、バス スキューはデータが有効であるデスティネーション クロック サイクル数を表すので、バス スキューを CE パス上の段数に調整する必要があります。

ソース クロック周期が 5 ns でデスティネーション クロック周期が 2.5 ns である場合、CDC パスのバス スキューは 10 ns (4×2.5 ns) に設定する必要があります。

```
set_bus_skew -from [get_cells src_hsdata_ff_reg*] -to [get_cells dest_hsdata_ff_reg*] 10.000
```



X18887-031717

図 6-1: set_bus_skew の例 1

注記: この CDC を完全なものにするため、set_max_delay 制約を追加してソース 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する必要があります。

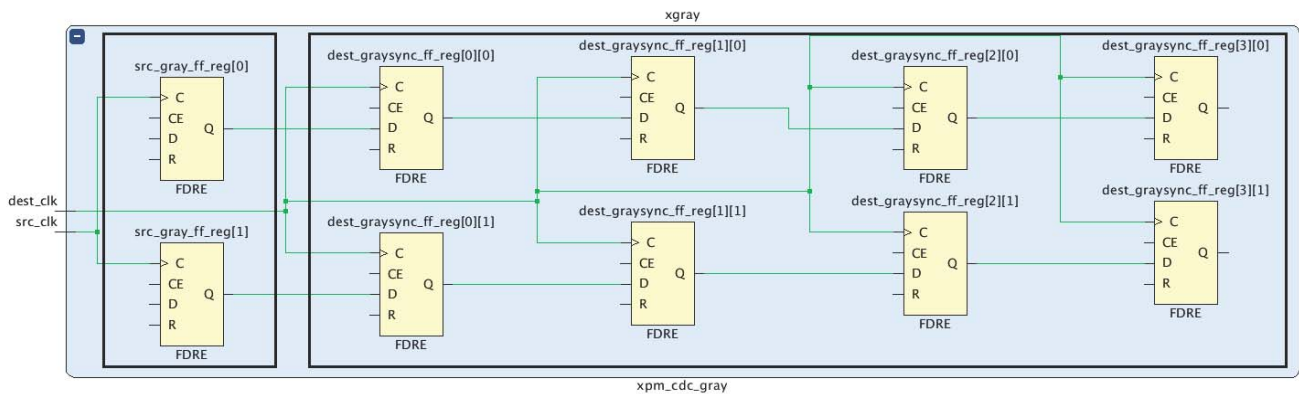
```
set_max_delay -datapath_only -from [get_cells src_hsdata_ff_reg*] -to [get_cells dest_hsdata_ff_reg*] 10.000
```

set_bus_skew の例 2

この例では、CDC はグレイコード バス上にあります。このシステムでは、グレイコード バスの 1 つの遷移のみがデスティネーション クロック ドメインで同時にキャプチャされるようにする必要があります。

ソース クロック周期が 5 ns でデスティネーション クロック周期が 2.5 ns である場合、CDC パスのバス スキューは 2.5 ns (クロック周期) に設定する必要があります。

```
set_bus_skew -from [get_cells src_gray_ff_reg*] -to [get_cells {dest_graysync_ff_reg[0]*}] 2.500
```

X18854-031717

図 6-2: set_bus_skew Example Two

注記: この CDC を完全なものにするため、set_max_delay 制約を追加してソース 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する必要があります。この場合、CDC は低速クロックと高速クロックの間にあり、バスの 1 つの遷移のみがデスティネーション クロック ドメインでキャプチャされるようにするため、最大遅延をソース クロック周期に設定します。

```
set_max_delay -datapath_only -from [get_cells src_gray_ff_reg*] -to [get_cells {dest_graysync_ff_reg[0]*}] 5.000
```

[Set Bus Skew] ダイアログ ボックス

Vivado IDE では、バス スキュー制約を複数の方法で設定できます。

- [Timing Constraints] ウィンドウを使用。[Window] → [Timing Constraint] → [Assertion] → [Set Bus Skew] をクリックします。

[Timing Constraints] ウィンドウで、バス スキュー制約を追加、削除、または変更します。

注記: ロックされている IP のバス スキュー制約は変更できません。

- CDC レポートから。[Reports] → [Timing] → [Report CDC] をクリックします。

CDC レポートの表で 2 つ以上の始点と 2 つ以上の終点を含む列を 1 つまたは複数選択し、右クリックして [Set Bus Skew] をクリックし、次のいずれかをクリックします。

- [Startpoint to Endpoint]:

選択した行に含まれる始点と終点の間にバス スキュー制約を設定します。

- [Source Clock to Destination Clock]:

始点と終点のクロック ドメイン間にバス スキュー制約を設定します。

注記: クロック ドメイン間にバス スキュー制約を設定すると、必要以上のパスに制約が適用されるので、通常はお勧めしません。インプリメンテーションの実行時間が長くなり、タイミング クロージャを達成するのが不可能になる可能性があります。

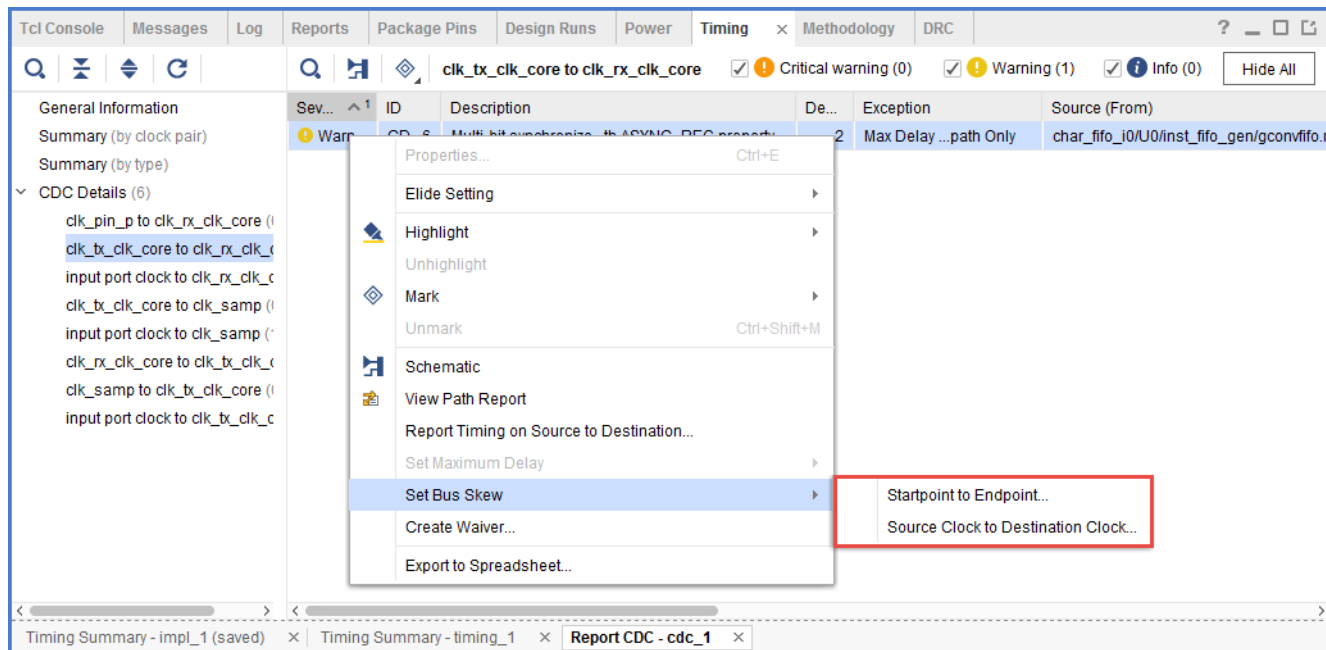


図 6-3: CDC レポートからバス スキュー制約を設定

注記: Vivado では、選択したオブジェクトに設定したバス スキュー制約が有効かどうかは検証されません。選択したオブジェクトに対してバス スキュー制約が適切なものであることを確認してください。

[Set Bus Skew] ダイアログ ボックスでは、次の図に示すように、バス スキュー値、始点、および終点を指定できます。

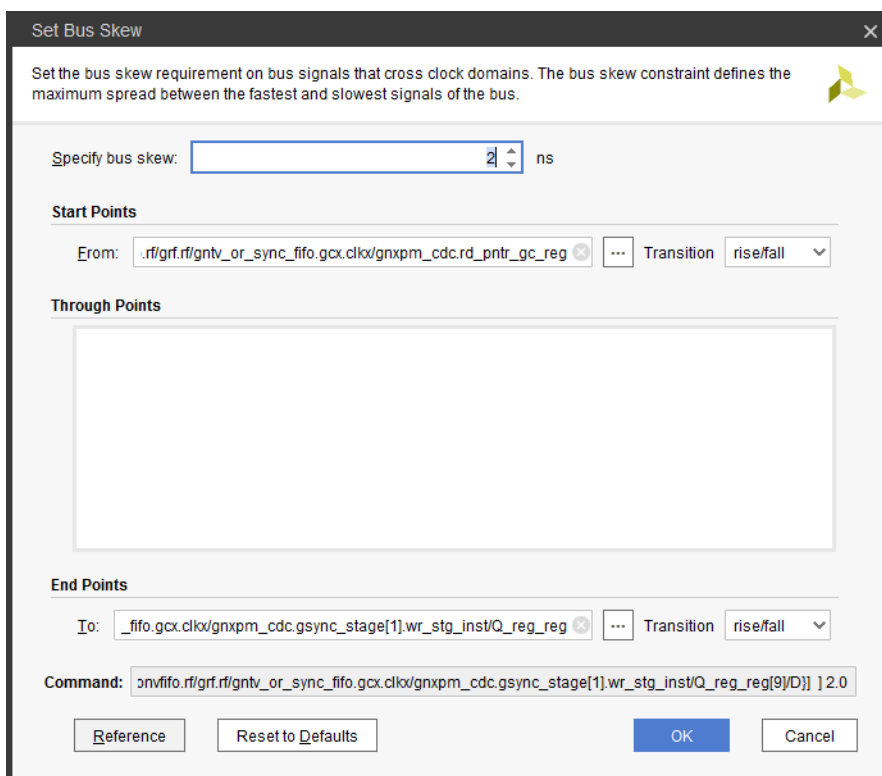


図 6-4: [Set Bus Skew] ダイアログ ボックス

XDC の優先順位

XDC の優先順位について

ザイリンクス デザイン制約 (XDC) の優先順位は、SDC (Synopsys Design Constraints) と同じです。この章では、制約の競合および重複がどのように解決されるかを説明します。

XDC 制約の順序

XDC 制約は順次解釈されるコマンドで、優先順位が同じ場合は、後のものが優先されます。

制約の優先順位の例

```
> create_clock -name clk1 -period 10 [get_ports clk_in1]
> create_clock -name clk2 -period 11 [get_ports clk_in1]
```

この例では、次の理由で最初のクロック定義が 2 つ目のクロック定義で置き換えられます。

- どちらも同じ入力に設定されている。
- `create_clock -add` オプションが使用されていない。

例外の優先順位

複数のタイミング例外が同じパスに適用されているなど、制約が重複する場合は、優先順位は次のようになります。

1. クロック グループ (`set_clock_groups`)
2. フォルス パス (`set_false_path`)
3. 最大遅延パス (`set_max_delay`) および最小遅延パス (`set_min_delay`)
4. マルチサイクル パス (`set_multicycle_path`)

注記: `set_bus_skew` 制約は、上記の制約の優先順位には影響しません。`set_bus_skew` 制約がほかの制約より優先されることはなく、またクロック グループ、最大遅延、フォルス パス、およびマルチサイクル パスにより無効になることはありません。これは、バス スキューは特定のパスに設定される制約ではなく、パス間に設定される制約だからです。

同じタイプの例外の場合は、制約が詳細に指定されているものほど優先されます。制約で使用されているフィルター オプションおよびオブジェクトのタイプによって、制約の特性を変更できます。

オブジェクトの優先順位は、次のとおりです。

1. ポート、ピン、およびセル

セル自体ではなくセルのピンが使用されます。

2. クロック

クロックの優先順位は、常にポート、ピン、およびセルよりも低くなります。クロック オブジェクトを使用するタイミング例外よりも、ポート、ピン、およびセルを使用して定義された別のタイミング例外の方が優先されます。

フィルターの優先順位を高い方から示すと、次のようになります。

1. -from -through -to
2. -from -to
3. -from -through
4. -from
5. -through -to
6. -to
7. -through



重要: -from または -to のいずれかに使用されているセルは、クロックがより限定的な -from -through -to で使用されていても、クロックより優先されます。

例外の優先順位の例

```
> set_max_delay 12 -from [get_clocks clk1] -to [get_clocks clk2]
> set_max_delay 15 -from [get_clocks clk1]
```

この例では、clk1 から clk2 へのパスに、2 つ目の制約ではなく 1 つ目の制約が適用されます。

例外で使用する -through オプションの数は、優先順位には影響しません。タイミング エンジンでは、最も厳しい制約が使用されます。

```
> set_max_delay 12 -from [get_cells inst0] -to [get_cells inst1]
> set_max_delay 15 -from [get_clocks clk1] -through [get_pins hier0/p0] -to
[get_cells inst1]
```

この例では、最初の制約ではセル オブジェクトのみが使用され、2 番目の制約ではクロック オブジェクトが使用されています。inst0 には clk1 クロックが供給されていますが、inst0 セルから inst1 セルへのパスには 2 番目の制約ではなく最初の制約が適用されます。

複数の -through オプションを含む例外制約の優先順位の例

```
> set_max_delay 4 -through [get_pins inst0/I0]
> set_max_delay 5 -through [get_pins inst0/I0] -through [get_pins inst1/I3]
```

両方の例外がタイミング エンジンで保持されます。タイミング解析には、より厳しい制約が使用されます。この例では、ピン inst1/I3 を通過するパスでも 4 ns 最大遅延制約が使用されます。



推奨: タイミング解析の結果が優先順位によって決定されるのを回避し、制約の効果を検証しやすくするため、同じパスに複数のタイミング例外を使用することは避けてください。

report_exceptions コマンドを使用してタイミング例外を検証することをお勧めします。このコマンドを実行すると、どのタイミング例外が無効になり無視されているかがわかります。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) [\[参照 4\]](#) を参照してください。

制約にオブジェクトではなく文字列が渡される場合は、Tcl インタープリターは次の順序でオブジェクトで文字列を検索します。

1. ポート
2. ピン
3. セル
4. ネット

検索はすべてのオブジェクトに対して実行されるわけではなく、あるタイプで文字列パターンに一致するオブジェクトが見つかったら、リストの後の方にある別のタイプのオブジェクトに同じ文字列パターンに一致するオブジェクトがあっても、最初に見つかったタイプのオブジェクトが返されます。

物理制約

物理制約について

Vivado® 統合設計環境 (IDE) では、デザイン オブジェクトの物理制約はオブジェクト プロパティを設定することにより指定します。次に例を示します。

- ロケーションや I/O 規格などの I/O 制約
- セル ロケーションなどの配置制約
- 固定配置などの配線制約
- コンフィギュレーション モードなどのコンフィギュレーション制約

タイミング制約と同様、物理制約も、デザインを開いたときに読み込むことができるようにするため、ザイリンクス デザイン制約 (XDC) ファイルまたは Tcl スクリプトに保存する必要があります。デザインをメモリに読み込んだら、Tcl コンソールを使用するか、または Vivado Design Suite IDE の編集ツールを使用して、インタラクティブに新しい制約を入力できます。

ほとんどの物理制約は、オブジェクトのプロパティとして定義します。

```
set_property <property> <value> <object list>
```

ただし、エリア制約には Pblock コマンドを使用します。

クリティカル警告

デザインに存在しないオブジェクトに適用されている制約など、XDC ファイルの無効な制約に対しては、クリティカル警告が表示されます。

プロパティの定義および使用法は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) [参照 11] を参照してください。



推奨: デザインを正しく制約するため、クリティカル警告をすべて確認することをお勧めします。無効な制約をインタラクティブに適用すると、エラーが発生します。

ネットリスト制約

ネットリスト制約は、合成およびインプリメンテーションで特別な方法で処理するため、ポート、ピン、ネット、セルなどのネットリスト オブジェクトに設定されます。



重要: これらの制約を使用した場合の影響を、十分に理解しておく必要があります。デザインのエリアが増加したり、デザイン パフォーマンスが低下したり、またはその両方が発生する可能性があります。

ネットリスト制約には、次のものがあります。

- 「[CLOCK_DEDICATED_ROUTE](#)」
- 「[MARK_DEBUG](#)」
- 「[DONT_TOUCH](#)」
- 「[LOCK_PINS](#)」

CLOCK_DEDICATED_ROUTE

クロック信号の配線方法を指定するには、ネットに `CLOCK_DEDICATED_ROUTE` を設定します。

`CLOCK_DEDICATED_ROUTE` プロパティはクロック ネットに使用し、デフォルト配線の代わりに使用する配線を指定します。この制約はタイミングの予測性および配線性に影響する可能性があるため、使用する場合は十分な注意が必要です。

たとえば、専用クロック配線が使用できない場合は、`CLOCK_DEDICATED_ROUTE` を `FALSE` に設定できます。`FALSE` に設定すると、クロックを入力ポートから `BUFG` や `MMCM` などのグローバルクロック リソースに配線するのに、汎用配線リソースを使用できるようになります。これは、デバイス パッケージ ピン割り当てが固定されている場合や、クロック入力を適切なクロック兼用入力ピン (CCIO) に割り当てることができない場合など、最終手段としてのみ使用してください。`FIXED_ROUTE` と共に使用しないと、配線が最適なものにならず、予測不可能になります。

このプロパティの詳細は、『UltraFast 設計手法 (Vivado Design Suite 用)』(UG949) [\[参照 5\]](#) のこのセクションを参照してください。

MARK_DEBUG

ネットを保持し、ネットリストに含まれるようにするには、RTL でネットに `MARK_DEBUG` を設定します。これにより、コンパイル フローの任意の時点で、ネットをロジック デバッグ ツールに接続できます。

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) [\[参照 12\]](#) のこのセクションを参照してください。

DONT_TOUCH

最下位セル、階層セル、またはネット オブジェクトがネットリスト最適化中に削除されないようにするには、DONT_TOUCH を設定します。DONT_TOUCH は、一般的には次の目的で使用されます。

- ネットが最適化で削除されないようにする。

DONT_TOUCH が設定されたネットは、合成またはインプリメンテーションで吸収されません。これは、ロジックのプローブや、デザインで予期しない最適化をデバッグする場合に便利です。複数の階層セグメントにまたがるネットを保持するには、そのネットの親 (`get_property PARENT $net`)、つまりネットのドライバーに一番近いネット セグメントに DONT_TOUCH を設定します。

- 手動で複製したロジックが統合されないようにする。

広範囲に伝搬されるファンアウトの大きいドライバーなど、ロジックを手動で複製するのが最適な場合があります。元のドライバーと手動で複製したドライバーに DONT_TOUCH を追加すると、合成およびインプリメンテーションの最適化でこれらのセルが削除されることはありません。

注記: DONT_TOUCH プロパティをリセットするには、`get_property` を使用します。DONT_TOUCH プロパティを 0 に設定してもプロパティはリセットされません。



ヒント: Vivado インプリメンテーションでは論理階層はフラット化されないで、インプリメンテーションで階層セルに DONT_TOUCH を使用しないでください。XDC 制約を適用する論理階層を保持するには、合成で KEEP_HIERARCHY を使用してください。

LOCK_PINS

LOCK_PINS はセルプロパティで、論理 LUT 入力 (I0、I1、I2、...) と LUT 物理入力ピン (A6、A5、A4、...) のマップを指定します。

通常、タイミング クリティカルな LUT 入力を高速の A6 および A5 物理 LUT 入力にマップするために使用されます。

LOCK_PINS 制約の例 1

次の例では、I1 を A6 に、I0 を A5 にマップしています (デフォルトのマップをスワップ)。

```
% set myLUT2 [get_cells u0/u1/i_365]
% set_property LOCK_PINS {I0:A5 I1:A6} $myLUT2
# Which you can verify by typing the following line in the Tcl Console:
% get_property LOCK_PINS $myLUT2
```

LOCK_PINS 制約の例 2

次の例では、LUT6 の I0 を A6 にマップしています。I1 ~ I5 のマップは固定されません。

```
% set_property LOCK_PINS I0:A6 [get_cell u0/u1/i_768]
```

I/O 制約

I/O 制約は次のものに設定します。

- ポート
- ポートに接続されているセル

一般的な制約は、次のとおりです。

- I/O 規格
- I/O ロケーション

Vivado Design Suite では、ISE® Design Suite の I/O 制約の多くがサポートされています。次の I/O プロパティのリストには、すべてのプロパティが含まれているわけではありません。

- 。 I/O プロパティの完全リスト、I/O ポートや I/O セルのプロパティの詳細、コードの構文例は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) [\[参照 11\]](#) を参照してください。

注記: 特に指定されていない限り、すべてのプロパティはポート オブジェクトに適用されます。

- 。 これらのプロパティの適用方法などについては、『7 シリーズ FPGA SelectIO リソース ユーザー ガイド』(UG471) の [\[参照 13\]](#) など、デバイスの SelectIO の資料を参照してください。

- **DRIVE**

出力バッファの駆動電流を mA で指定します。一部の I/O 規格でのみ使用可能です。

- **IOSTANDARD**

I/O 規格を設定します。

- **SLEW**

デバイス出力のスルー レート (遷移レート) を設定します。

- **IN_TERM**

入力ポートの入力終端抵抗のコンフィギュレーションを設定します。

- **DIFF_TERM**

IBUFDS_DIFF_OUT などのプリミティブに対して 100 オームの差動終端のオン/オフを切り替えます。

- **KEEPER**

トライステート出力または双方向ポートにウィーク ドライバーを適用し、駆動されていない場合に値を保持します。

- **PULLTYPE**

トライステート出力または双方向ポートにウィーク Low または High を適用し、フローティングしないようにします。

- **DCI_CASCADE**

マスターおよびスレーブ バンクのセットを定義します。DCI 基準電圧は、マスター バンクからスレーブにチェーン接続されます。DCI_CASCADE は IOBANK オブジェクトに設定されます。

- **INTERNAL_VREF**

I/O バンクの Vref ピンを解放し、代わりに内部で生成された Vref を使用します。INTERNAL_VREF は IOBANK オブジェクトに設定されます。

- **IODELAY_GROUP**

IDELAY および IODELAY セルのセットを IDELAYCTRL とグループにし、デザインの IDELAYCTRL が自動的に複製および配置されるようにします。

- **IOB**

フリップフロップをファブリック スライスではなく I/O ロジックに配置するよう試みます。このプロパティは、ポートではなくレジスタに設定する必要があります。



重要: IOB の処理方法は、ISE Design Suite と Vivado Design Suite で異なります。Vivado ツールでは、IOB をポートおよびポートに接続されているレジスタセルの両方に設定できます。ポートとそのレジスタに競合する値が設定されている場合、レジスタに設定されている値が使用されます。Vivado ツールで利用できる値は、TRUE および FALSE のみです。FORCE は TRUE として処理され、AUTO は無視されます。IOB の TRUE 設定を適用できない場合、ISE とは異なり、Vivado ツールではエラーではなくクリティカル警告が生成されます。

- **IOB_TRI_REG**

UltraScale+ デバイスの HDIO 用で、HDIO バンクの IOB のトライステート信号を駆動する IOB フリップフロップを、ファブリック スライスではなく、I/O ロジックに配置するよう試みます。このプロパティは、ポートではなくレジスタに設定する必要があります。

配置制約

配置制約はセルに適用し、デバイス内でのロケーションを制御します。Vivado IDE では、ISE Design Suite および PlanAhead™ ツールの配置制約の多くがサポートされています。

- **LUTNM**

2 つの LUT に固有の名前を指定し、1 つの LUT サイトに配置します。HLUTNM とは異なり、LUTNM は、異なる階層セルに属している LUT をまとめるために使用できます。

- **HLUTNM**

同じ階層にある 2 つの LUT に固有の名前を指定し、1 つの LUT サイトに配置します。

複数回インスタンス化されているセル内には、HLUTNM を使用します。

- **PROHIBIT**

サイトへの配置を禁止します。

- **PBLOCK**

論理ブロックに設定し、デバイスの物理領域に制約します。

PBLOCK は、読み取り専用のセル プロパティで、セルが割り当てられている Pblock の名前です。セル Pblock のメンバーを変更するには、XDC Tcl コマンド `add_cells_to_pblock` および `remove_cells_from_pblock` を使用します。

- **PACKAGE_PIN**

ターゲット デバイス パッケージのピンのデザイン ポートのロケーションを指定します。

- **LOC**

ネットリストの論理エレメントをデバイス上のサイトに配置します。

- **BEL**

ネットリストの論理エレメントをデバイス上のスライス内の特定の BEL に配置します。

詳細は、次を参照してください。

- [第 7 章「XDC の優先順位」](#)
- [第 9 章「相対配置マクロの定義」](#)

配置タイプ

次の 2 種類の配置があります。

- 「固定配置」
- 「固定されていない配置」

固定配置

固定配置とは、次のいずれかの方法でユーザーが指定した配置のことです。

- 手動配置
- XDC 制約
- メモリに読み込まれたデザインのセル オブジェクトに `IS_LOC_FIXED` または `IS_BEL_FIXED` を使用します。

固定されていない配置

固定されていない配置は、インプリメンテーション ツールで実行される配置です。配置が固定と設定されていると、制約が設定されているセルはインプリメンテーションで移動できません。固定配置は、単純な LOC または BEL として XDC ファイルに保存されます。

- `IS_LOC_FIXED`

LOC 制約を固定されていないものから固定されたものに変更します。

- `IS_BEL_FIXED`

BEL 制約を固定されていないものから固定されたものに変更します。

配置制約の例 1

次の例では、ブロック RAM を RAMB18_X0Y10 に配置し、その位置に固定しています。

```
% set_property LOC RAMB18_X0Y10 [get_cells u_ctrl0/ram0]
```

配置制約の例 2

次の例では、LUT をスライスの C5LUT BEL に配置し、その BEL 割り当てを固定しています。

```
% set_property BEL C5LUT [get_cells u_ctrl0/lut0]
```

配置制約の例 3

次の例では、入力遅延を短くするため入力バス レジスタを ILOGIC セルに配置しています。

```
% set_property IOB TRUE [get_cells mData_reg*]
```

配置制約の例 4

次の例では、2 つの小型の LUT を、O5 および O6 出力の両方を使用する 1 つの LUT6_2 にまとめています。

```
% set_property LUTNM L0 [get_cells {u_ctrl0/dmux0 u_ctrl0/dmux1}]
```

配置制約の例 5

次の例では、ブロック RAM の最初の列が使用されないようにしています。

```
% set_property PROHIBIT TRUE [get_sites {RAMB18_X0Y* RAMB36_X0Y*}]
```

配置制約の例 6

次の例では、配置でクロック領域 X0Y0 が使用されないようにしています。

```
% set_property PROHIBIT TRUE [get_sites -of [get_clock_regions X0Y0]]
```

配置制約の例 7

次の例では、SLR0 が使用されないようにしています。

```
% set_property PROHIBIT TRUE [get_sites -of [get_slrs SLR0]]
```



重要: 1 つのセルに BEL と LOC の両方のプロパティを割り当てる場合、まず BEL を割り当ててから、LOC を割り当てる必要があります。

配線制約

配線制約はネット オブジェクトに適用し、配線リソースを制御します。

固定配線

固定配線は、ISE の指定配線 (Directed Routing) と同様、配線を固定する機能です。ネットの配線リソースの固定には、3 つのネット プロパティが関係します。表 8-1 を参照してください。

表 8-1: ネット プロパティ

プロパティ	機能
ROUTE	読み取り専用のネット プロパティ
IS_ROUTE_FIXED	配線全体を固定するようマーク
FIXED_ROUTE	ネットの一部を固定配線

ネットの配線が確実に固定されるようにするには、その配線のすべてのセルを固定しておく必要があります。

次に、配線を完全に固定する例を示します。この例では、図 8-1 のデザインのネット netA (青色でハイライト) の配線を固定する制約を作成します。

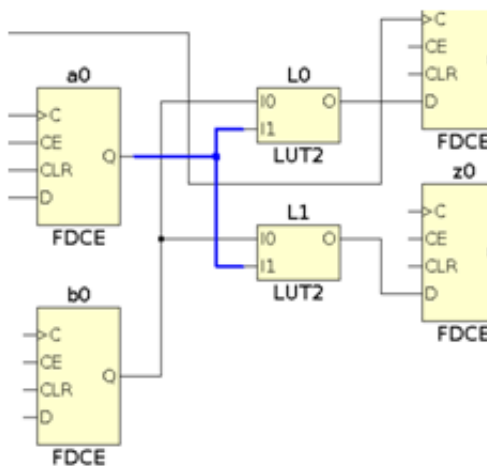


図 8-1: 配線制約の例を示すためのシンプルなデザイン

インプリメント済みデザインをメモリに読み込むと、ネットの配線情報をクエリできるようになります。

```
% set net [get_nets netA]
% get_property ROUTE $net
{ CLBLL_LL_CQ CLBLL_LOGIC_OUTS6 FAN_ALT5 FAN_BOUNCE5 { IMUX_L17 CLBLL_LL_B3 }
IMUX_L11 CLBLL_LL_A4 }
```

配線は、配線ノード名のリストで示され、ファンアウトは中かっこ ({ }) で示されます。配線を固定するには、ネットに次のプロパティを設定します。

```
% set_property IS_ROUTE_FIXED TRUE $net
```

制約を今後のために XDC ファイルにバックアノテートするには、固定されたネットに接続されているすべてのセルの配置も保持する必要があります。この情報は、[Schematic] または [Device] ウィンドウでセルを選択し、[Properties] ウィンドウの LOC/BEL プロパティ値で確認するか、または、Tcl コンソールでこれらの値を直接クエリできます。

```
% get_property LOC [get_cells {a0 L0 L1}]
SLICE_X0Y47 SLICE_X0Y47 SLICE_X0Y47
% get_property BEL [get_cells {a0 L0 L1}]
SLICEL.CFF SLICEL.A6LUT SLICEL.B6LUT
```

固定された配線はタイミング クリティカルであることが多いので、LUT ピンのマップも LUT の LOCK_PINS プロパティで指定し、配線時にピンがスワップされないようにする必要があります。

各論理ピンのサイト ピンは、Tcl コンソールでクエリできます。

```
% get_site_pins -of [get_pins {L0/I1 L0/I0}]
SLICE_X0Y47/A4 SLICE_X0Y47/A2
% get_site_pins -of [get_pins {L1/I1 L1/I0}]
SLICE_X0Y47/B3 SLICE_X0Y47/B2
```

netA の配線を固定するのに必要な完全な XDC 制約は、次のとおりです。

```
set_property BEL CFF [get_cells a0]
set_property BEL A6LUT [get_cells L0]
set_property BEL B6LUT [get_cells L1]
set_property LOC SLICE_X0Y47 [get_cells {a0 L0 L1}]
set_property LOCK_PINS {I1:A4 I0:A2} [get_cells L0]
set_property LOCK_PINS {I1:A3 I0:A2} [get_cells L1]
set_property FIXED_ROUTE { CLBLL_LL_CQ CLBLL_LOGIC_OUTS6 FAN_ALT5 FAN_BOUNCES5 {
IMUX_L17 CLBLL_LL_B3 } IMUX_L11 CLBLL_LL_A4 } [get_nets netA]
```

XDC ではなく、インタラクティブな Tcl コマンドを使用している場合、次のように、1 つの place_cell コマンドで一度に複数の配置制約を指定できます。

```
place_cell a0 SLICE_X0Y47/CFF L0 SLICE_X0Y47/A6LUT L1 SLICE_X0Y47/B6LUT
```

place_cell の詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 10] を参照してください。

コンフィギュレーション制約

コンフィギュレーション制約は、ビットストリーム生成用のグローバル制約で、現在のデザインに適用します。コンフィギュレーション モードなどの制約がこれに含まれます。

コンフィギュレーション制約の例 1

次の例は、CONFIG_MODE を M_SELECTMAP に設定します。

```
% set_property CONFIG_MODE M_SELECTMAP [current_design]
```

コンフィギュレーション制約の例 2

次の例は、デバッグ ビットストリームをオンにします。

```
% set_property BITSTREAM.GENERAL.DEBUGBITSTREAM Yes [current_design]
```

コンフィギュレーション制約の例 3

次の例では、CRC チェックをディスエーブルにしています。

```
% set_property BITSTREAM.GENERAL.CRC Disable [current_design]
```

ビットストリーム生成プロパティおよび定義のリストは、『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908) [\[参照 12\]](#) のこのセクションを参照してください。

相対配置マクロの定義

相対配置マクロの定義について

相対配置マクロ (RPM) は、複数の基本ロジック エLEMENT (BEL) をグループ化したものです。次に、ロジック エLEMENT の例を示します。

- FF
- LUT
- DSP
- RAM

RPM は、小さなロジック グループを近くに配置して、リソース効率を改善し内部接続を高速にするために主に使用します。

デザイン エLEMENT のセットの定義

U セット (U_SET) または HU セット (HU_SET) 制約は、デザイン エLEMENT のセットを定義します。

- セット内の各ELEMENTは、相対ロケーション (RLOC) 制約を使用して、同じセットのほかのELEMENTに相対的に配置されます。
- 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。

U_SET、HU_SET、および RLOC 制約には、次のような特徴があります。

- HDL デザイン ファイルでプロパティとして定義する必要があります。
- ザイリンクス デザイン制約 (XDC) ではサポートされていません。



ヒント: デザイン内で RPM のように機能するマクロ オブジェクトを Vivado® Design Suite で定義するには、create_macro または update_macro を使用します。154 ページの「XDC マクロ」を参照してください。

U_SET、HU_SET、および RLOC 制約の詳細は、『Vivado Design Suite プロパティ リファレンス ガイド』(UG912) [参照 11] を参照してください。

RPM の作成

RPM を作成するには、次の手順に従います。

1. セルをセットにグループ化します。
2. RPM セットのセルに相対ロケーションを定義します。
3. ターゲット デバイスで RPM の配置を固定するには、RPM に RLOC_ORIGIN または LOC 制約を指定します。

注記: この手順はオプションです。

セルの RPM セットへの割り当て

RLOC 制約が割り当てられた階層モジュールのデザイン エLEMENT は、自動的に RPM セットにグループ化されます。

このグループ化には、デザイン階層と RLOC 制約の組み合わせにより暗示的に定義される H_SET 制約が使用されます。

デザイン階層の 1 つのブロックに含まれる RLOC 制約が設定されたデザイン エLEMENT はすべて、U_SET または HU_SET などのセット制約が設定されていないければ、同じ H_SET に含まれると考えられます。

デザイン エLEMENT を明示的にグループ化

H_SET は、デザイン階層と RLOC 制約から暗示的に設定されますが、U_SET および HU_SET を使用してデザイン エLEMENT を明示的に RPM セットにまとめることもできます。

デザイン エLEMENT を U_SET を使用して明示的にグループ化

U_SET を使用すると、階層またはデザインでの位置にかかわらず、セルをグループ化できます。同じ set_name を持つすべてのセルは、同じ RPM セットのメンバーとなります。

U_SET 制約で設定できるデザイン エLEMENT は、プリミティブまたはプリミティブではないシンボルです。

プリミティブではないシンボルに U_SET 制約を設定すると、その下位階層にある RLOC 制約が設定されているすべてのプリミティブシンボルに適用されます。

デザイン エLEMENT を HU_SET を使用して明示的にグループ化

HU_SET には、明示的にユーザーが定義した、階層を含めた名前が付けられます。これにより、RLOC 制約を異なる階層レベルのセルに設定可能な階層型 RPM を作成できます。

階層が同じ set_name を持つセルはすべて、同じセットのメンバーとなります。

VHDL で RPM セットを定義する構文

RPM セットを VHDL の属性として定義するには、次の構文を使用します。

```
attribute U_SET : string;
attribute HU_SET : string;
...
attribute U_SET of my_reg : label is "uset0";
attribute HU_SET of other_reg : label is "huset0";
```

Verilog で RPM セットを定義する構文

RPM セットを Verilog の属性として定義する構文は、次のとおりです。

U_SET の例

```
(* U_SET = "uset0", RLOC = "X0Y0" *) FD my_reg (.C(clk), .D(d0), .Q(q0));
```

HU_SET の例

```
(* HU_SET = "huset0", RLOC = "X0Y0" *) FD other_reg (.C(clk), .D(d1), .Q(q1));
```



推奨: Vivado 合成で H_SET および HU_SET の RPM を使用する際は、RPM を含むモジュールまたはインスタンスの階層の境界を保持してください。そうすると、階層がなくなった場合でも、同じ階層にある RPM 間で名前の競合を回避できます。階層保持の詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) [\[参照 8\]](#) を参照してください。

[Physical Constraints] ウィンドウでの RPM 定義

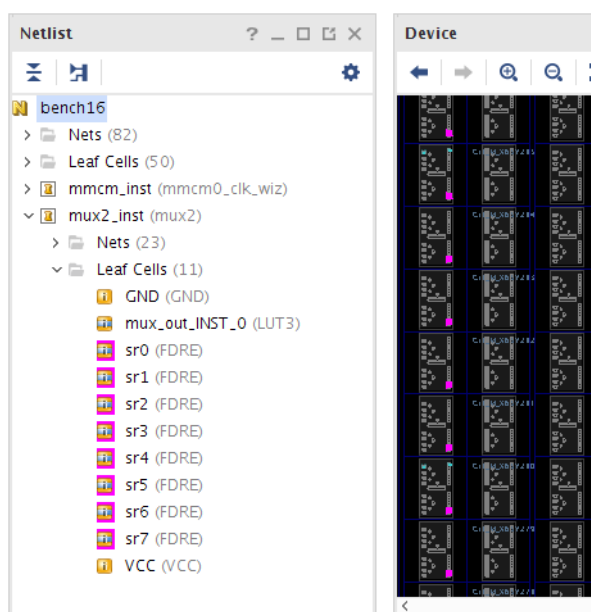


図 9-1: [Physical Constraints] ウィンドウでの RPM 定義

RPM セットは、プロパティとして HDL ソース ファイルに含める必要があります。合成後、RPM に関連するプロパティは、Vivado 配置で使用するため、ネットリスト オブジェクトに読み取り専用のプロパティとして示されます。

RPM 定義の表示

RPM 定義を表示するには、[Physical Constraints] ウィンドウを使用します。図 9-1 を参照してください。

RPM 定義を表示するには、次の手順に従います。

1. RPMs フォルダーを展開して RPM のリストを表示します。
2. RPM を選択してプロパティを確認するか、関連セルを選択します。



ヒント: RPM は、[Physical Constraints] ウィンドウから [Device] ウィンドウにドラッグして、配置および固定できます。RPM は、セルごとではなく、1 つの形として移動します。

opt_design 実行時の RPM の保持

opt_design では最適化が自由に実行され、RLOC 制約が設定されているのにもかかわらず RPM に属する LUT が削除されてしまうことがあります。opt_design の最適化により RPM 内のロジックが削除されないようにするには、RPM に属するすべてのセルの DONT_TOUCH プロパティを TRUE に設定する必要があります。DONT_TOUCH プロパティは、RTL または XDC で設定できます。

相対ロケーションの割り当て

デザイン オブジェクトに相対ロケーションを割り当てるには、RLOC プロパティを使用します。RLOC プロパティは、RPM セットの各セルの相対的な XY の座標を指定します。

RLOC プロパティを指定するには、次のいずれかの座標システムを使用できます。

- 「スライス ベースの相対座標」
- 「絶対 RPM グリッド ベース座標」

次の構文を使用します。

```
RLOC=XmYn
```

説明:

- m: オブジェクトの相対または絶対 X 座標を示す整数。
- n: オブジェクトの相対または絶対 Y 座標を示す整数。

スライス ベースの相対座標

相対グリッド システムは、次のようなシステムです。

- 標準グリッドとも呼ばれます。
- ほとんどの RPM に適しています。
- 1 つの RPM のすべてのセルが同じサイト タイプ (スライス、ブロック RAM、DSP など) に属する同種 RPM に使用されます。

注記: オブジェクトは、同じ RPM セット内のほかのオブジェクトに相対的に配置されます。

相対グリッドは標準的な長方形のグリッドで、各グリッドのエレメントは同じサイズになります。たとえば、次の Verilog コード例では、8 スライスの高さの列になり、各スライスに FD セルが含まれます。

```
(* RLOC = "X0Y0" *) FD sr0 (.C(clk), .D(d[0]), .Q(y[0]));
(* RLOC = "X0Y1" *) FD sr1 (.C(clk), .D(d[1]), .Q(y[1]));
(* RLOC = "X0Y2" *) FD sr2 (.C(clk), .D(d[2]), .Q(y[2]));
(* RLOC = "X0Y3" *) FD sr3 (.C(clk), .D(d[3]), .Q(y[3]));
(* RLOC = "X0Y4" *) FD sr4 (.C(clk), .D(d[4]), .Q(y[4]));
(* RLOC = "X0Y5" *) FD sr5 (.C(clk), .D(d[5]), .Q(y[5]));
(* RLOC = "X0Y6" *) FD sr6 (.C(clk), .D(d[6]), .Q(y[6]));
(* RLOC = "X0Y7" *) FD sr7 (.C(clk), .D(d[7]), .Q(y[7]));
```

BEL/LOC 制約

複雑な構造では、RLOCに加えて BEL または LOC 制約を指定する必要がある場合があります。BEL 制約は、LUT をレジスタと揃えるなど、RPM セット内のセルを揃えるのに使用します。LOC 制約は、RPM セットはデバイス内の特定のサイトに配置されるので、通常は使用されません。BEL または LOC 制約を指定する必要がある場合は、これらの制約のソースを混合しないようにすることが重要です。すべての BEL/LOC 制約を RTL または XDC のいずれかのみで指定し、両方を組み合わせて使用しないでください。次に、RTL で BEL 制約を指定する例を示します。

Verilog ファイル:

```
(*BEL="H6LUT",RLOC="X0Y0"*) LUT6 S0_LUTH (...);
(*BEL="G6LUT",RLOC="X0Y0"*) LUT6 S0_LUTG (...);
(*BEL="F6LUT",RLOC="X0Y0"*) LUT4 S0_LUTF (...);
(*BEL="E5LUT",RLOC="X0Y0"*) LUT4 S0_LUTE (...);
(*BEL="D6LUT",RLOC="X0Y0"*) LUT6 S0_LUTD (...);
(*BEL="C6LUT",RLOC="X0Y0"*) LUT6 S0_LUTC (...);
(*BEL="B6LUT",RLOC="X0Y0"*) LUT4 S0_LUTB (...);
(*BEL="A5LUT",RLOC="X0Y0"*) LUT4 S0_LUTA (...);

(*BEL="CARRY8",RLOC="X0Y0"*) CARRY8#(.CARRY_TYPE("DUAL_CY4")) S0_CARRY8(...);

(*BEL="HFF2",RLOC="X0Y0"*) FD FD_out5 (...);
(*BEL="GFF2",RLOC="X0Y0"*) FD FD_out4 (...);
(*BEL="FFF2",RLOC="X0Y0"*) FD FD_out3 (...);
(*BEL="DFF2",RLOC="X0Y0"*) FD FD_out2 (...);
(*BEL="CFF2",RLOC="X0Y0"*) FD FD_out1 (...);
(*BEL="BFF2",RLOC="X0Y0"*) FD FD_out0 (...);
```

注記: コマンドをシンプルにするため、INIT 文字列は省略されています。

次のコマンド例では、RPM は RTL で定義されていますが、BEL 制約は XDC で指定されています。

Verilog ファイル:

```
(*RLOC="X0Y0"*) LUT6 S0_LUTH (...);
(*RLOC="X0Y0"*) LUT6 S0_LUTG (...);
(*RLOC="X0Y0"*) LUT4 S0_LUTF (...);
(*RLOC="X0Y0"*) LUT4 S0_LUTE (...);
(*RLOC="X0Y0"*) LUT6 S0_LUTD (...);
(*RLOC="X0Y0"*) LUT6 S0_LUTC (...);
(*RLOC="X0Y0"*) LUT4 S0_LUTB (...);
(*RLOC="X0Y0"*) LUT4 S0_LUTA (...);

(*RLOC="X0Y0"*) CARRY8#(.CARRY_TYPE("DUAL_CY4")) S0_CARRY8(...);

(*RLOC="X0Y0"*) FD FD_out5 (...);
(*RLOC="X0Y0"*) FD FD_out4 (...);
(*RLOC="X0Y0"*) FD FD_out3 (...);
(*RLOC="X0Y0"*) FD FD_out2 (...);
(*RLOC="X0Y0"*) FD FD_out1 (...);
(*RLOC="X0Y0"*) FD FD_out0 (...);
```

注記: コマンドをシンプルにするため、INIT 文字列は省略されています。

XDC ファイル:

```
set_property BEL CARRY8 [get_cells S0_CARRY8]
set_property BEL HFF2 [get_cells FD_out5]
set_property BEL GFF2 [get_cells FD_out4]
set_property BEL FFF2 [get_cells FD_out3]
set_property BEL DFF2 [get_cells FD_out2]
set_property BEL CFF2 [get_cells FD_out1]
set_property BEL BFF2 [get_cells FD_out0]
set_property BEL A5LUT [get_cells S0_LUTA]
set_property BEL B6LUT [get_cells S0_LUTB]
set_property BEL C6LUT [get_cells S0_LUTC]
set_property BEL D6LUT [get_cells S0_LUTD]
set_property BEL E5LUT [get_cells S0_LUTE]
set_property BEL F6LUT [get_cells S0_LUTF]
set_property BEL G6LUT [get_cells S0_LUTG]
set_property BEL H6LUT [get_cells S0_LUTH]
```

絶対 RPM グリッド ベース座標

RPM_GRID システムは、セルが異なるサイト タイプ (スライス、ブロック RAM、DSP などの組み合わせ) に属する異種 RPM に使用されます。特定のザイリンクス デバイスにマップする絶対座標システムです。

セルはさまざまなサイズのサイトに配置できるため、RPM_GRID システムでは絶対 RPM_GRID 座標が使用されます。RPM_GRID 値は、Vivado IDE でサイトを選択すると、[Site Properties] ウィンドウに表示されます。座標は、RPM_X および RPM_Y サイト プロパティを使用して Tcl コマンドでクエリすることもできます。

RAM_GRID 座標の VHDL 例

次の VHDL 例では、RPM_GRID 座標を使用して RLOC 制約を定義しています。

- 2つのシフトレジスタをブロック RAM に相対的に配置します。
- 4段を入力に接続します。
- 4段を出力に接続します。

```
attribute RLOC : string;
attribute RPM_GRID : string;
attribute RLOC of di_reg3 : label is "X25Y0";
attribute RLOC of di_reg2 : label is "X27Y0";
attribute RLOC of di_reg1 : label is "X29Y0";
attribute RLOC of di_reg0 : label is "X31Y0";
attribute RLOC of ram0 : label is "X34Y0";
attribute RLOC of out_reg3 : label is "X37Y0";
attribute RLOC of out_reg2 : label is "X39Y0";
attribute RLOC of out_reg1 : label is "X41Y0";
attribute RLOC of out_reg0 : label is "X43Y0";
```

RPM_GRID システムを使用するためのプロパティ設定

RPM_GRID システムを使用するには、RPM セットのセルに次のようにプロパティを設定します。

```
attribute RPM_GRID of ram0 : label is "GRID";
```

少なくとも 1 つのセルの RPM_GRID プロパティが GRID に設定されていると、RPM_GRID 座標システムが使用されます。

RPM_GRID 座標はターゲットデバイスに基づく絶対値ですが、RPM セットの要素の相対的な配置を定義します。

インプリメンテーション中、RPM セットはデバイス上の任意の適切な位置に配置できます。

RAM_GRID 座標の値

RPM_GRID 座標値は、FPGA のスライスの座標値とは大きく異なります。RPM_GRID 座標値には、次のような特徴があります。

- Vivado ツールでデバイス サイトの RPM_X および RPM_Y プロパティとして保存されます。
- get_property を使用してクエリできます。

次の例では、次を実行します。

- 選択したスライスから RPM 座標を取得します。
 - join コマンドを使用して、X および Y の座標を必要なフォーマットで出力します。
- ```
join "X[get_property RPM_X [get_selected_objects]]Y[get_property RPM_Y
[get_selected_objects]]"
X25Y394
```



## RTL ソース ファイルでの RLOC プロパティの定義

標準グリッドは単純で、相対的であるため、RPM の RLOC プロパティは RTL ソース ファイルで直接定義できます。

RPM\_GRID 座標はターゲット デバイスから抽出する必要があるため、次の作業が必要になります。

- 合成後に RPM\_GRID の正しい値を見つけるため、デザインで繰り返し実行します。
- RTL ソース ファイルに座標をプロパティとして追加します。
- 配置の前にネットリストを再合成します。

## RPM への固定ロケーションの割り当て

オプションで、RLOC\_ORIGIN または LOC 制約を使用して、デバイス上に RPM を配置して、そのロケーションを固定できます。Vivado IDE では、これらのプロパティにより RPM の座標の原点 (RPM の左下角) の位置が固定されます。RPM の残りのセルはそれぞれ、相対ロケーション (RLOC) を使用して配置され、原点からオフセットされます。



図 9-2: RLOC\_ORIGIN による RPM の配置

次の例では、RLOC\_ORIGIN を使用して固定された階層 RPM を示しています。RLOC 制約は RPM レジスタ セルに割り当てられ、高さ 2 X 幅 3 の配置パターンを作成しています。



Verilog:

```
(* RLOC = "X0Y0" *) FDC sr0...
(* RLOC = "X1Y0" *) FDC sr1...
(* RLOC = "X2Y0" *) FDC sr2...
(* RLOC = "X0Y1" *) FDC sr3...
(* RLOC = "X1Y1" *) FDC sr4...
(* RLOC = "X2Y1" *) FDC sr5...
```

この RPM を 3 回インスタンスエートし、各セルに RLOC を設定します。

```
(* RLOC = "X0Y0" *) ffs u0...
(* RLOC = "X3Y2" *) ffs u1...
(* RLOC = "X6Y4" *) ffs u2...
```

最後に、RLOC\_ORIGIN を X74Y15 に設定して u0 セルに割り当てます。配置は、[図 9-2](#) に示すようになります。[表 9-1](#) に、図でのセルのハイライト色を示します。

表 9-1: セルのハイライト色

| セル | ハイライト色 |
|----|--------|
| u0 | 黄色     |
| u1 | 緑色     |
| u2 | 赤      |



**ヒント:** RPM ではロジック エLEMENTの相対配置が指定されますが、インプリメンテーションを反復実行した場合にロジックを接続するのに特定の配線リソースが使用されるとは限りません。

使用される配線の制御については、[142 ページの「配線制約」](#)を参照してください。

## XDC マクロ

XDC マクロを使用すると、合成後にセルを相対配置できます。マクロは RPM と同様の特徴を持ちますが、XDC および Tcl を使用してインタラクティブに変更できます。マクロは、相対配置制約によりグループ化された最下位セルから作成されます。

RPM は HDL コードで管理しますが、マクロは XDC 制約を使用して管理します。RPM を自動的にマクロに変換することはできません。同様に、マクロを自動的に HDL コードに追加することはできません。RPM は、マクロとは異なるオブジェクトではないので、XDC マクロ コマンドを RPM に使用することはできません。

表 9-2: RPM とマクロの比較

|                | RPM                | マクロ                                    |
|----------------|--------------------|----------------------------------------|
| 定義             | HDL 属性             | XDC 制約                                 |
| 合成後のアクセス       | 読み出し専用             | 読み出し/書き込み                              |
| 階層             | あり (H_SET/HU_SET)  | なし                                     |
| RLOC ターゲット     | 最下位セルおよび最下位以外のセル   | 最下位セルのみ                                |
| サイト タイプの混合     | 可 (RPM_GRID 属性を使用) | 可<br>(update_macro -absolute_grid を使用) |
| オブジェクトとしてのアクセス | なし                 | あり                                     |
| 保存先            | ネットリスト             | XDC または Tcl スクリプト                      |

## マクロの指定

マクロを指定するには、次の XDC Tcl コマンドを使用します。

- `create_macro`
- `update_macro`
- `delete_macros`
- `get_macros`

各コマンドで、undo および redo がサポートされています。

次に、各コマンドについて説明します。

### create\_macro

create\_macro は新しいマクロ オブジェクトを作成します。

固有のマクロ名を指定する必要があります。既存のマクロと同じ名前のマクロを作成しようとすると、エラーが生成されます。

### create\_macro の構文

```
create_macro <name>
```

## create\_macro の例

```
create_macro m0
```

m0 という名前のマクロ オブジェクトを作成します。



**ヒント:** LUT とフリップフロップのアライメントが最適になるようにするには、マクロを作成する際に BEL ロケーションを指定します。BEL ロケーションは、セル オブジェクトのプロパティとして別に設定する必要があります。次に例を示します。

```
set_property BEL AFF [get_cell u2/sr0]
```

## update\_macro

update\_macro コマンドは、マクロに最下位セルおよび相対配置 (RLOC) を追加します。

RLOC の構文および機能は、RPM の RLOC 属性と同じです。すべてのセルを一度に指定する必要があります。部分的に定義したり、段階的に定義していくことはできません。

## update\_macro の構文

```
update_macro [-absolute_grid] <macro name> <cell-RLOC list>
```

説明:

- -absolute\_grid: 混合スライスおよびスライス以外のサイト用に絶対グリッドを選択します。
  - X および Y の値は、サイト プロパティである RPM\_X および RPM\_Y になります。
  - 絶対グリッドの値は RPM\_GRID の値と同じです。
- macro name: アップデートするマクロの名前を指定します。
- cell-RLOC list: セルと RLOC のペアの Tcl リストを指定します。
 

```
{cell10 RLOC(cell10) cell11 RLOC(cell11) - cell1N RLOC(cell1N)}.
```

  - すべてのマクロ セルおよび RLOC を一度に指定する必要があります。マクロを段階的に構築することはできません。
  - 既存のマクロをアップデートするには、再作成する必要があります。

## update\_macro の例 1

```
update_macro m1 {u2/sr0 X0Y0 u2/sr1 X0Y1}
```

- u2/sr0 および u2/sr1 をマクロ m1 に追加します。
- u2/sr0 の RLOC を X0Y0 に割り当てます。
- u2/sr1 の RLOC を X0Y1 に割り当てます。

update\_macro の例 2 は、同じ設定を異なる構文で実行します。

## update\_macro の例 2

```
set rlocs [list u2/sr0 X0Y0 u2/sr1 X0Y1]
update_macro m1 $rlocs
```

### update\_macro の例 3

この例では、絶対グリッドを使用します。

```
set rlocs {ireg X2Y38 qlreg X17Y40 q2reg X17Y40}
update_macro -absolute_grid m2 $rlocs
```

### delete\_macros

delete\_macros コマンドは、指定したマクロを削除します。

### delete\_macro の構文

```
delete_macros <pattern>
```

### delete\_macro の例

```
delete_macros m1
```

### get\_macros

get\_macros コマンドは、デザインのマクロ オブジェクトを返します。

### get\_macros の構文

```
get_macros [pattern]
```

引数を指定しない場合は、get\_macros コマンドはデザインのすべてのマクロを返します。マクロ名を指定すると、対応するマクロ オブジェクトが返されます。

### get\_macros の例

get\_macros コマンドは、ほかのオブジェクト コマンドと共に使用できます。例:

```
% create_macro m1
% update_macro m1 {u2/sr0 X0Y0 u2/sr1 X0Y1}
% get_cells -of [get_macros m1]
u2/sr0 u2/sr1
% get_macros -of [get_cells u2]
m1
```

次のコマンドでは、セルに完全に含まれるマクロすべてが返されます。

```
get_macros -of [get_cells $cells]
```

get\_cells を使用すると、次のような間接的な組み合わせも可能です。

```
get_macros -of [get_cells -of [get_pblocks pb0]]
```

このコマンドでは、Pblock pb0 に含まれるマクロが返されます。

## マクロの管理

マクロは XDC 制約として保存されます。定義上、これらは Tcl コマンドです。これにより、マクロを XDC 制約ファイルと Tcl スクリプトの両方で使用でき、インタラクティブに使用できます。

マクロを保存するには `write_xdc` コマンドを使用し、読み込むには `read_xdc` コマンドを使用します。特定セルに適用範囲を限定するには、`-cell` オプションを使用します。

`-cell` オプションは、あるマクロの相対的な配置を、別の階層にある同様のインスタンスに適用する場合に、特に便利です。

### マクロの管理の例 1

マクロを含む、メモリ内のすべての XDC 制約を書き出します。

```
% write_xdc constrs.xdc
```

### マクロの管理の例 2

デザインにあるセルのインスタンスが 3 つ含まれているとします。

```
inst_0, inst_1, and inst_2.
```

マクロは `inst_0` 内に作成されます。

```
% create_macro m0
% update_macro m0 {reg0 X0Y0 reg1 X0Y1}
% write_xdc -cell inst_0 inst_0.xdc
```

### マクロの管理の例 3

セル `inst_0` に対して、マクロ `m0` を含む XDC 制約をすべて保存します。

```
% write_xdc -cell inst_0.xdc inst_0.xdc
```

### マクロの管理の例 4

セル `inst_0` からのマクロ `m0` を含む XDC 制約を読み込み、`inst_1` および `inst_2` に適用します。

```
% read_xdc inst_0 -cell {inst_1 inst_2}
% get_macros
m0 inst_1_m0 inst_2_m0
```



**ヒント:** マクロを読み出して、`-cell` で別のセルに適用する場合、その新しいマクロには独自の名前を付ける必要があります。セル名が接頭辞としてマクロ名に追加され、新しいマクロ名が作成されます。例 4 では、新しいマクロ `inst_1_m0` および `inst_2_m0` が作成されます。

## マクロのプロパティ

マクロ オブジェクトには、次のプロパティがあります。

- ABSOLUTE\_GRID
- CLASS
- NAME
- RLOCS

## マクロのプロパティの例

```
% report_property [get_macros m1]
Property Type Read-only Visible Value
ABSOLUTE_GRID bool true true 0
CLASS string true true macro
NAME string true true m1
RLOCS string* true true u2/sr0 X0Y0 u2/sr1 X0Y1
```

次に、各プロパティについて説明します。

### ABSOLUTE\_GRID

RLOC でデフォルトのグリッド システムを使用するか、絶対グリッド システムを使用するかを示すブール プロパティです。

デフォルトは `false` です。update\_macro で `-absolute_grid` を使用すると、このプロパティは `true` になります。

絶対グリッドでは、サイトの `RPM_X` および `RPM_Y` プロパティに対応する座標が使用されるので、異なるサイト タイプに配置されたセルからマクロを作成できます。

### CLASS

オブジェクトをマクロとして識別します。

### NAME

マクロ オブジェクトの名前は、create\_macro で使用された名前か、read\_xdc -cell を使用している場合は、セル階層名が接頭辞として付けられているマクロ名になります。

### RLOCS

マクロ セルとその RLOC プロパティのリストを、update\_macro コマンドで使用するのと同じフォーマットで含む文字列です。

マクロ セルには、さらに次のプロパティがあります。

- RLOC: セルの相対ロケーション プロパティ (RLOC) 値。
- MACRO\_NAME: セルが属するマクロの名前。

前述の例を使用すると、次のようになります。

```
% get_property RLOC [get_cells {u2/sr0 u2/sr1}]
X0Y0 X0Y1
% get_property MACRO_NAME [get_cells {u2/sr0 u2/sr1}]
m1 m1
```

## opt\_design 実行時の XDC マクロの保持

opt\_design では最適化が自由に実行され、RLOC 制約が設定されているのにもかかわらず XDC マクロに属する LUT が削除されます。opt\_design の最適化により XDC マクロ内のロジックが削除されないようにするには、XDC マクロに属するすべてのセルの DONT\_TOUCH プロパティを TRUE に設定する必要があります。DONT\_TOUCH プロパティは、RTL または XDC で設定できます。

## アドバンス XDC マクロの例

このセクションでは、次のアドバンス XDC マクロの例を示します。

- 「相対グリッド マクロの例」
- 「絶対グリッド マクロの例」

### 相対グリッド マクロの例

最も一般的なマクロは同じサイト タイプに属するセルで構成されているので、デフォルトではマクロの RLOC 座標に相対グリッドが使用されます。

次に、マクロの RLOC から生成された相対配置の単純な例を示します。このマクロは、2x2 パターンに並べられた 2 つの SRL > FF > FF 回路で構成されます。図 9-3 を参照してください。

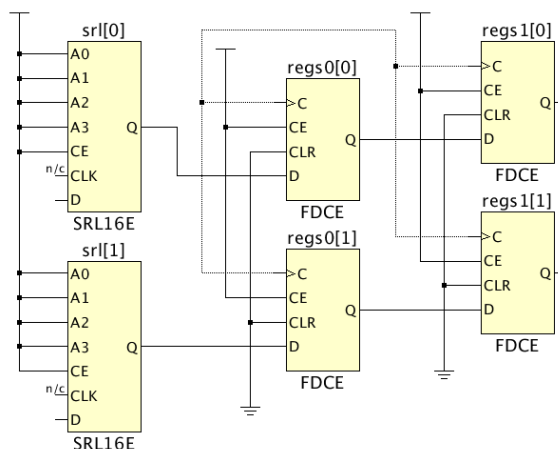


図 9-3: サンプル回路の回路図

適切な相対配置を作成するには、次のようにセルに **RLOC** を設定します。

```
srl[0] X0Y0
regs0[0] X0Y0
regs1[0] X1Y0
srl[1] X0Y1
regs0[1] X0Y1
regs1[1] X1Y1
```

次のコマンドは、m0 という名前のマクロを作成します。

```
create_macro m0
update_macro m0 {srl[0] X0Y0 regs0[0] X0Y0 regs1[0] X1Y0 srl[1] X0Y1 regs0[1] X0Y1
regs1[1] X1Y1}
```

マクロは自動的に配置させるか、セットとして手動で配置できます。マクロの配置は図 9-4 のようになります。

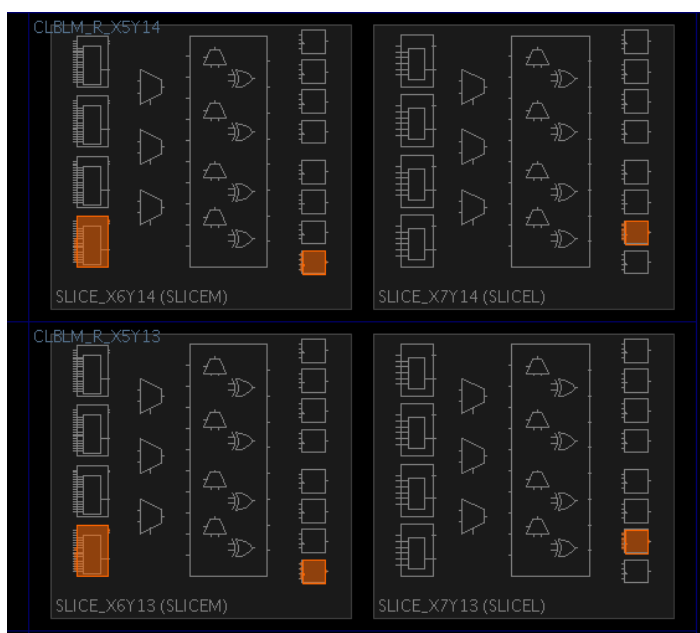


図 9-4: マクロの配置例

マクロには、LUTRAM に基づく SRL が含まれており、これは **SLICEM** タイプ スライスにのみ配置可能なので、ロケーションは多少制限されます。このマクロは、**SLICEM** 列の右に **SLICEL** 列がある場所のみ配置可能です。



**注意:** 多くのオブジェクトが配置されているスライスが一箇所に集中していると、密集の原因となり、配線が困難となり、パフォーマンスに悪影響を及ぼす可能性があります。



## 絶対グリッド マクロの例

異なるサイト タイプのセルをマクロに含める場合は、絶対グリッドを使用する必要があります。

絶対グリッド (RPM グリッドとも呼ばれる) は、デバイス内のロケーションに基づいてサイトの座標を定義する絶対座標システムです。絶対グリッドでは、サイトのサイズも考慮されます。RAM および DSP ブロックは、スライスよりも幅が広がっています。絶対グリッドを図 9-5 に示します。

この例では、3つの異なるタイプのセルから、絶対グリッドを使用してマクロが作成されています。入力ポートからの入力データパスが、2 段のレジスタを介してブロック RAM に接続されています。これが図 9-5 に示されています。

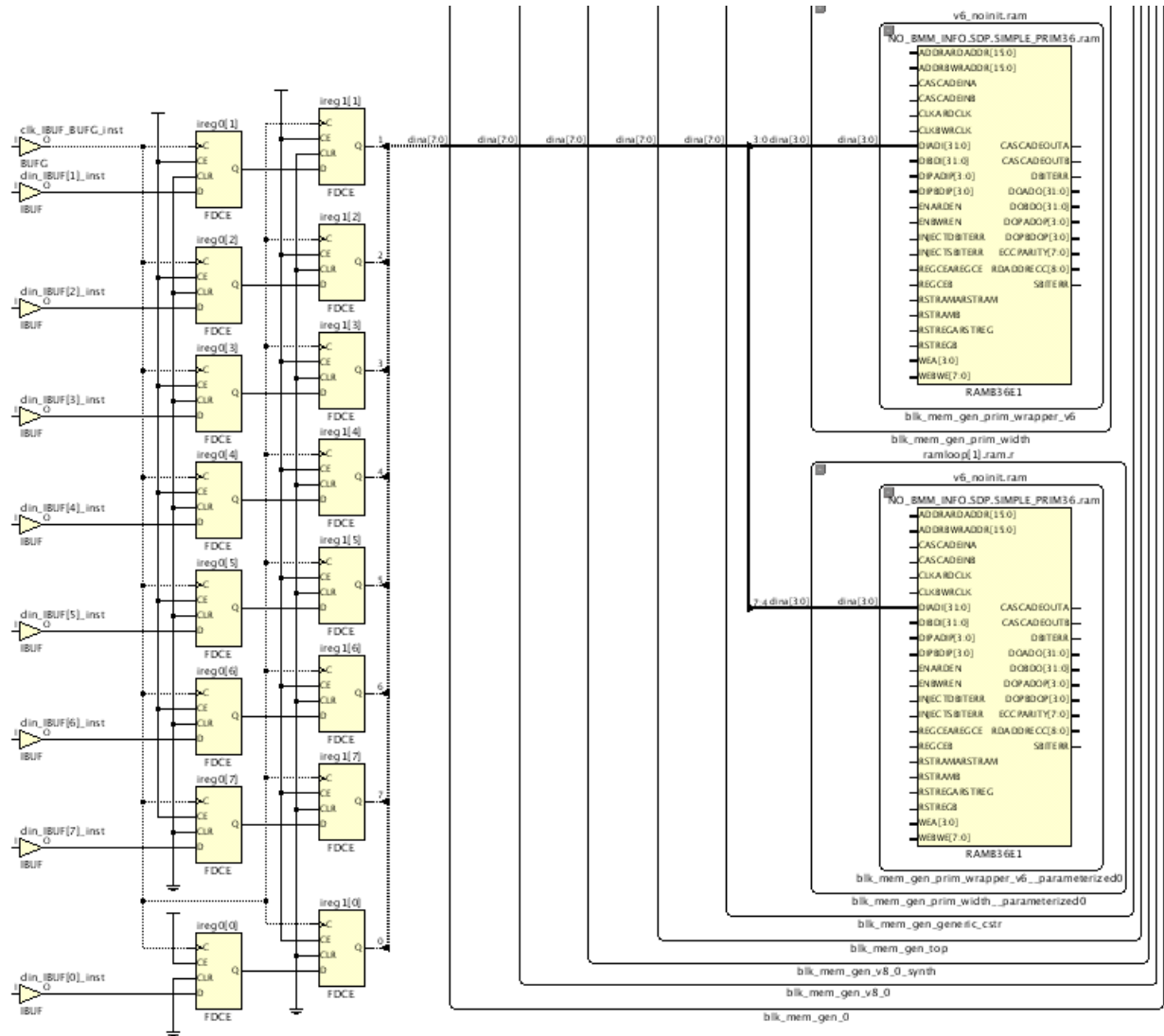


図 9-5: 絶対グリッドの回路例

マクロの作成には、セルのリストと、それらの絶対グリッドを使用した相対ロケーション (RLOC) が必要です。マクロを作成する際、絶対グリッド マクロの相対配置をイメージするのは難しいかも知れません。



**推奨:** セルを一時的にデバイスの絶対ロケーションに配置し、各セルの絶対グリッド RLOC 値を取得します。

図 9-6 に示すように、セルはまず手動で配置し、適切なロケーションに調整します。

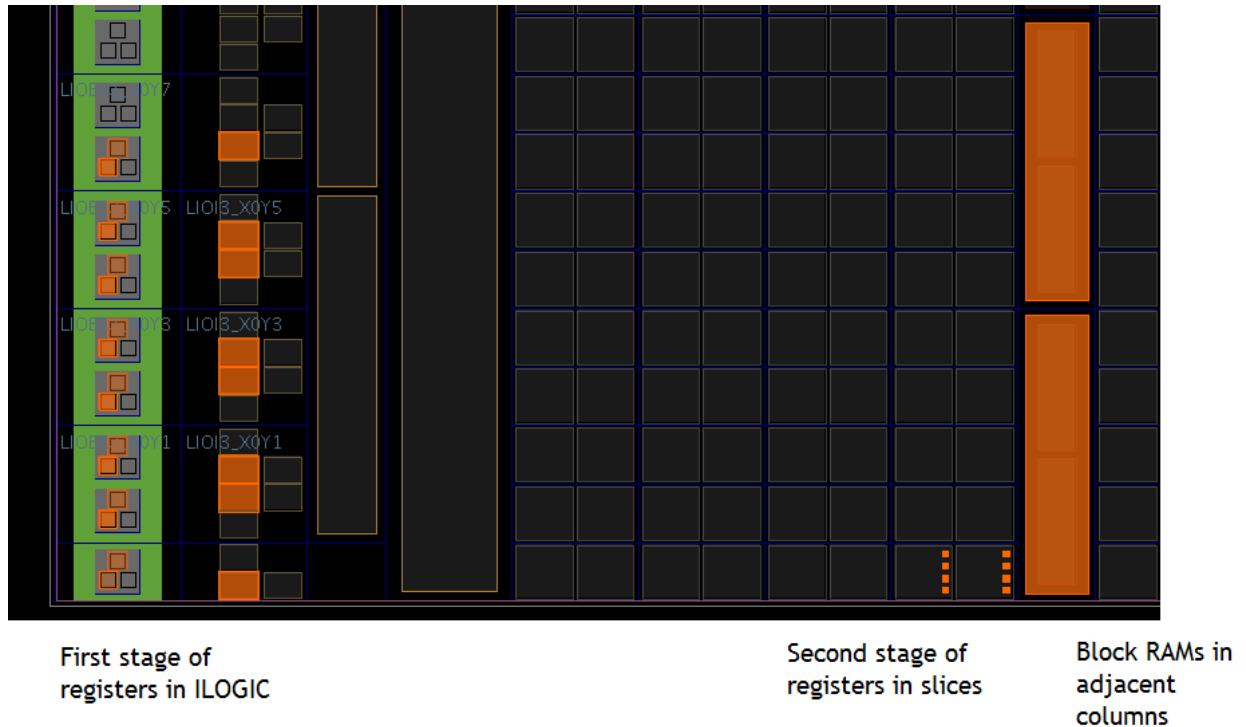


図 9-6: 絶対グリッド マクロ用にセルを手動配置

絶対グリッドでは絶対ロケーションが指定されますが、マクロの相対配置を満足できれば、デバイスのどこにでも配置できます。この例では、左下角を基点として相対ロケーションが指定されています。

ただし、絶対グリッド ロケーションでは絶対配置ではなく相対配置が指定されます。これにより、相対配置が保持されれば、マクロをデバイスのどこにでも配置できます。

この例は、ILOGIC、スライス、およびブロック RAM が含まれていて多少複雑であり、マクロのロケーションもある程度制限されますが、図 9-7 にオレンジ色でハイライトされている 3 つのどのロケーションにでも配置できます。

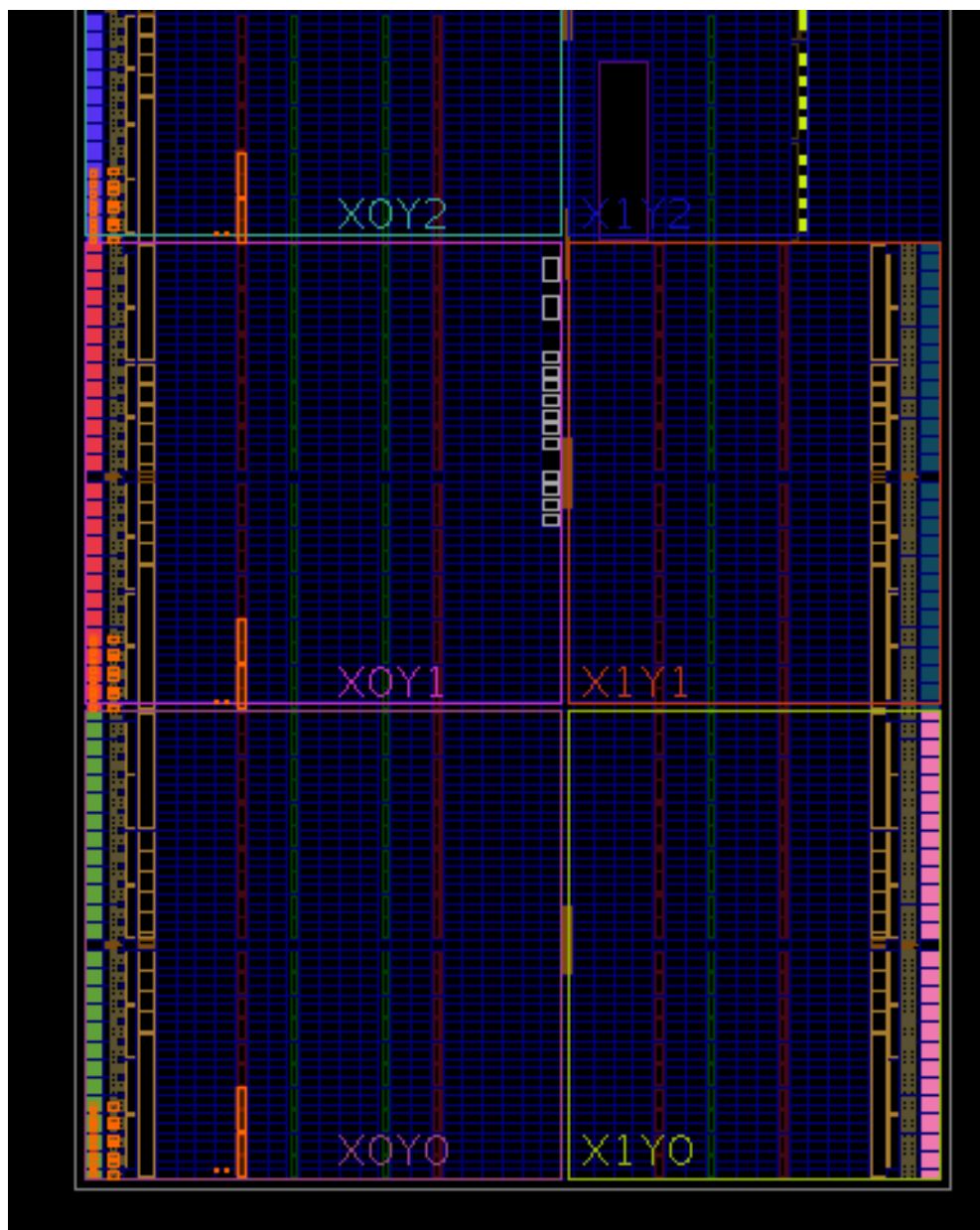


図 9-7: XDC マクロを配置可能な 3 つのロケーション

絶対グリッドの RLOC を判断するには、RPM\_X および RPM\_Y のプロパティを使用します。たとえば、下のブロック RAM はサイト RAMB36\_X0Y0 に配置されます。

セルではなくサイトを選択すると、RPM\_X が 33、RPM\_Y が 0 と表示されます (図 9-8)。これらが絶対グリッド座標です。それに対応する RLOC の値は X33Y0 です。

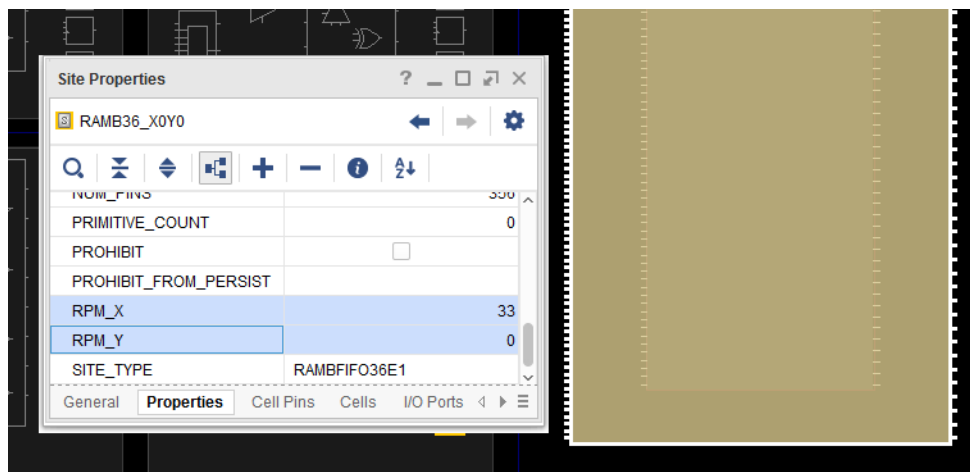


図 9-8: ブロック RAM の絶対グリッド座標

同じ方法を使用して、スライスの絶対 RLOC 値を取得できます (図 9-9)。このスライス内のセルの RLOC は X31Y0 です。

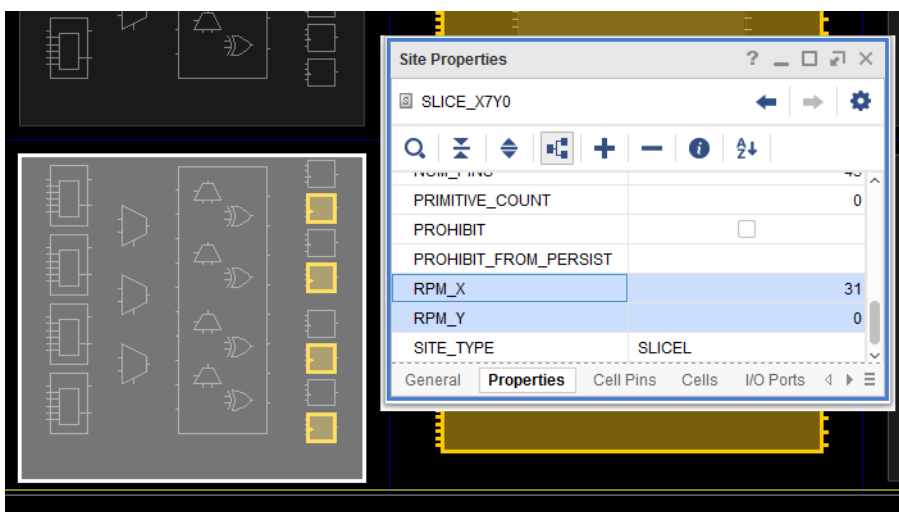


図 9-9: スライスの絶対グリッド座標

m0 という名前のマクロを作成するには、2 つのコマンドがあります。

```
create_macro m0
update_macro m0 -absolute_grid <cell0 rloc0 cell1 rloc1 cell2 rloc2 ... cellN rlocN>
```

マクロにこの例のように多数のセルが含まれている場合、Tcl コマンドを使用して `update_macro` に必要な `cell-rloc` リストを簡単に作成および指定できます。セルが配置済みの場合、絶対グリッドの RLOC は、次の Tcl コマンド `proc getAbsRLOC` を使用して取得できます。

```
proc getAbsRLOC {cell} {
 set site [get_sites -of [get_cells $cell]]
 set X [get_property RPM_X $site]
 set Y [get_property RPM_Y $site]
 return "X${X}Y${Y}"
}
```

### 例: 変数 `rloc` をブロック RAM のセルの RLOC に割り当てる

```
% set rloc [getAbsRLOC $ram0]
X33Y0
```

Tcl コマンド `dict` は、`update_macro` コマンド用に、セルと絶対グリッド RLOC のディクショナリ (連想配列) を構築するために使用できます。Tcl 連想配列は、キーと値のペアの配列です。セルおよび RLOC は、`dict` コマンドを使用して、配列にできます。配列キーがマクロのセルオブジェクトで、配列値がセルの RLOC です。これは、多数のセルを含むマクロを作成するプロセスを自動化するのに役立ちます。次の例では絶対グリッドを使用していますが、この方法は相対グリッドにも適用できます。

`$cells` はマクロセルのリストで、`$cells` の各セルが必要なマクロパターンを作成するように配置されているとすると、次の Tcl プロシージャは `update_macro` コマンド用にセルと RLOC のペアのリストを作成します。

```
proc buildRLOCList {cells} {
 set rlocs [dict create] ; # initialize dictionary called rlocs
 foreach cell $cells {
 # dictionary key is cell, value is absolute RLOC
 dict set rlocs $cell [getAbsRLOC $cell]
 }
 return $rlocs
}
```

### 例: 回路例の RLOC リストの構築

```
create macro cell list: input register stage and BRAM cells
set cells [get_cells -hier [list ireg0* ireg1* *SIMPLE_PRIM36.ram]]
create_macro m0
update_macro m0 -absolute_grid [buildRLOCList $cells]
```

`buildRLOCList` で作成されたディクショナリ リストを表示するには、次のコマンドを使用します。

```
$ puts [buildRLOCList $cells]
{ireg0[6]} X2Y10 {ireg0[5]} X2Y11 {ireg0[4]} X2Y6 {ireg0[3]} X2Y7 . . .
```

マクロセルの数が多く、階層にマクロセルが埋もれている場合は、セルと RLOC のペアのリストを明示的に指定するのは困難であり、エラーが発生しやすくなります。Tcl を使用すると、XDC マクロの作成および管理が簡単になります。

## RPM の XDC マクロへの変換

相対配置制約をインプリメントするには、XDC マクロを使用する方が推奨されるため、実現可能な場合は常に RPM を XDC マクロに変換することをお勧めします。これには、手動で HDL ソースから RPM 属性を削除して、同等の XDC マクロを作成します。また、Tcl を使用して自動的に RPM 属性を XDC マクロに置換することもできます。

自動プロセスでは、次が実行されます。

1. すべての HDL ソースで各 RPM 属性を次のような同様に命名された文字列に置き換えます。

- hu\_set を m\_hu\_set に置換。
- u\_set を m\_u\_set に置換。
- rloc を m\_rloc に置換。

これにより、RPM は処理されなくなりますが、非アクティブな属性は、セルプロパティのように合成済みネットリストに渡されるようになります。

2. 合成済みデザインを開くか、link\_design を実行して、この非アクティブなプロパティに基づいて XDC マクロを作成します。たとえば、HU\_SET にはそれぞれ m\_hu\_set というセルプロパティが付いており、同等の XDC マクロを作成するのに使用できます。元の HU\_SET 内のセルには、それぞれ RLOC に変換可能な m\_rloc プロパティが付きます。
3. この XDC マクロ定義を含むようになった制約を保存します。

変換には、Tcl を使用し、一意の m\_hu\_set または m\_u\_set 値に基づいた XDC マクロ セル リストを構築するのが最適な方法です。次は、単純な VHDL 変換例です。

元の VHDL ソースには、set0 という HU\_SET RPM が含まれ、X0Y0 という RLOC を含むセルと、X0Y1 という RLOC を含むセルの 2 つのセルが含まれます。

```
signal r0 : std_logic;
signal r1 : std_logic;

attribute hu_set : string;
attribute rloc : string;

attribute hu_set of r0 : signal is "set0";
attribute hu_set of r1 : signal is "set0";

attribute rloc of r0 : signal is "X0Y0";
attribute rloc of r1 : signal is "X0Y1";
```

次に、VHDL ソースで、hu\_set および RLOC が同様に命名された非アクティブな属性に置換されます。

```
signal r0 : std_logic;
signal r1 : std_logic;

attribute m_hu_set : string;
attribute m_rloc : string;

attribute m_hu_set of r0 : signal is "set0";
attribute m_hu_set of r1 : signal is "set0";

attribute m_rloc of r0 : signal is "X0Y0";
attribute m_rloc of r1 : signal is "X0Y1";
```

合成後は、これらの同様に命名されたプロパティに基づいてセルがフィルターできるようになります。

```
Vivado% get_cells -filter {m_hu_set == "set0"}
r0_reg r1_reg

Vivado% get_property m_rloc [get_cells {r0_reg r1_reg}]
X0Y0 X0Y1
```

これにより、XDC マクロを作成するのに必要な情報が提供され、RPM が置換されます。

```
Vivado% create_macro set0
Vivado% update_macro set0 {r0_reg X0Y0 r1_reg X0Y1}
```

これら 2 つの XDC 制約は、デザイン制約の一部として保存できます。RPM 変換の大部分には、Tcl スクリプトの使用をお勧めします。次は、HU\_SET RPM を XDC マクロに変換するスクリプト例です。

```
create a sorted list of all unique RPMs according to m_hu_set values
set RPMs [lsort -uniq [get_property m_hu_set [get_cells -hier -filter
{primitive_level != INTERNAL}]]]

remove the first element which is empty (no m_hu_set property)
set RPMs [lrange $RPMs 1 end]

iterate over list of RPMs, convert each to an XDC macro
get each RPM cell of the RPM with its RLOC
build a list for the update_macro command
foreach rpm $RPMs {
 create_macro $rpm
 set cells [get_cells -hier -filter "m_hu_set == $rpm"]
 set rlocs [list]
 foreach cell $cells {
 lappend rlocs $cell
 lappend rlocs [get_property m_rloc $cell]
 }
 update_macro $rpm $rlocs
 puts "created XDC macro $rpm, cell list: $rlocs"
}
```

## サポートされる XDC および SDC コマンド

この付録では、ザイリンクス Vivado® 統合設計環境 (IDE) でサポートされる XDC (Xilinx Design Constraints) および SDC (Synopsys Design Constraints) コマンドについて説明します。

### XDC ファイルで使用可能なコマンド

表 A-1: XDC ファイルで使用可能なコマンド

| タイミング制約                | 物理制約                       | 汎用                      |
|------------------------|----------------------------|-------------------------|
| create_clock           | add_cells_to_pblock        | set                     |
| create_generated_clock | create_pblock              | expr                    |
| group_path             | delete_pblock              | list                    |
| set_clock_groups       | remove_cells_from_pblock   | filter                  |
| set_clock_latency      | resize_pblock              | current_instance        |
| set_data_check         | create_macro               | get_hierarchy_separator |
| set_disable_timing     | delete_macros              | set_hierarchy_separator |
| set_false_path         | update_macro               | get_property            |
| set_input_delay        | set_package_pin_val        | set_property            |
| set_output_delay       | デバッグ制約                     | set_units               |
| set_max_delay          | create_debug_core          | endgroup                |
| set_min_delay          | create_debug_port          | startgroup              |
| set_multicycle_path    | connect_debug_port         | create_property         |
| set_case_analysis      |                            | current_design          |
| set_clock_sense        | 消費電力制約                     | ネットリスト制約                |
| set_clock_uncertainty  | set_power_opt              | set_load                |
| set_input_jitter       | set_switching_activity     | set_logic_dc            |
| set_max_time_borrow    | reset_switching_activity   | set_logic_one           |
| set_propagated_clock   | set_operating_conditions   | set_logic_zero          |
| set_system_jitter      | reset_operating_conditions | set_logic_unconnected   |
| set_external_delay     | 除外制約                       | make_diff_pair_ports    |
| set_bus_skew           | create_waiver              |                         |



表 A-1: XDC ファイルで使用可能なコマンド (続き)

| デバイス オブジェクトのクエリ       | タイミング オブジェクトのクエリ     | ネットリスト オブジェクトのクエリ |
|-----------------------|----------------------|-------------------|
| get_iobanks           | all_clocks           | all_cpus          |
| get_package_pins      | get_path_groups      | all_dsps          |
| get_sites             | get_clocks           | all_fanin         |
| get_bel_pins          | get_generated_clocks | all_fanout        |
| get_bels              | get_timing_arcs      | all_hsios         |
| get_nodes             | get_speed_models     | all_inputs        |
| get_pips              | フロアプラン オブジェクトのクエリ    | all_outputs       |
| get_site_pins         | get_pblocks          | all_rams          |
| get_site_pips         | get_macros           | all_registers     |
| get_slrs              |                      | all_ffs           |
| get_tiles             |                      | all_latches       |
| get_wires             |                      | get_cells         |
| get_pkgpin_bytegroups |                      | get_nets          |
| get_pkgpin_nibbles    |                      | get_pins          |
|                       |                      | get_ports         |
|                       |                      | get_debug_cores   |
|                       |                      | get_debug_ports   |

## サポートされる SDC コマンド

注記: -quiet および -verbose オプションは、すべてのザイリンクス Tcl コマンドでサポートされるので、次の表にはリストしていません。

表 A-2: サポートされる SDC コマンド

| SDC 1.9                                | ザイリンクス SDC                             | 注記                                                                                               |
|----------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------|
| current_instance<br>[instance_name]    | current_instance<br>[instance_name]    | read_xdc -cells/-ref または SCOPED_TO_xxx 制約ファイル プロパティが使用される場合、Vivado IDE での get_ports の処理方法は異なります。 |
| expr                                   | expr                                   |                                                                                                  |
| list                                   | list                                   | Vivado IDE では、Tcl リストはオブジェクトのコンテナとしても使用されます。                                                     |
| set                                    | set                                    |                                                                                                  |
| set_hierarchy_separator<br>[separator] | set_hierarchy_separator<br>[separator] |                                                                                                  |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                           | 注記                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| <code>set_units</code><br><code>[-capacitance cap_units]</code><br><code>[-resistance res_unit]</code><br><code>[-time time_unit]</code><br><code>[-voltage voltage_units]</code><br><code>[-current current_unit]</code><br><code>[-power power_unit]</code>                                                                                                                                                                                                              | <code>set_units</code><br><code>[-capacitance arg]</code><br><code>[-resistance arg]</code><br><code>[-time arg]</code><br><code>[-voltage arg]</code><br><code>[-current arg]</code><br><code>[-power arg]</code><br><code>[-suffix arg]</code><br><code>[-digits arg]</code>                                                                                                       | <code>set_units -time</code> で Vivado IDE のタイミングの単位を変更することはできません。                               |
| <code>all_clocks</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <code>all_clocks</code>                                                                                                                                                                                                                                                                                                                                                              |                                                                                                 |
| <code>all_inputs</code><br><code>[-level_sensitive]</code><br><code>[-edge_triggered]</code><br><code>[-clock clock_name]</code>                                                                                                                                                                                                                                                                                                                                           | <code>all_inputs</code>                                                                                                                                                                                                                                                                                                                                                              |                                                                                                 |
| <code>all_outputs</code><br><code>[-level_sensitive]</code><br><code>[-edge_triggered]</code><br><code>[-clock clock_name]</code>                                                                                                                                                                                                                                                                                                                                          | <code>all_outputs</code>                                                                                                                                                                                                                                                                                                                                                             |                                                                                                 |
| <code>all_registers</code><br><code>[-no_hierarchy]</code><br><code>[-clock clock_name]</code><br><code>[-rise_clock clock_name]</code><br><code>[-fall_clock clock_name]</code><br><code>[-cells]</code><br><code>[-data_pins]</code><br><code>[-clock_pins]</code><br><code>[-slave_clock_pins]</code><br><code>[-async_pins]</code><br><code>[-output_pins]</code><br><code>[-level_sensitive]</code><br><code>[-edge_triggered]</code><br><code>[-master_slave]</code> | <code>all_registers</code><br><code>[-no_hierarchy]</code><br><code>[-clock args]</code><br><code>[-rise_clock args]</code><br><code>[-fall_clock args]</code><br><code>[-cells]</code><br><code>[-data_pins]</code><br><code>[-clock_pins]</code><br><code>[-async_pins]</code><br><code>[-output_pins]</code><br><code>[-level_sensitive]</code><br><code>[-edge_triggered]</code> |                                                                                                 |
| <code>current_design</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                | <code>current_design</code>                                                                                                                                                                                                                                                                                                                                                          | Vivado IDE では、現在のデザインはメモリに読み込まれているデザインを指しており、最上位モジュールまたはエンティティ以外の別のモジュールまたはエンティティに変更することはできません。 |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                 | ザイリンクス SDC                                                                                                                                                                                                                                                          | 注記                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <code>get_cells</code><br><code>[-hierarchical]</code><br><code>[-hsc separator]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>-of_objects objects</code><br><code>patterns</code> | <code>get_cells</code><br><code>[-hierarchical]</code><br><code>[-hsc arg]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[-of_objects args]</code><br><code>[patterns]</code><br><code>[-filter arg]</code><br><code>[-match_style arg]</code> |                                                                                         |
| <code>get_clocks</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code>                                                                                                     | <code>get_clocks</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[patterns]</code><br><code>[-filter arg]</code><br><code>[-of_objects args]</code><br><code>[-match_style arg]</code><br><code>[-include_generated_clocks]</code>               | Vivado IDE では、クロック ツリー上のクロック オブジェクトをクエリするために <code>-of_objects</code> オプションがサポートされます。   |
| <code>get_lib_cells</code><br><code>[-hsc separator]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code>                                                                 | <code>get_lib_cells</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code><br><code>[-filter arg]</code><br><code>[-include_unsupported]</code><br><code>[-of_objects args]</code>                                                      | Vivado IDE では、デザインに読み込むことができるのは 1 つのデバイス ライブラリのみなので、ライブラリ セルをクエリする際にライブラリを指定する必要はありません。 |
| <code>get_lib_pins</code><br><code>[-hsc separator]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code>                                                                  | <code>get_lib_pins</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code><br><code>[-filter arg]</code><br><code>[-of_objects args]</code>                                                                                              |                                                                                         |
| <code>get_libs</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code>                                                                                                       | <code>get_libs</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[patterns]</code><br><code>[-filter arg]</code>                                                                                                                                   |                                                                                         |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                               | 注記 |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| <code>get_nets</code><br><code>[-hierarchical]</code><br><code>[-hsc separator]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>-of_objects objects</code><br><code>patterns</code> | <code>get_nets</code><br><code>[-hierarchical]</code><br><code>[-hsc arg]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[-of_objects args]</code><br><code>[patterns]</code><br><code>[-filter arg]</code><br><code>[-match_style arg]</code><br><br><code>[-top_net_of_hierarchical_group]</code><br><code>[-segments]</code><br><code>[-boundary_type arg]</code> |    |
| <code>get_pins</code><br><code>[-hierarchical]</code><br><code>[-hsc separator]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>-of_objects objects</code><br><code>patterns</code> | <code>get_pins</code><br><code>[-hierarchical]</code><br><code>[-hsc arg]</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[-of_objects args]</code><br><code>[patterns]</code><br><code>[-leaf]</code><br><code>[-filter arg]</code><br><code>[-match_style arg]</code>                                                                                               |    |
| <code>get_ports</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>patterns</code>                                                                                                     | <code>get_ports</code><br><code>[-regex]</code><br><code>[-nocase]</code><br><code>[patterns]</code><br><code>[-filter arg]</code><br><code>[-of_objects args]</code><br><code>[-match_style arg]</code>                                                                                                                                                                                 |    |
| <code>create_clock</code><br><code>-period period_value</code><br><code>[-name clock_name]</code><br><code>[-waveform edge_list]</code><br><code>[-add]</code><br><code>[source_objects]</code>        | <code>create_clock</code><br><code>-period arg</code><br><code>[-name arg]</code><br><code>[-waveform args]</code><br><code>[-add]</code><br><code>[objects]</code>                                                                                                                                                                                                                      |    |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                  | ザイリンクス SDC                                                                                                                                                                                                                    | 注記 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| <pre>create_generated_clock   [-name clock_name]   -source master_pin   [-edges edge_list]   [-divide_by factor]   [-multiply_by factor]   [-duty_cycle percent]   [-invert]   [-edge_shift shift_list]   [-add]   [-master_clock clock]   [-combinational]   source_objects</pre>                       | <pre>create_generated_clock   [-name arg]   [-source args]   [-edges args]   [-divide_by arg]   [-multiply_by arg]   [-duty_cycle arg]   [-edge_shift args]   [-add]   [-master_clock arg]   [-combinational]   objects</pre> |    |
| <pre>group_path   [-name group_name]   [-default]   [-weight weight_value]   [-from from_list]   [-rise_from from_list]   [-fall_from from_list]   [-to to_list]   [-rise_to to_list]   [-fall_to to_list]   [-through through_list]   [-rise_through through_list]   [-fall_through through_list]</pre> | <pre>group_path   [-name arg]   [-weight 1 2]   [-from args]   [-to args]   [-through args]</pre>                                                                                                                             |    |
| <pre>set_clock_groups   [-name name]   [-logically_exclusive]   [-physically_exclusive]   [-asynchronous]   [-allow_paths]   -group   clock_list</pre>                                                                                                                                                   | <pre>set_clock_groups   [-name arg]   [-logically_exclusive]   [-physically_exclusive]   [-asynchronous]   [-group args]</pre>                                                                                                |    |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                                                     | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                | 注記 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| <code>set_clock_latency</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-min]</code><br><code>[-max]</code><br><code>[-source]</code><br><code>[-late]</code><br><code>[-early]</code><br><code>[-clock clock_list]</code><br><code>delay</code><br><code>object_list</code>                                                                                                                                               | <code>set_clock_latency</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-min]</code><br><code>[-max]</code><br><code>[-source]</code><br><code>[-late]</code><br><code>[-early]</code><br><code>[-clock args]</code><br><code>latency</code><br><code>objects</code>                                                     |    |
| <code>set_clock_sense</code><br><code>[-positive]</code><br><code>[-negative]</code><br><code>[-pulse pulse]</code><br><code>[-stop_propagation]</code><br><code>[-clock clock_list]</code><br><code>pin_list</code>                                                                                                                                                                                                                        | <code>set_clock_sense</code><br><code>[-positive]</code><br><code>[-negative]</code><br><code>[-pulse arg]</code><br><code>[-stop_propagation]</code><br><code>[-clocks args]</code><br><code>pins</code>                                                                                                                                 |    |
| <code>set_clock_uncertainty</code><br><code>[-from from_clock]</code><br><code>[-rise_from rise_from_clock]</code><br><code>[-fall_from fall_from_clock]</code><br><code>[-to to_clock]</code><br><code>[-rise_to rise_to_clock]</code><br><code>[-fall_to fall_to_clock]</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-setup]</code><br><code>[-hold]</code><br><code>uncertainty</code><br><code>[object_list]</code> | <code>set_clock_uncertainty</code><br><code>[-from args]</code><br><code>[-rise_from args]</code><br><code>[-fall_from args]</code><br><code>[-to args]</code><br><code>[-rise_to args]</code><br><code>[-fall_to args]</code><br><br><code>[-setup]</code><br><code>[-hold]</code><br><code>uncertainty</code><br><code>[objects]</code> |    |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                      | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                            | 注記 |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| set_data_check<br>[-from <i>from_object</i> ]<br>[-to <i>to_object</i> ]<br>[-rise_from <i>from_object</i> ]<br>[-fall_from <i>from_object</i> ]<br>[-rise_to <i>to_object</i> ]<br>[-fall_to <i>to_object</i> ]<br>[-setup]<br>[-hold]<br>[-clock <i>clock_object</i> ]<br><i>value</i>                                                                                                                     | set_data_check<br>[-from <i>args</i> ]<br>[-to <i>args</i> ]<br>[-rise_from <i>args</i> ]<br>[-fall_from <i>args</i> ]<br>[-rise_to <i>args</i> ]<br>[-fall_to <i>args</i> ]<br>[-setup]<br>[-hold]<br>[-clock <i>args</i> ]<br><i>value</i>                                                                                          |    |
| set_disable_timing<br>[-from <i>from_pin_name</i> ]<br>[-to <i>to_pin_name</i> ]<br><i>cell_pin_list</i>                                                                                                                                                                                                                                                                                                     | set_disable_timing<br>[-from <i>arg</i> ]<br>[-to <i>arg</i> ]<br><i>objects</i>                                                                                                                                                                                                                                                      |    |
| set_false_path<br>[-setup]<br>[-hold]<br>[-rise]<br>[-fall]<br>[-from <i>from_list</i> ]<br>[-to <i>to_list</i> ]<br>[-through <i>through_list</i> ]<br>[-rise_from <i>rise_from_list</i> ]<br>[-rise_to <i>rise_to_list</i> ]<br>[-rise_through<br><i>rise_through_list</i> ]<br>[-fall_from <i>fall_from_list</i> ]<br>[-fall_to <i>fall_to_list</i> ]<br>[-fall_through<br><i>fall_through_list</i> ]<br> | set_false_path<br>[-setup]<br>[-hold]<br>[-rise]<br>[-fall]<br>[-from <i>args</i> ]<br>[-to <i>args</i> ]<br>[-through <i>args</i> ]<br>[-rise_from <i>args</i> ]<br>[-rise_to <i>args</i> ]<br>[-rise_through <i>args</i> ]<br>[-fall_from <i>args</i> ]<br>[-fall_to <i>args</i> ]<br>[-fall_through <i>args</i> ]<br>[-reset_path] |    |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 注記                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| <code>set_input_delay</code><br><code>[-clock clock_name]</code><br><code>[-clock_fall]</code><br><code>[-level_sensitive]</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-max]</code><br><code>[-min]</code><br><code>[-add_delay]</code><br><code>[-network_latency_included]</code><br><code>[-source_latency_included]</code><br><code>delay_value</code><br><code>port_pin_list</code>                                                                                                              | <code>set_input_delay</code><br><code>[-clock args]</code><br><code>[-clock_fall]</code><br><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-max]</code><br><code>[-min]</code><br><code>[-add_delay]</code><br><code>[-network_latency_included]</code><br><code>[-source_latency_included]</code><br><code>delay</code><br><code>objects</code><br><code>[-reference_pin args]</code>                                                                            | Vivado IDE では、内部ピンでは入力遅延はサポートされません。 |
| <code>set_max_delay</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-from from_list]</code><br><code>[-to to_list]</code><br><code>[-through through_list]</code><br><code>[-rise_from rise_from_list]</code><br><code>[-rise_to rise_to_list]</code><br><code>[-rise_through</code><br><code>rise_through_list]</code><br><code>[-fall_from fall_from_list]</code><br><code>[-fall_to fall_to_list]</code><br><code>[-fall_through</code><br><code>fall_through_list]</code><br><code>delay_value</code> | <code>set_max_delay</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-from args]</code><br><code>[-to args]</code><br><code>[-through args]</code><br><code>[-rise_from args]</code><br><code>[-rise_to args]</code><br><code>[-rise_through args]</code><br><br><code>[-fall_from args]</code><br><code>[-fall_to args]</code><br><code>[-fall_through args]</code><br><br><code>delay</code><br><code>[-reset_path]</code><br><code>[-datapath_only]</code> |                                     |
| <code>set_max_time_borrow</code><br><code>delay_value</code><br><code>object_list</code>                                                                                                                                                                                                                                                                                                                                                                                                                                   | <code>set_max_time_borrow</code><br><code>delay</code><br><code>objects</code>                                                                                                                                                                                                                                                                                                                                                                                                |                                     |



表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                                     | 注記 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| set_min_delay<br>[-rise]<br>[-fall]<br>[-from <i>from_list</i> ]<br>[-to <i>to_list</i> ]<br>[-through <i>through_list</i> ]<br>[-rise_from <i>rise_from_list</i> ]<br>[-rise_to <i>rise_to_list</i> ]<br>[-rise_through<br><i>rise_through_list</i> ]<br>[-fall_from <i>fall_from_list</i> ]<br>[-fall_to <i>fall_to_list</i> ]<br>[-fall_through<br><i>fall_through_list</i> ]<br><i>delay_value</i>                                                        | set_min_delay<br>[-rise]<br>[-fall]<br>[-from <i>args</i> ]<br>[-to <i>args</i> ]<br>[-through <i>args</i> ]<br>[-rise_from <i>args</i> ]<br>[-rise_to <i>args</i> ]<br>[-rise_through <i>args</i> ]<br>[-fall_to <i>args</i> ]<br>[-fall_from <i>args</i> ]<br>[-fall_through <i>args</i> ]<br><br><i>delay</i><br>[-reset_path]                                                              |    |
| set_multicycle_path<br>[-setup]<br>[-hold]<br>[-rise]<br>[-fall]<br>[-start]<br>[-end]<br>[-from <i>from_list</i> ]<br>[-to <i>to_list</i> ]<br>[-through <i>through_list</i> ]<br>[-rise_from <i>rise_from_list</i> ]<br>[-rise_to <i>rise_to_list</i> ]<br>[-rise_through<br><i>rise_through_list</i> ]<br>[-fall_from <i>fall_from_list</i> ]<br>[-fall_to <i>fall_to_list</i> ]<br>[-fall_through<br><i>fall_through_list</i> ]<br><i>path_multiplier</i> | set_multicycle_path<br>[-setup]<br>[-hold]<br>[-rise]<br>[-fall]<br>[-start]<br>[-end]<br>[-from <i>args</i> ]<br>[-to <i>args</i> ]<br>[-through <i>args</i> ]<br>[-rise_from <i>args</i> ]<br>[-rise_to <i>args</i> ]<br>[-rise_through <i>args</i> ]<br>[-fall_from <i>args</i> ]<br>[-fall_to <i>args</i> ]<br>[-fall_through <i>args</i> ]<br><br><i>path_multiplier</i><br>[-reset_path] |    |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                                                                        | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                                          | 注記                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| <code>set_output_delay</code><br><code>[-clock clock_name]</code><br><code>[-clock_fall]</code><br><code>[-level_sensitive]</code><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-max]</code><br><code>[-min]</code><br><code>[-add_delay]</code><br><code>[-network_latency_included]</code><br><code>[-source_latency_included]</code><br><code>delay_value</code><br><code>port_pin_list</code> | <code>set_output_delay</code><br><code>[-clock args]</code><br><code>[-clock_fall]</code><br><br><code>[-rise]</code><br><code>[-fall]</code><br><code>[-max]</code><br><code>[-min]</code><br><code>[-add_delay]</code><br><code>[-network_latency_included]</code><br><code>[-source_latency_included]</code><br><code>delay</code><br><code>objects</code><br><code>[-reference_pin args]</code> | Vivado IDE では、内部ピンでは出力遅延はサポートされません。                        |
| <code>set_propagated_clock</code><br><code>object_list</code>                                                                                                                                                                                                                                                                                                                                                  | <code>set_propagated_clock</code><br><code>object</code>                                                                                                                                                                                                                                                                                                                                            | Vivado IDE では、すべてのクロックはデフォルトで伝搬クロックとして処理されます。              |
| <code>set_case_analysis</code><br><code>value</code><br><code>port_or_pin_list</code>                                                                                                                                                                                                                                                                                                                          | <code>set_case_analysis</code><br><code>value</code><br><code>objects</code>                                                                                                                                                                                                                                                                                                                        |                                                            |
| <code>set_load</code><br><code>[-min]</code><br><code>[-max]</code><br><code>[-subtract_pin_load]</code><br><code>[-pin_load]</code><br><code>[-wire_load]</code><br><code>value</code><br><code>objects</code>                                                                                                                                                                                                | <code>set_load</code><br><code>[-max]</code><br><code>[-min]</code><br><br><code>capacitance</code><br><code>objects</code><br><code>[-rise]</code><br><code>[-fall]</code>                                                                                                                                                                                                                         | Vivado IDE では、 <code>set_load</code> コマンドは消費電力解析のみのコマンドです。 |
| <code>set_logic_dc</code><br><code>port_list</code>                                                                                                                                                                                                                                                                                                                                                            | <code>set_logic_dc</code><br><code>objects</code>                                                                                                                                                                                                                                                                                                                                                   |                                                            |
| <code>set_logic_one</code><br><code>port_list</code>                                                                                                                                                                                                                                                                                                                                                           | <code>set_logic_one</code><br><code>objects</code>                                                                                                                                                                                                                                                                                                                                                  |                                                            |

表 A-2: サポートされる SDC コマンド (続き)

| SDC 1.9                                                                                                                                                                                                                                                                                                                                                             | ザイリンクス SDC                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 注記                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>set_logic_zero</code><br><code>port_list</code>                                                                                                                                                                                                                                                                                                               | <code>set_logic_zero</code><br><code>objects</code>                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                 |
| <code>set_operating_conditions</code><br><code>[-library lib_name]</code><br><code>[-analysis_type</code><br><code>analysis_type]</code><br><code>[-max max_condition]</code><br><code>[-min min_condition]</code><br><code>[-max_library max_lib]</code><br><code>[-min_library min_lib]</code><br><code>[-object_list objects]</code><br><code>[condition]</code> | <code>set_operating_conditions</code><br><br><code>[-voltage args]</code><br><code>[-grade arg]</code><br><code>[-process arg]</code><br><code>[-junction_temp arg]</code><br><code>[-ambient_temp arg]</code><br><code>[-thetaja arg]</code><br><code>[-thetasa arg]</code><br><code>[-airflow arg]</code><br><code>[-heatsink arg]</code><br><code>[-thetajb arg]</code><br><code>[-board arg]</code><br><code>[-board_temp arg]</code><br><code>[-board_layers arg]</code> | <p>Vivado IDE で <code>set_operating_conditions</code> コマンドを使用すると、動作条件が消費電力解析のみに設定され、タイミング レポートには影響しません。Vivado IDE タイミング エンジン は、<code>config_timing_analysis</code> コマンドで制御されます。<code>config_timing_analysis</code> の詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) <a href="#">[参照 10]</a> を参照してください。</p> |

## サポートされない SDC コマンド

次の SDC コマンドはサポートされません。

- `set_clock_gating_check`
- `set_clock_transition`
- `set_ideal_latency`
- `set_ideal_network`
- `set_ideal_transition`
- `set_max_fanout`

注記: 合成中、最大ファンアウトは MAX\_FANOUT 属性により制御されます。

- `set_drive`
- `set_driving_cell`
- `set_fanout_load`
- `set_input_transition`
- `set_max_area`
- `set_max_capacitance`
- `set_max_transition`
- `set_min_capacitance`
- `set_port_fanout_number`
- `set_resistance`
- `set_timing_derate`
- `set_voltage`
- `set_wire_load_min_block_size`
- `set_wire_load_mode`
- `set_wire_load_model`
- `set_wire_load_selection_group`
- `create_voltage_area`
- `set_level_shifter_strategy`
- `set_level_shifter_threshold`
- `set_max_dynamic_power`
- `set_max_leakage_power`

# その他の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。

---

## Documentation Navigator およびデザイン ハブ

ザイリンクス Documentation Navigator (DocNav) では、ザイリンクスの資料、ビデオ、サポート リソースにアクセスでき、特定の情報を取得するためにフィルター機能や検索機能を利用できます。DocNav を開くには、次のいずれかを実行します。

- Vivado IDE で [Help] → [Documentation and Tutorials] をクリックします。
- Windows で [スタート] → [すべてのプログラム] → [Xilinx Design Tools] → [DocNav] をクリックします。
- Linux コマンド プロンプトに「docnav」と入力します。

ザイリンクス デザイン ハブには、資料やビデオへのリンクがデザイン タスクおよびトピックごとにまとめられており、これらを参照することでキー コンセプトを学び、よくある質問 (FAQ) を参考に問題を解決できます。デザイン ハブにアクセスするには、次のいずれかを実行します。

- DocNav で [Design Hubs View] タブをクリックします。
- ザイリンクス ウェブサイトの[デザイン ハブ](#) ページを参照します。

注記: DocNav の詳細は、ザイリンクス ウェブサイトの [Documentation Navigator](#) ページを参照してください。



注意: DocNav からは、日本語版は参照できません。ウェブサイトのデザイン ハブ ページをご利用ください。

## 参考資料

### Vivado Design Suite ユーザー ガイドおよびリファレンス ガイド

このガイドでは、次の Vivado® Design Suite ガイドが参照されています。

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. 『ISE から Vivado Design Suite への移行ガイド』(UG911)
2. 『Vivado Design Suite ユーザー ガイド: システム レベル デザイン入力』(UG895)
3. 『Vivado Design Suite ユーザー ガイド: I/O およびクロック プランニング』(UG899)
4. 『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)
5. 『UltraFast 設計手法ガイド (Vivado Design Suite 用)』(UG949)
6. 『AXI Quad SPI LogiCORE IP 製品ガイド』(PG153: 英語版、日本語版)
7. 『Vivado Design Suite ユーザー ガイド: Vivado IDE の使用』(UG893)
8. 『Vivado Design Suite ユーザー ガイド: 合成』(UG901)
9. 『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904)
10. 『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835)
11. 『Vivado Design Suite プロパティ リファレンス ガイド』(UG912)
12. 『Vivado Design Suite ユーザー ガイド: プログラムおよびデバッグ』(UG908)
13. 『7 シリーズ FPGA SelectIO リソース ユーザー ガイド』(UG471: 英語版、日本語版)

### その他のザイリンクス リソース

このガイドでは、次の資料も参照されています。

14. [ザイリンクス アンサー 59893](#)

## 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。

1. [トレーニング クラス: Vivado Design Suite を使用した FPGA の設計 1](#)
2. [トレーニング クラス: Vivado Design Suite を使用した FPGA の設計 2](#)
3. [トレーニング クラス: Vivado Design Suite を使用した FPGA の設計 3](#)
4. [トレーニング クラス: Vivado Design Suite を使用した FPGA の設計 4](#)
5. [Vivado Design Suite QuickTake ビデオ チュートリアル](#)
6. [Vivado Design Suite QuickTake ビデオ: Vivado タイミング制約ウィザードの使用](#)
7. [Vivado Design Suite QuickTake ビデオ: アドバンス クロック制約と解析](#)
8. [Vivado Design Suite QuickTake ビデオ: 入力遅延の設定](#)
9. [Vivado Design Suite QuickTake ビデオ: 出力遅延の設定](#)

## 10. Vivado Design Suite QuickTake ビデオ: UCF 制約の XDC への変換

## 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になります。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

オートモーティブ製品 (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ます。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す。

© Copyright 2012-2020 Xilinx, Inc. Xilinx、Xilinx のロゴ、Artix、ISE、Kintex、Spartan、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。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。

この資料に関するフィードバックおよびリンクなどの問題につきましては、[jpn\\_trans\\_feedback@xilinx.com](mailto:jpn_trans_feedback@xilinx.com) まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。