Vivado Design Suite チュートリアル

制約の使用

UG945 (v2012.3) 2012 年 10 月 16 日

XILINX®

本資料は英語版 (v2012.3) 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。

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 までお知らせください。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。

改訂履歴

日付	バージョン	改訂内容
2012 年 8 月 6 日	2012.2	初版
2012 年 8 月 20 日	2012.2	手順 4：クロック関係のレポート：生成されたクロックとプライマリ クロック (sysClk) の関係についての詳細を追加
2012 年 10 月 16 日	2012.3	リリースの確認

目次

Vivado チュートリアル : 制約の使用

概要

このチュートリアルは複数の演習から構成されており、ザイリンクス Vivado™ 統合設計環境 (IDE) でデザインに制約を設定する方法を学習します。Vivado の制約フォーマットは、XDC (Xilinx Design Constraints) と呼ばれ、業界標準の Synopsys® デザイン制約とザイリンクス制約を組み合わせたものです。

XDC は単なる文字列ではなく、Tcl コマンドです。ほかの Tcl コマンドと同じように、Vivado Tcl インタープリタにより順次読み出され、処理されます。デザイン制約はフローのさまざまな段階でいろいろな方法で入力できます。XDC を 1 つまたは複数のファイルに保存し、それを Vivado のプロジェクト モードで制約セットに追加したり、非プロジェクト モードで read_xdc コマンドを使用し、メモリに同じファイルを直接読み出すことができます。プロジェクト モードおよび非プロジェクト モードの詳細は、『Vivado Design Suite ユーザー ガイド : デザイン フローの概要』(UG892) を参照してください。Vivado でファイルを開き、Tcl コンソール、または Tcl コマンド プロンプトに直接コマンドとして制約を入力することもできます。これは、デザインの新しい制約を定義、検証、デバッグするにあたり特に強力な機能です。

Vivado Design Suite の合成およびインプリメンテーション ツールはタイミング ドリブンで、デザイン目標を満たし、正しい動作を得るためには、正確で正しいタイミング制約を設定することが重要となります。Vivado ツールはタイミング ドリブンであるため、適切にデザインに制約を設定することが大切です。厳しく設定しすぎると、デザイン目標が非現実的となり、ツールのランタイムが長くなり、最適とは言えない結果となる可能性があります。逆に制約が緩いと、マルチサイクル遅延のあるパスやフォルス パスの解析など、不必要な最適化が行われ、実際のクリティカル パスの解析がおざなりになってしまうこともあります。

このチュートリアルでは、デザイン制約の定義および適用方法について学び、次のような演習が含まれています。

- 演習 1 : [Timing Constraints] ビューの使用
- 演習 2 : 物理制約の設定

ソフトウェア要件

このチュートリアルには、Vivado Design Suite 2012.3 またはそれ以降のバージョンのものが必要です。

ハードウェア要件

サポートされているオペレーティング システムは、Redhat 5.6 Linux 64、Windows (64 および 32 ビット) です。

Vivado ツールを使用するには、2GB 以上の RAM が推奨されます。

チュートリアル デザインの説明

このチュートリアルでは、project_cpu_netlist という小型デザインを含むサンプル デザインを使用します。最上位 EDIF ネットリスト ソース ファイルと、XDC 制約ファイルがあります。

このデザインは、XC7K70T デバイスをターゲットにしています。小型デザインを使用することで、チュートリアルを最小ハードウェア要件で実行し、時間内に完了させることができるだけでなく、データ サイズも小さくすることができます。

チュートリアル デザイン ファイルのディレクトリ

このチュートリアルのファイルは Vivado Design Suite 次のディレクトリにあります。

- ***<Xilinx_2012.3_install_area>*/Vivado/<version>/examples/Vivado_Tutorial**

また、ローカル ディレクトリにチュートリアル ファイルを保存したり、変更を加える前の元の状態にファイルを戻すため、提供されている ZIP ファイルを随時抽出することもできます。

書き込み権のあるディレクトリに ZIP ファイルを解凍します。

- ***<Xilinx_2012.3_install_area>*/Vivado/<version>/examples/Vivado_Tutorial.zip**

抽出された Vivado_Tutorial ディレクトリは、このチュートリアルでは「<*Extract_Dir*>」と表記されます。

注記 : このチュートリアルの演習では、チュートリアル デザイン データを変更します。演習を開始するときは常に Vivado_Tutorial のコピーを使用してください。

演習 1 : [Timing Constraints] ビューの使用

この演習では、デザインの制約作成方法を 2 つ学びます。Vivado IDE に含まれている CPU ネットリスト サンプル デザインを使用します。

手順 1： サンプル プロジェクトを開く

Vivado IDE を起動します。

- Windows デスクトップにある Vivado IDE のアイコンをクリックします。
- コマンド ターミナルで「vivado」と入力します。

Getting Started ページで [Open Example Project] をクリックし、[CPU (Synthesized)] デザインを選択します。

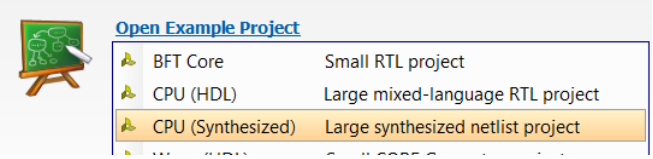


図 1： サンプル デザインを開く

元のサンプル デザインを上書きしないように、プロジェクトを別のディレクトリに保存します。

[File] → [Save Project As] をクリックし、プロジェクト名およびディレクトリを指定します。

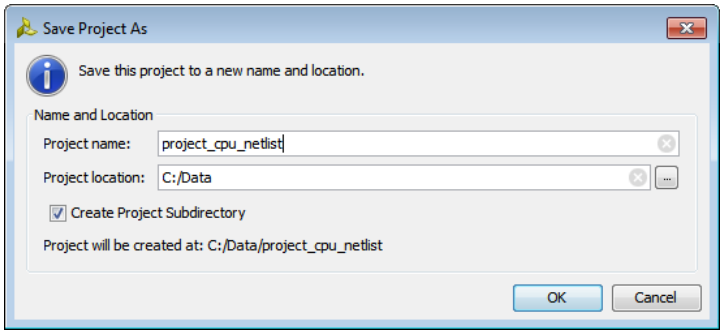


図 2 : [Save Design As] ダイアログ ボックス

プロジェクトが指定ディレクトリに保存されました。

Vivado IDE の [Project Summary] ビューにプロジェクト情報が表示されます。これはネットリスト プロジェクトなので、Flow Navigator には合成のオプションはありません。[Project Summary] ビューに、デザイン フローの次のステップである Vivado インプリメンテーションが表示されています。

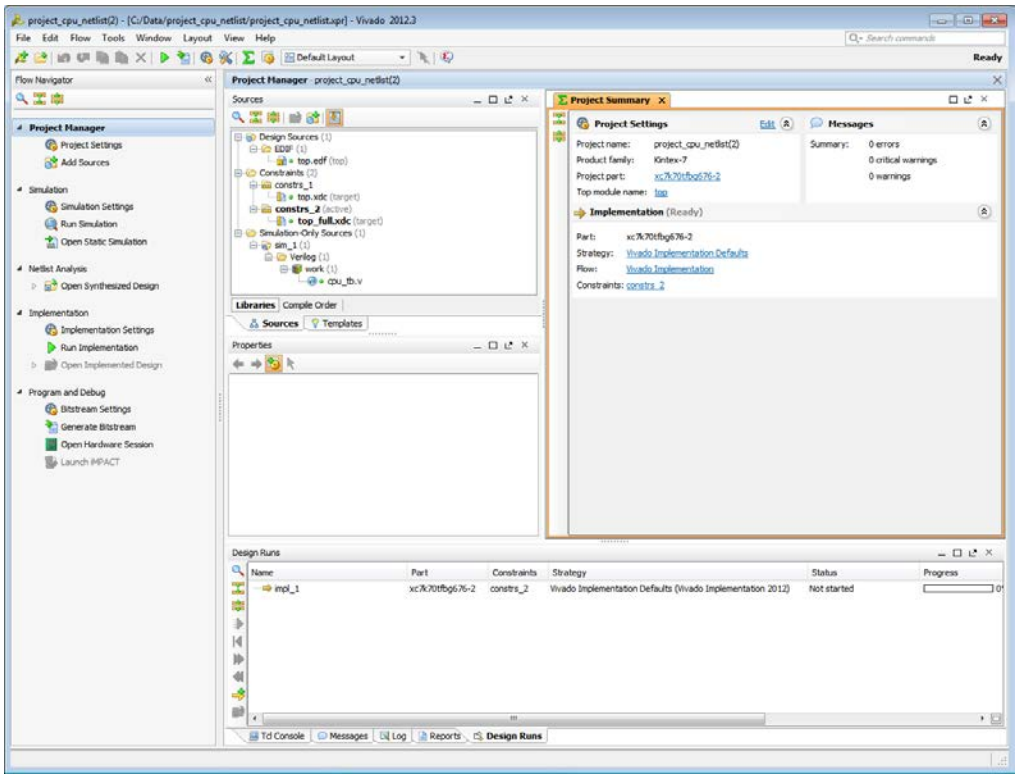


図 3 : [Project Summary] ビュー

手順 2： 制約セットおよびファイルの定義

重要 : Vivado Design Suite では、UCF フォーマットはサポートされません。UCF 制約を XDC コマンドに移行する方法は、『Vivado Design Suite 移行手法ガイド 』(UG911) を参照してください。

新しい制約セットを作成し、それを XDC 制約ファイルに追加するところから始めます。サンプル デザインには既に 2 つの制約セットが含まれていますが、この演習では使用しません。

1. Flow Navigator の [Project Manager] の下にある [Add Sources] をクリックします。
2. [Add Sources] ダイアログ ボックスに表示されているリストから [Add or Create Constraints] を選択し、[Next] をクリックします。
3. [Add or Create Constraints] ダイアログ ボックスの [Specify Constraint Set] ドロップダウン メニューから [Create Constraint Set] を次の図のように選択します。

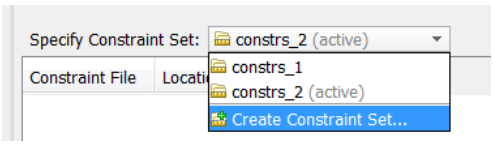


図 4： 制約セットの作成

4. [Create Constraint Set Name] ダイアログ ボックスの [Specify Constraint Set] で「lab1」を指定して [OK] をクリックします。
5. [Make active] チェックボックスをオンにします。
6. プロジェクトに新しい XDC ファイルを追加するため [Create File] を選択します。[File name] に「timing」と入力し、[File location] は <Local to Project> のままにしておき、[OK] をクリックします。

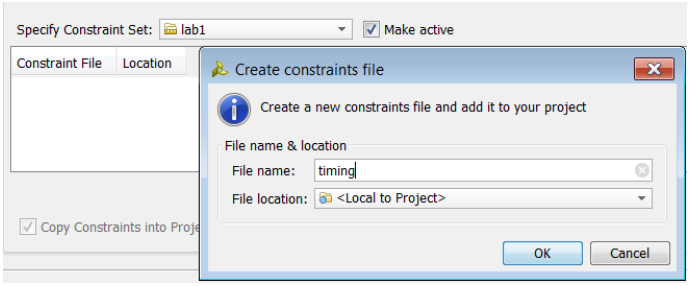


図 5 : 制約ファイル名

lab1 制約セットに timing.xdc ファイルが追加されます。

7. [Finish] をクリックして、新しい制約セットおよび XDC ファイルの作成を完了させます。

次の図にあるように、[Sources] ビューに新しい制約セットと XDC ファイルが表示されているはずです。制約セットは、作成時に指定したようにアクティブになっています。

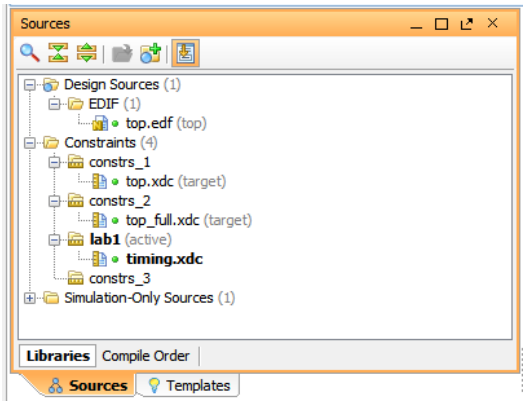


図 6 : [Sources] ビュー

手順 3 : [Timing Constraints] ビューの使用

合成されたデザインを開き、新しいタイミング制約をいくつか作成します。タイミング解析と、タイミング ドリブンの配置配線を実行するには、クロックが 1 つ以上必要なため、このデザインに対しクロックを作成します。

1. Flow Navigator で [Open Synthesized Design] をクリックします。

   合成されたネットリストが開き、[Device] ビューに表示されます。

2. Flow Navigator で [Netlist Analysis] → [Edit Timing Constraints] をクリックします。Vivado IDE で [Timing Constraints] ビューが開きます。

   このビューには 3 つのセクションがあります。

   - **制約のツリー表示** : 図 7 : [Timing Constraints] ビュー に示すように、[Timing Constraints] ビューの左上にあります。カテゴリ別にまとめられた標準タイミング制約がこのセクションに表示されます。このセクションで制約名をダブルクリックすると、制約ウィザードが開き、選択した制約を定義できるようになっています。
   - **制約の表** : 図 7 の右上にあります。このセクションには、ツリー表示で選択されているタイプのタイミング制約が表形式で表示されます。制約ウィザードを使用せずに、直接制約を定義したり編集する場合は、これを使用します。
   - **すべての制約** : [Timing Constraints] ビューの下部にあり、デザインで定義されているタイミング制約がすべて表示されます。

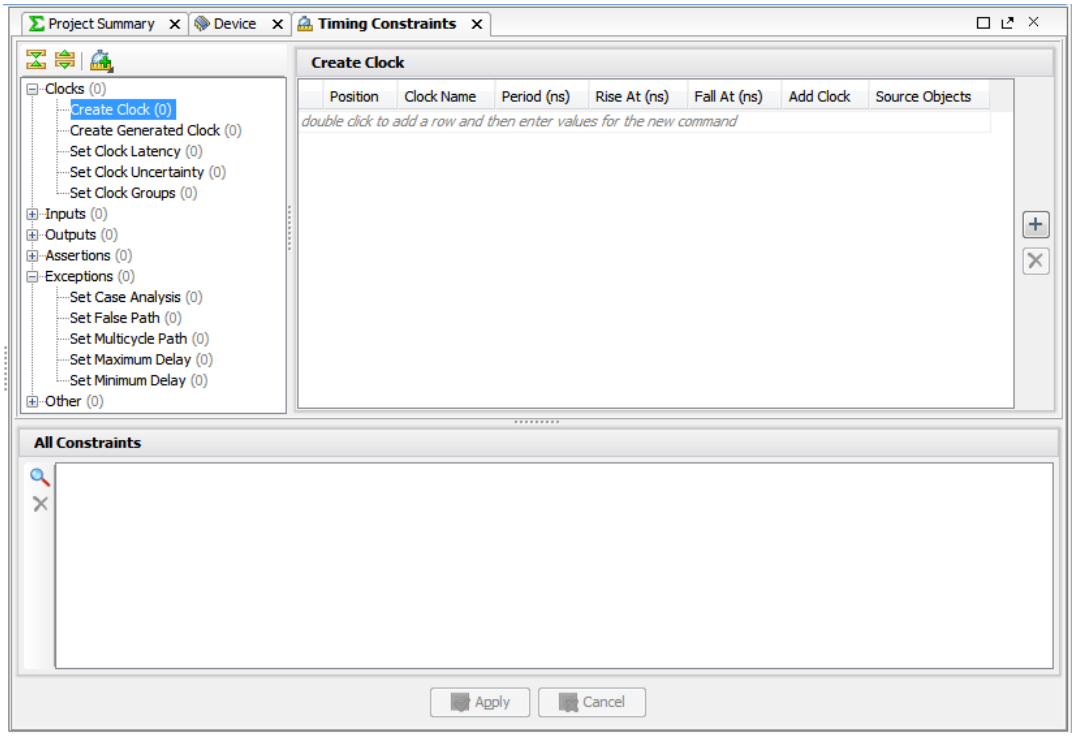


図 7 : [Timing Constraints] ビュー

3. ツリー表示で [Clocks] の下にある [Create Clock] をダブルクリックします。図のように Create Clock ウィザードが開きます。

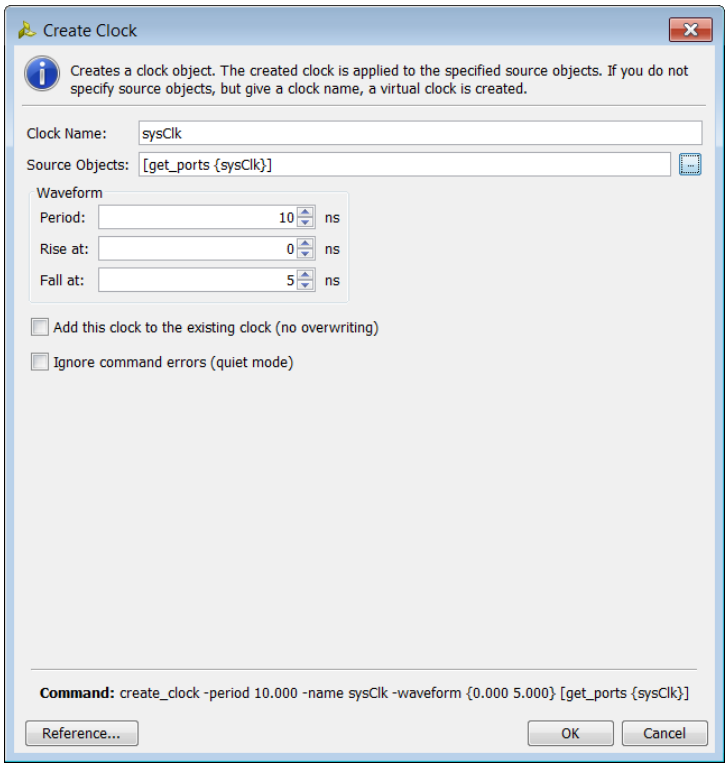


図 8 : Create Clocks ウィザード

a. [Clokc Name] に「sysClk」と入力します。

クロック名には任意の名前を指定でき、デザインのエレメント (ポートやピン) に一致させる必要はありません。ただし、通常、プライマリ クロックの名前はその入力ポートの名前と同じです。

b. [Source Objects] フィールドの横にある参照ボタン ( ) をクリックし、図 9 : [Specify Clock Sources Objects] ダイアログ ボックス にあるように [Specify Clock Sources Objects] を開きます。

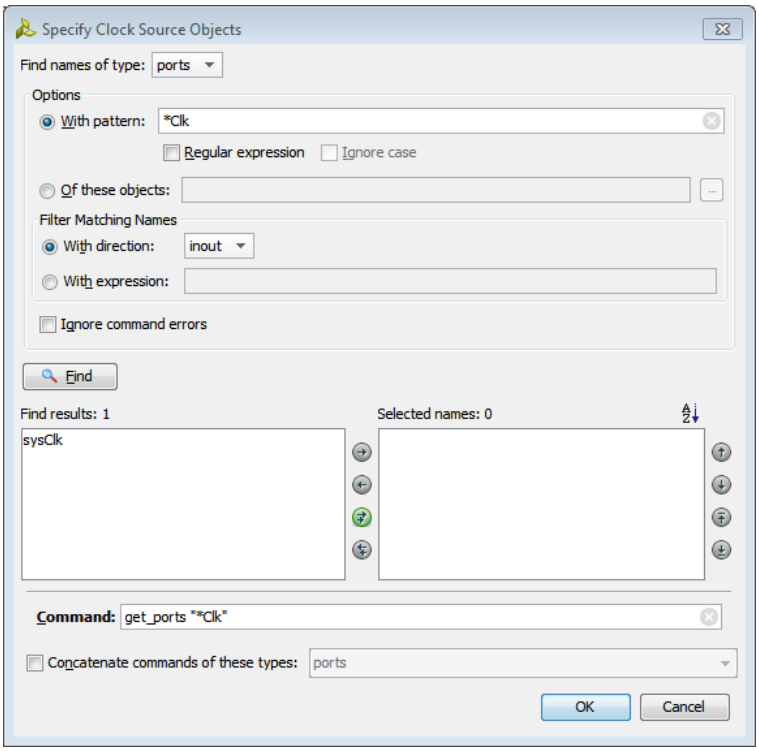


図 9 : [Specify Clock Sources Objects] ダイアログ ボックス

c. [Find names of type] で [ports] を選択します。
d. [With pattern] でアスタリスク (*) に「Clk」を追加し、[Find] ボタンをクリックします。

   [Find results] に「sysClk」が表示されます。
e. この [sysClk] を選択し、緑色の右矢印をクリックして、[Selected names] に移動させます。

ヒント : [sysClk] をダブルクリックして [Find results] から [Selected names] に移動させることもできます。

このダイアログ ボックスの下部にある [Command] フィールドに表示されている文字が変わります。「get_ports」というコマンドが次のように変わります。

```
get_ports {sysClk}
```

f. [OK] をクリックして、クロック ソースの指定を終了し、Create Clock ウィザードに戻ります。

   ウィザードは 図 8 : Create Clocks ウィザードのように表示されているはずです。[Waveform] セクションにあるデフォルト値をそのまま使用します。つまり、10ns の周期、50% のデューティ サイクルです。これらの値は、上下矢印をクリックするか、直接値を入力して適宜変更できます。[Command] フィールドは次のようになっているはずです。

```
create_clock –period 10.000 –name sysClk –waveform {0.000 5.000} [get_ports {sysClk}]
```

Vivado IDE では、デザイン ウィザードで作成されたすべての制約が Tcl コマンドとして表示されます。Tcl コマンド構文を学んだり、追加前に最終的な制約を確認するのに便利です。

g. [OK] をクリックして Create Clock ウィザードを終了し、図 10 : 追加された sysClk 制約にあるように sysClk のクロック制約が作成されます。

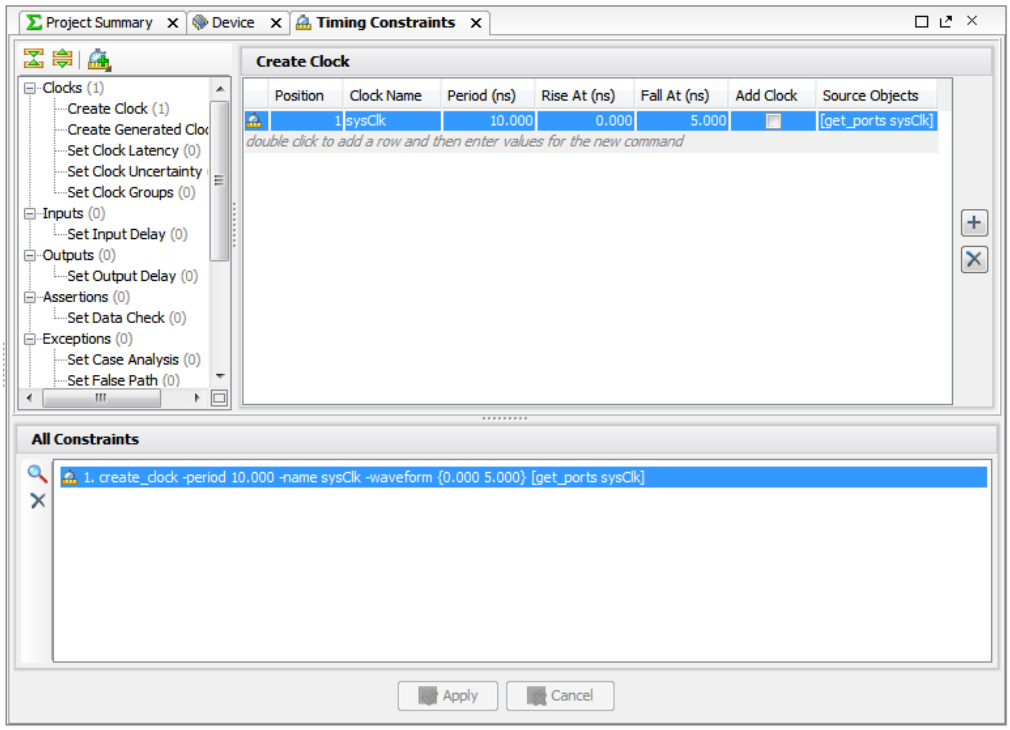


図 10 : 追加された sysClk 制約

ツリー表示のセクションの [Create Clock] に制約が 1 つ追加され、デザインにクロックが 1 つあることが確認できます。作成した sysClk のプロパティは、[Timing Constraints] ビューのほかのセクションでも確認できます。

このタイミング制約はインメモリ デザインに追加されていますが、timing.xdc 制約ファイルには書き込まれていません。新しい制約をこのファイルに書き込むには、[Save Constraints] コマンドを使用する必要があります。

重要 : 制約を timing.xdc ファイルに書き込むには、[Save Constraints] コマンドを使用する必要があります。

手順 4： クロック関係のレポート

先ほどの手順で定義した sysClk も含め、ここではデザインにあるさまざまなクロックの関係をレポートを確認します。

1. Flow Navigator で [Netlist Analysis] → [Report Clock Interaction] をクリックし、[Report clock Interaction] ダイアログ ボックスで [OK] をクリックし、デフォルト値を使用します。

   Vivado IDE では、デザインのさまざまなクロックの関係を示すグラフィカルなマトリックスが生成されます。このデザインの場合、プライマリ クロック (sysClk) は、6 つの追加クロックを生成する MMCM に接続されています。この図は、これらの生成されたクロックの関係を示しています。sysClk とほかのクロックとの間には関連がないため、ここでは sysClk は表示されていません。つまり、別クロックで取り込まれる信号を sysClk が出力することはなく、また sysClk が別クロックで出力された信号を取り込むことはありません。

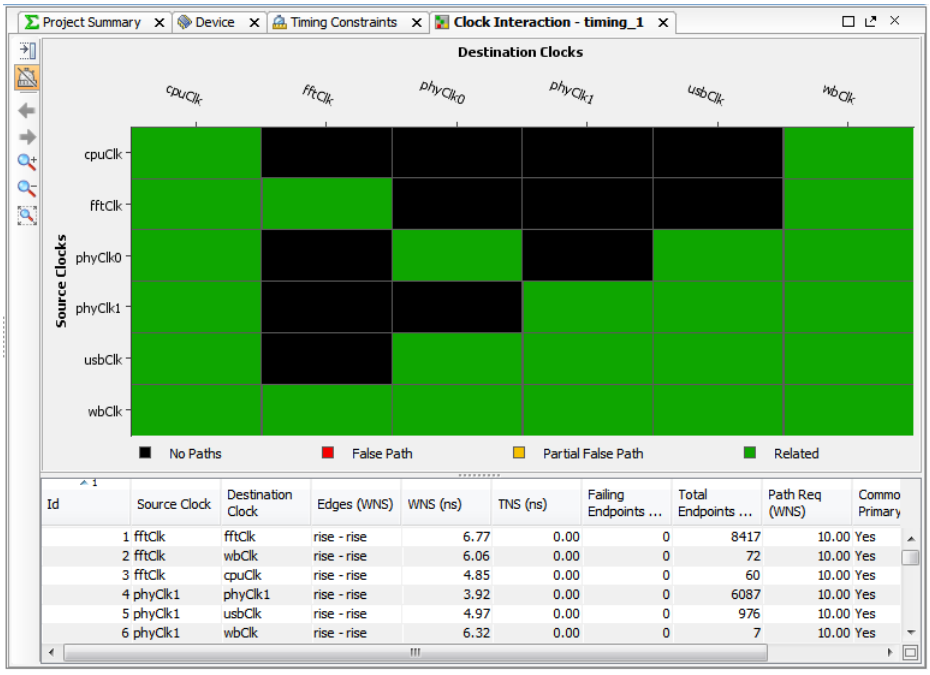
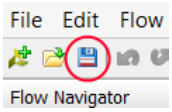


図 11： クロック関係のレポート

手順 5： 制約の保存

このデザインに対しプライマリ クロックを作成しましたが、その制約は Vivado Design Suite のインメモリ デザインにのみ存在します。この制約はまだ timing.xdc ファイルに保存されていません。

クロック制約を作成すると、[Save Constraints] アイコンを使用できるようになります。



1. [Save Constraints] アイコンをクリックします。

   [No Target Constraints File] ダイアログ ボックスが開きます。timing.xdc ファイルは lab1 の制約セットにあるのですが、ターゲットの制約ファイルではないため、このダイアログ ボックスが表示されます。既存ファイルに制約を保存するか、または新しいファイルを作成できます。

2. 次の図のように、既存の timing.xdc ファイルを選択し、[OK] をクリックします。

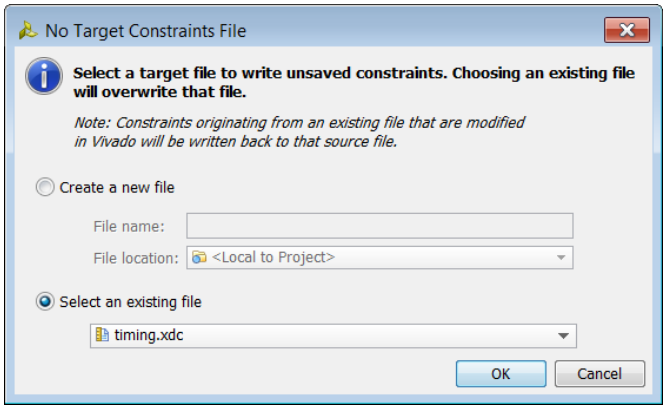


図 12 : [No Target Constraints File] ダイアログ ボックス

注記 : 制約を保存すると、[Save Constraints] アイコンは使用できなくなり、制約ファイルが最新のものであることを示します。

3. [Sources] ビューで lab1 制約セットの timing.xdc ファイルをダブルクリックします。

   Vivado のテキスト エディターで timing.xdc ファイルが開き、create_clock コマンドがカラーのリンク表示になっています。

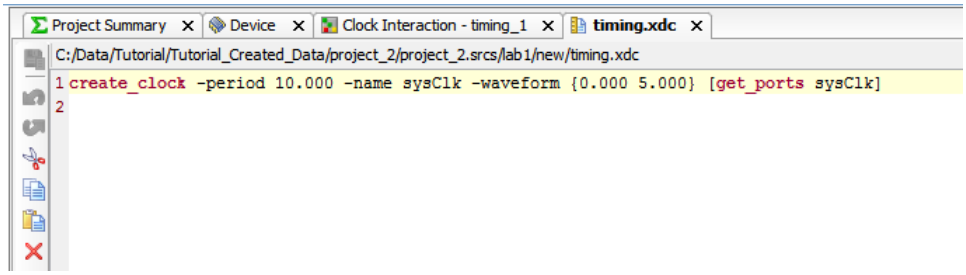


図 13 : timing.xdc ファイル

ヒント : 任意数のサードパーティのテキスト エディターをサポートするため Vivado IDE をカスタマイズできます。詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) を参照してください。

手順 6： 表を使用したタイミング制約の入力

手順 3 で説明した方法でほかのタイミング制約も作成できます。また、図 10 ： 追加された sysClk 制約にあるように、[Timing Constraints] ビューの表に直接タイミング制約を入力することもできます。

デフォルトでは、Vivado IDE でデザインの I/O ポートを起点または終点とするパスのタイミングは計算されません。まず、ポートに入力/出力遅延制約を割り当てる必要があります。この手順では、or1200_clmode ポートに入力遅延を割り当てます。しかし、その前に、このポートを起点とするパスのカスタム タイミング レポートを生成します。

1. [Tools] → [Timing] → [Report Timing] をクリックし、[Report Timing] ダイアログ ボックスを開きます。

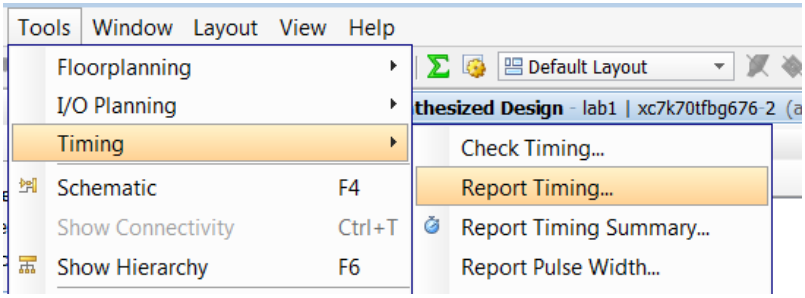


図 14 : [Report Timing] コマンド

図 15 に示すように [Report Timing] ダイアログ ボックスが開きます。指特定入力ポートからのタイミングをレポートするには、[From] フィールドに「get_ports {or1200_clmode}」と入力します。

2. [Start Points] の [From] フィールドに「get_ports」と直接入力するか、参照ボタン ( ) をクリックして [Choose Start Points] ダイアログ ボックスでポートを検索します。

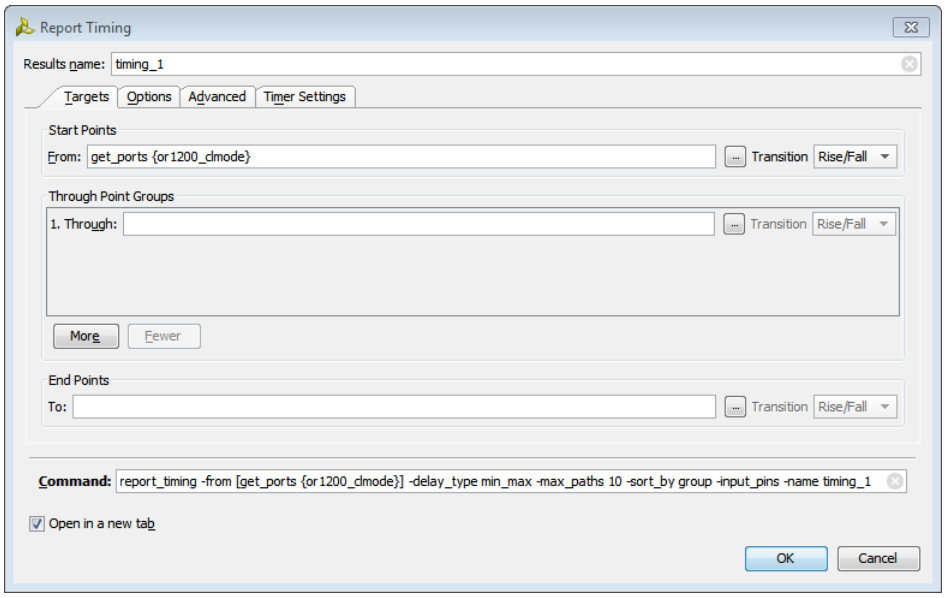


図 15 : [Report Timing] ダイアログ ボックス

ヒント： [Choose Start Points] ダイアログ ボックスで目的の入力ポートを検索するには、ポートのタイプを指定し、「or1200*」と指定します。

[Report Timing] ダイアログ ボックスの下部にある [Command] フィールドに Tcl コマンドが表示されます。

3. [OK] をクリックし、タイミング レポートを生成します。

Vivado IDE の [Timing] タブには、ワーストのセットアップ違反が 10、ワーストのホールド違反が 10 表示されています。レポートされている違反にはすべて無限のスラックがあります。図 16 にあるように、[Source Clock] 列には「input port clock」が表示されています。これらの制約が付いていないパスのタイミングは考慮されていません。

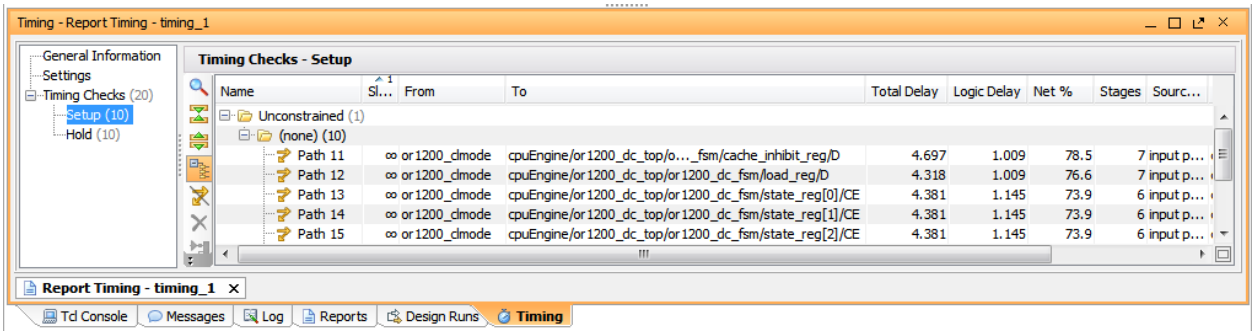


図 16： タイミング結果のレポート

report_timing コマンドの実行の詳細を見るには、Tcl コンソールを確認します。

4. [Timing Constraints] ビューが開いていない場合は、[Window] → [Timing Constraints] をクリックして開きます。
5. [Timing Constraints] ビューのツリー表示セクションの [Inputs] の下にある [Set Input Delay] を選択します。

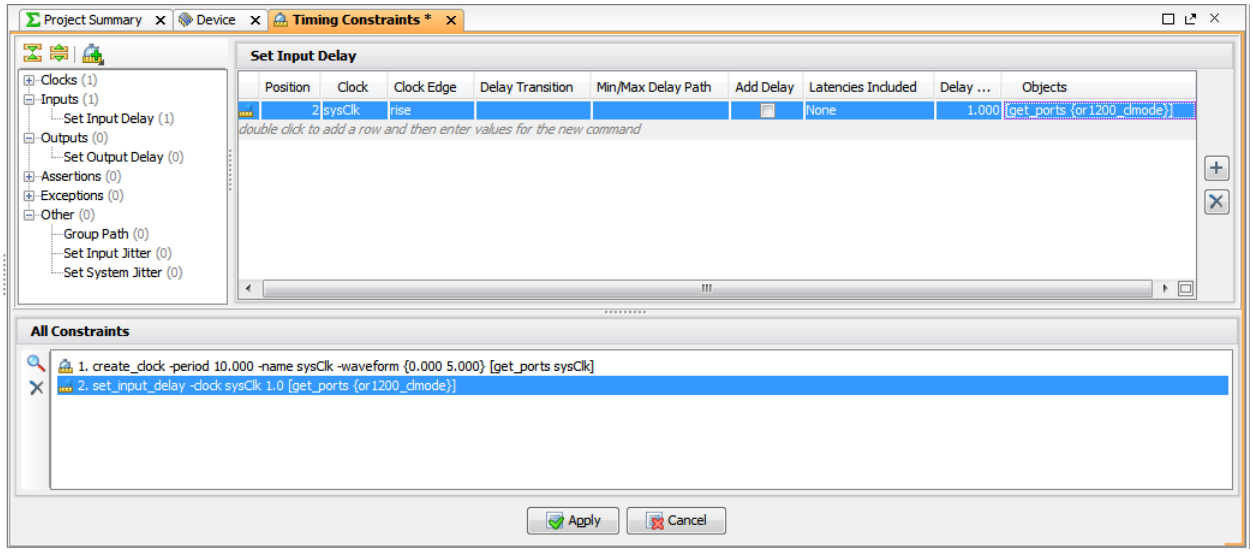


図 17： 入力遅延の設定

図 7 にある Create Clock 制約とは異なる列が Ser Input Delay の表には表示されていることに注目してください。

6. set_input_delay 制約の値を手動で入力するには、表の中でダブルクリックします。

   指定列に次の値を入力します。

   - [Clock] : sysClk
   - [Delay Value] : 1 ns
   - [Objects] : [get_ports {or1200_clmode}]

   get_ports コマンドは、[Objects] 列で直接入力するか、または参照ボタン ( ) をクリックして [Specify Delay Objects] ダイアログ ボックスを開き、or1200_clmode ポートを検索します。

ヒント : [Specify Delay Objects] ダイアログ ボックスで目的の入力ポートを検索するには、ポートのタイプを指定し、「or1200*」と指定します。

完了すると、表は 図 17 : 入力遅延の設定のようになるはずです。制約の横にある アイコンは、制約がまだデザインに適用されていないことを示します。このアイコンは [All Constraints] セクションにも表示されています。

6. インメモリ デザインにこの制約を適用するには、[Timing Constraints] ビューの下にある [Apply] ボタンをクリックします。
7. Tcl コンソールの上方向キーをクリックして Tcl コマンドをスクロールし前の手順で実行した report_timing コマンドを検索し、もう一度実行します。

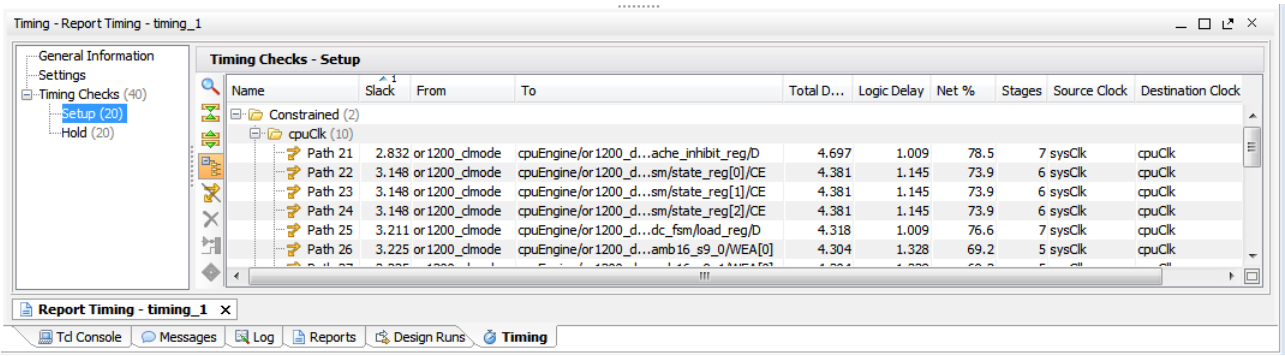


図 18： 問題のなかったタイミング結果のレポート

各パスの [Slack] 列に値が入力されていること、[Source Clock] および [Destination Clock] に値があることに注目してください。これらのパスはインプリメンテーションで考慮されます。

また、ファイルに保存されていない新しい制約があるため、[Save Constraints] アイコンが再び使用可能になっていることにも注目してください。このアイコンをクリックすると、timing.xdc ファイルの最後に set_input_delay コマンドが書き込まれます。前の手順で create_clock コマンドを保存したときにこのファイルはターゲットとして設定されています。

8. チュートリアルを終了する前に、[File] → [Save Constraints] をクリックして、タイミング制約を保存します。

9. Vivado IDE を終了するか、そのままにしておいて 演習 2 に進みます。

まとめ

Vivado IDE の [Timing Constraints] ビューで制約ウィザードおよび表を使用して、タイミング制約をデザインに追加する方法を学びました。
また、Tcl コンソールを使用して制約を Tcl コマンドとして追加および適用する方法も学びました。
デザイン制約を作成するため XDC ファイルを直接編集する方法もあります。

演習 2： 物理制約の設定

この演習では、GUI での操作が Tcl コマンドとして出力されるよう、CPU ネットリストのサンプル デザインに対して物理制約を作成します。Tcl コンソールのインタラクティブな機能を利用し、デザインをさまざまな点から検討し、解析します。繰り返し使用できるよう、また、フローのさまざまな段階に挿入できるよう、複雑な操作は簡単にスクリプト化されます。

手順 1： サンプル プロジェクトを開く

ヒント ： 演習 1 から作業を続けていて、サンプル プロジェクトが開いている場合は、手順 2 に進んでください。

1. Vivado IDE を起動します。
   - Windows デスクトップにある Vivado IDE のアイコンをクリックします。
   - コマンド ターミナルで「vivado」と入力します。
2. Getting Started ページで [Open Example Project] をクリックし、[CPU (Synthesized)] デザインを選択します。

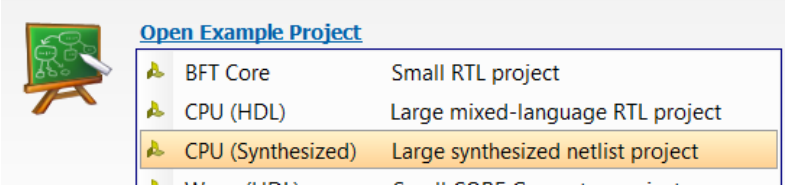


図 19： サンプル デザインを開く

元のサンプル デザインを上書きしないように、プロジェクトを別のディレクトリに保存します。

3. [File] → [Save Project As] をクリックし、プロジェクト名およびディレクトリを指定します。

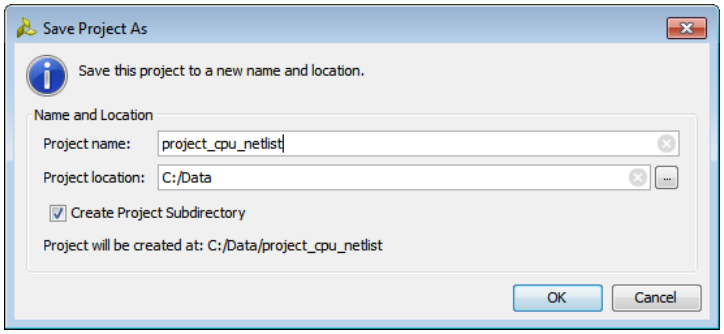


図 20 : [Save Design As] ダイアログ ボックス

プロジェクトが指定ディレクトリに保存されました。

手順 2： 配置制約の追加

物理制約を作成するため、合成されたデザインを開き、デザイン階層の一部を確認して、ロジック エレメントを配置し始めます。

1. Flow Navigator で [Open Synthesized Design] をクリックします。

   合成されたネットリストが開き、[Device] ビューに表示されます。

2. [Netlist] ビューを選択して、[clkgen] の階層を展開させます。
3. [Primitives] フォルダーを展開し、[mmcm_adv_inst (MMCME2_ADV)] を選択します。

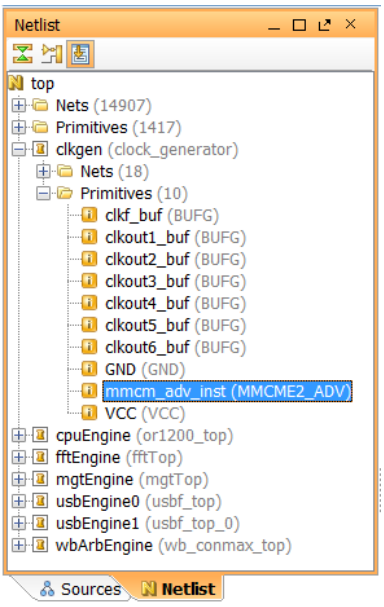


図 21 : [Netlist] ビュー

4. [Instance Properties] ビューの [Attributes] タブに IS_LOC_FIXED または IS_BEL_FIXED の属性が表示されていないことを確認します。
5. Tcl コンソールに次のコマンドを入力します。

   ```
   get_property IS_LOC_FIXED [get_cells clkgen/mmcm_adv_inst]
   ```

   ゼロが返されます。つまり、オブジェクトは特定ロケーションには固定されていません。

6. [Device] ビューの右下を拡大表示し、[Clock Region X1Y0] の下半分を表示させ、選択したオブジェクトの配置準備をします。
7. [Netlist] ビューで [mmcm_adv_inst] をクリックし、[Device] ビューにドラッグして右下の MMCME2_ADV に配置します。

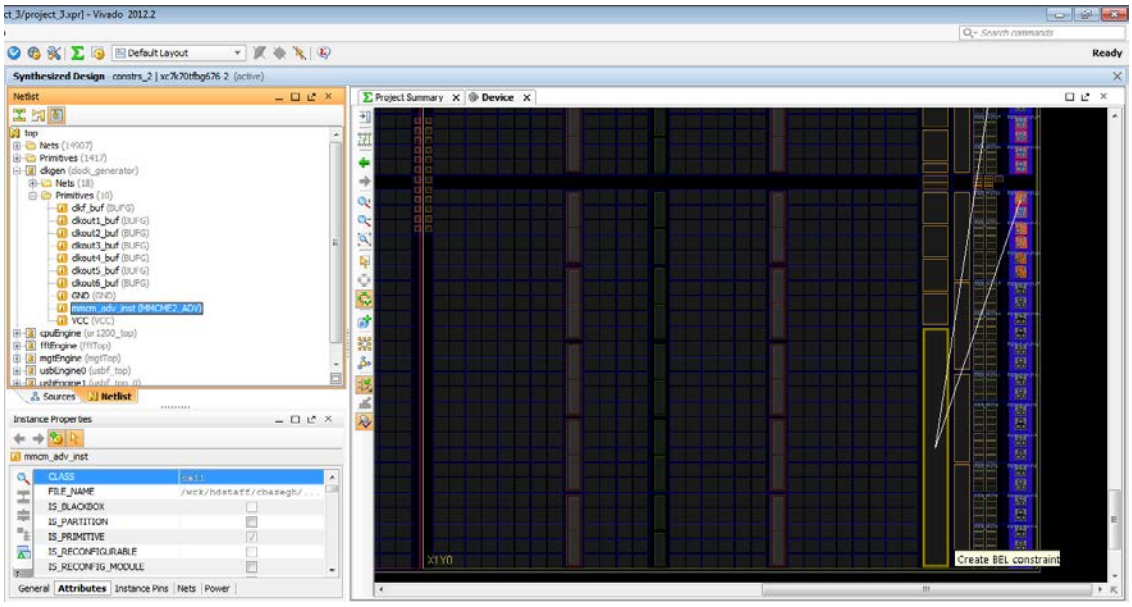


図 22 : MMCM の配置

Tcl コンソールを確認します。次の 3 つのコマンドが表示されているはずです。

```
startgroup
place_cell clkgen/mmcm_adv_inst MMCME2_ADV_X1Y0/MMCME2_ADV
endgroup
```

startgroup と endgroup の Tcl コマンドは、Vivado ツールで undo 機能をサポートするため、一連のコマンドをまとめるものです。操作を間違えたときは、Tcl コンソールで undo コマンドを使用して、配置を取り消し、もう一度やり直すことができます。Tcl コマンドの startgroup、endgroup、undo に関する詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) を参照してください。

8. [Instance Properties] ビューの [Attributes] タブをクリックし、MMCM が配置されていることを確認します。

   IS_BEL_FIXED および IS_LOC_FIXED の属性が、オブジェクトが配置されたことを反映して表示されています。

ヒント : オブジェクトを配置するとき、IS_LOC_FIXED のみが設定されるか、または IS_BEL_FIXED も設定されるかは、[Device] ビューのインスタンスのドラッグ アンド ドロップ モードによって決まります。[Device] ビューの使用についての詳細は、『Vivado Design Suite ユーザー ガイド : Vivado IDE の使用』(UG893) を参照してください。

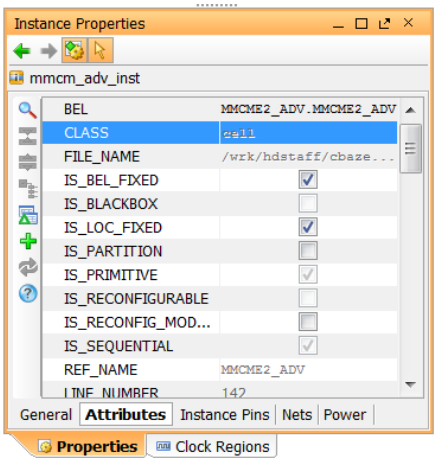
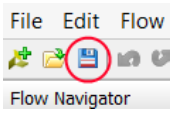


図 23 : BEL および LOC 配置制約

IS_BEL_FIXED および IS_LOC_FIXED 属性は物理制約で、そのオブジェクトの配置を反映しています。これらの制約は Vivado インプリメンテーションで使用されますが、変更はされません。しかし、属性が無効である場合は、デザイン フローの後段階でエラーが発生します。

[Device] ビューで mmcm_adv_inst を配置すると、[Save Constraints] アイコンを使用できるようになります。物理制約は Vivado ツールのインメモリ デザインに追加されましたが、制約ファイルにはまだ保存されていません。



手順 3 : 追加物理制約の定義

この手順では、PACKAGE_PIN 制約や PROHIBIT 制約などの追加物理制約をデザインに定義します。

1. ツール バーから [I/O Planning] レイアウトを選択します。

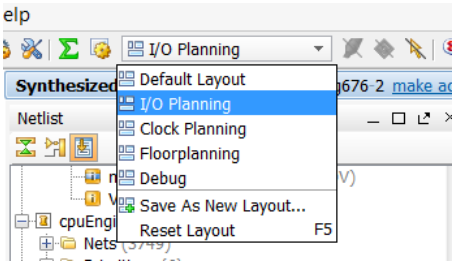


図 24 : レイアウトの選択

[I/O Planning] レイアウトには [Package]、[I/O Ports]、[Package Pins] ビューが表示され、I/O ポートを割り当てしやすくなっています。

このチュートリアルでは、PCB レイアウトが既に完了していて、FPGA パッケージで一部のピンにアクセスできないものと想定します。配置配線中に Vivado ツールでこれらのピンが使用されないようにすることができます (I/O 割り当てをすべて完了していないものとする)。

2. [Package] ビューで **AA8** ピンを選択します。

   [Package] ビューの X/Y 軸の値を指定し、パッケージでピンを検索しやすくします。

3. ピンを右クリックし、[Set Prohibit] をクリックします。

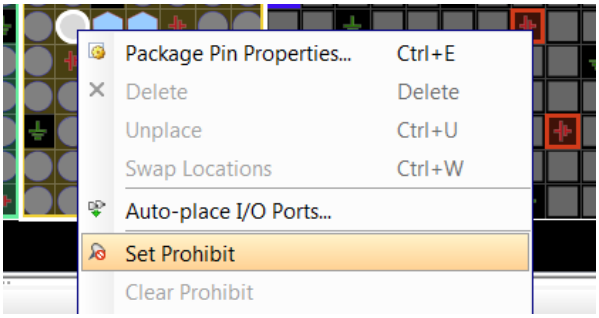


図 25 : [Set Prohibit] コマンド

   ピンの選択を解除すると、このサイトに赤い丸に斜めの線が入ったマークで表示され、このサイトが使用できないことを示しています。

4. Tcl コンソールで Vivado IDE で生成された Tcl コマンドを確認します。

```
set_property prohibit 1 [get_sites AA8]
```

手順 4： セル プロパティを使用した制約の定義

このチュートリアルで先に説明したようにタイミングおよび配置の制約を作成することができますが、セルのプロパティを変更して、Vivado インプリメンテーションでのその処理方法を制御することもできます。多くの物理制約はセル オブジェクトのプロパティとして定義されています。

たとえば、デザインにある RAM にタイミングの問題があることが発覚したとします。この RAM セルのプロパティを変更し、パイプライン レジスタに追加することが可能です。このアプローチで問題がないことを設計および検証チームから確認が得られたら、デザインを変更することができます。

合成の後に RTL に戻るのはコストが高くつきすぎることがあるため、次のようにネットリストで変更します。

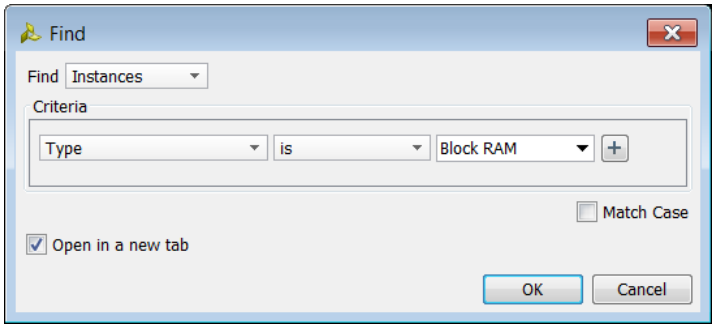


図 26 : [Find] ダイアログ ボックス

1. [Edit] → [Find] をクリックし、図 26 にあるように [Find] ダイアログ ボックスを開き、ブロック RAM を検索します。
   a. [Find] で [Instances] を選択します。
   b. [Criteria] で [Type]、[is]、[Block RAM] と選択します。
   c. [OK] をクリックします。

   [Find Results] ビューが開きます。
2. 一番最初に表示されるセルを選択します。これは「RAMB36E1」というセルのはずです。

   ```
   fftEngine/fftInst/ingressLoop[7].ingressFifo/…
   ```

   [Instance Properties] ビューの [Attributes] タブで、DOA_REG および DOB_REG がゼロに設定されているはずです。つまり、出力レジスタがディスエーブルになっているということです。
3. Tcl コンソールで直接、このセルのカスタム タイミング レポートを生成します。次の Tcl コマンドを入力します。

   ```
   report_timing -from [get_cells
   fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.
   block_ram_performance.fifo_ram_reg]
   ```

ヒント : [Instance Properties] ビューの [General] タブからセル名をコピーして、Tcl コンソールに貼り付けることができます。

レポートのデータ パス セクションで、この RAM に対し、1.800ns が追加されています。

```
RAMB36E1 (Prop_ramb36e1_CLKBWRCLK_DOBDO[16])
                             1.800    -0.078 r  fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg/DOBDO[16]
net (fo=2, unplaced)         1.097     1.019    fftInst/toBft[15]_39[0]
                                             r  transformLoop[7].ct/xOutReg_reg/B[0]
DSP48E1 (Setup_dsp48e1_CLK_B[0])
                             0.317     1.336    transformLoop[7].ct/xOutReg_reg
```

4. [Instance Properties] ビューの [Attributes] タブで、このセルの DOA_REG および DOB_REG プロパティを選択し、その値を 0 から 1 に変更します。
5. [Instance Properties] ビューで [Apply] ボタンをクリックします。

   Tcl コンソールで 2 つの set_property コマンドが実行されたのが確認できます。

   これらはオブジェクトのプロパティであり、タイミング制約としては直接定義されていないため、Vivado ツールでタイミング データをアップデートする必要があります。この後、タイミング レポートに戻り、プロパティの変更が反映されていることを確認します。
6. Tcl コンソールで次のコマンドを入力して、タイミング データをアップデートします。

   ```
   update_timing -full
   ```

7. 選択したセルのタイミング レポートを生成し直します。次の Tcl コマンドを入力します。

```
report_timing -from [get_cells
fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.
block_ram_performance.fifo_ram_reg]
```

8. RAM のデータ パス遅延が 0.622ns になっていることに注目してください。

次に、デザインに対しコンフィギュレーション モードを設定します。これは別のプロパティで物理制約でもあります。またこの場合は、セルのプロパティではなく、デザインのプロパティです。まず、現在のデザインのプロパティをすべてリストします。

1. Tcl コンソールで次のコマンドを入力してプロパティをリストします。

   ```
   list_property [current_design]
   ```

   現在のデザインに含まれるすべてのプロパティのリストが返されます。このリストを読みやすくするため、標準 Tcl コマンドである join を使用して、「¥n」という改行文字を追加して、プロパティごとに改行されて表示されるようにします。

   ```
   join [list_property [current_design]] ¥n
   ```

2. ここでは、CONFIG_MODE というプロパティを選択します。このプロパティに使用可能な値を確認するため、list_property_value という Tcl コマンドを使用します。

   ```
   join [list_property_value CONFIG_MODE [current_design]] ¥n
   ```

3. このプロジェクトに対しては、CONFIG_MODE プロパティを M_SERIAL に設定します。

   ```
   set_property CONFIG_MODE M_SERIAL [current_design]
   ```

   これでコンフィギュレーション モードを設定することができました。

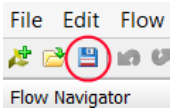
4. CONFIG_MODE プロパティが正しく設定されていることを確認するには、get_property コマンドを使用します。

   ```
   get_property CONFIG_MODE [current_design]
   ```

   Vivado ツールにより「M_SERIAL」という値が返されます。

手順 5： 制約の保存

新しいデザイン制約があるため [Save Contraints] アイコンが有効になっています。演習 2 で変更したセルおよびデザインのプロパティが Vivado ツールのインメモリ デザインに追加されていますが、またターゲット制約ファイルには保存されていません。



1. [Save Constraints] アイコンをクリックします。

   これでここまでで定義した物理制約がターゲット制約ファイルに保存されました。

2. [Sources] ビューにあるアクティブ制約セットから XDC を選択し、Vivado IDE のテキスト エディターでこのファイルを開きます。

   注記 : 開いた制約ファイルは、演習 1 から続けて使用したものか、演習 2 で Vivado IDE を再起動させて使用したものなのかによって異なります。

   ファイルには、set_property コマンドが 5 つだけ保存されていることを確認してください。制約のみが XDC に書き込まれていて、オブジェクト クエリーやレポート コマンドは含まれていません。

   - `set_property LOC MMCME2_ADV_X1Y0 [get_cells clkgen/mmcm_adv_inst]`
   - `set_property PROHIBIT true [get_sites AA8]`
   - `set_property CONFIG_MODE M_SERIAL [current_design]`
   - `set_property DOA_REG 1 [get_cells {fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg}]`
   - `set_property DOB_REG 1 [get_cells {fftEngine/fftInst/ingressLoop[7].ingressFifo/buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg}]`

3. Vivado IDE を終了します。

まとめ

この演習では、Vivado IDE および Tcl コンソールの両方を使用して、物理制約を作成し確認する方法を学びました。Vivado IDE での操作のほとんどは、Tcl コンソールで Tcl コマンドとして実行されることも確認でき、Vivado IDE には物理制約やタイミング制約を設定するための強力でインタラクティブな機能があり、これを制約ファイルに保存して、適宜再利用できることを学習しました。