Vivado Design Suite Tutorial

Using Constraints

UG945 (v2012.2) August 8, 2012

XILINX®

Notice of Disclaimer

Table of Contents

Vivado Using Constraints Tutorial

Overview

This tutorial is comprised of various labs, each of which seeks to demonstrate an aspect of constraining a design in the Xilinx® Vivado™ Integrated Design Environment (IDE). In Vivado, the constraints format is called Xilinx Design Constraints (XDC), which is a combination of the industry standard Synopsys® Design Constraints and proprietary Xilinx constraints.

XDCs are not just simple strings; they are TCL commands that the Vivado Tcl interpreter sequentially reads and parses just like any Tcl command. You can enter design constraints in several ways at different points in the design flow. You can store XDCs in one or more files that can be added to a constraint set in Vivado Project Mode, or directly read the same files into memory using the `read_xdc` command in Non-Project Mode. For more information on Project and Non-Project Modes, refer to the *Vivado Design Suite User Guide: Design Flows Overview (UG892)*. With a design open in Vivado, you can also type constraints as commands directly in the Tcl console or at the Tcl command prompt. This is particularly powerful for defining, validating and debugging new constraints interactively in the design.

The Vivado Design Suite synthesis and implementation tools are timing driven. Having accurate and correct timing constraints is vital for meeting design goals and ensuring correct operation. Because the Vivado tools are timing driven, it is important to fully constrain a design, but not over-constrain, or under-constrain it. Over-constraining a design can lead to long run-times and sub-optimal results because the tool can struggle with unrealistic design objectives. Under constraining a design can cause the Vivado tools to perform unnecessary optimizations, such as examining paths with multicycle delays or false paths, and prevent focus on the real critical paths.

This tutorial discusses different methods for defining and applying design constraints. The labs included in this tutorial are:

- **Lab #1**: Using the Timing Constraint Editor
- **Lab #2**: Using the Tcl Console

Software Requirements

This tutorial requires that the 2012.2 Vivado Design Suite software release or later is installed.

Hardware Requirements

The supported Operating Systems include Redhat 5.6 Linux 64 and Windows 64 and 32 bit.

Xilinx recommends a minimum of 2 GB of RAM when using the Vivado tool.

Tutorial Design Description

The sample design used throughout this tutorial consists of a small design called `project_cpu_netlist`. There is a top-level EDIF netlist source file, as well as an XDC constraints file.

The design targets an XC7K70T device. A small design is used to allow the tutorial to be run with minimal hardware requirements and to enable timely completion of the tutorial, as well as to minimize the data size.

Preparing the Tutorial Design Files

You can find the files for this tutorial in the Vivado Design Suite examples directory at the following location:

- ***<Xilinx_2012.2_install_area>*/Vivado/<version>/examples/Vivado_Tutorial**

You can also extract the provided zip file, at any time, to write the tutorial files to your local directory, or to restore the files to their starting condition.

Extract the zip file contents from the software installation into any write-accessible location.

- ***<Xilinx_2012.2_install_area>*/Vivado/<version>/examples/Vivado_Tutorial.zip**

The extracted `Vivado_Tutorial` directory is referred to as the <*Extract_Dir*> in this Tutorial.

Note: You will modify the tutorial design data while working through this tutorial. You should use a new copy of the original `Vivado_Tutorial` directory each time you start this tutorial.

Lab #1: Using the Timing Constraint Editor

In this lab, you will learn two methods of creating constraints for a design. You will be using the CPU Netlist example design that is included in the Vivado IDE.

Step 1: Opening the Example Project

1. Start by loading Vivado IDE.
   - Launch Vivado IDE from the icon on the Windows desktop, or
   - Type **vivado** from a command terminal.
2. From the Getting Started page, click **Open Example Project** and select the **CPU (Synthesized)** design.

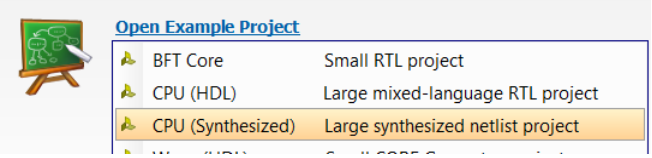


Figure 1: Open Example Design

3. The design project opens in the Vivado IDE, and a dialog box opens stating that the project is read-only and prompting you to save the project to a new location.

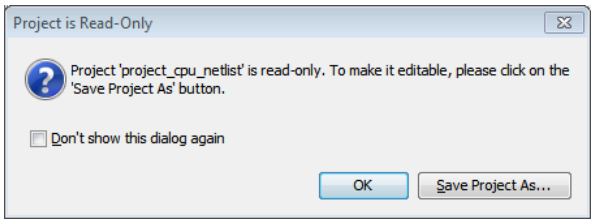


Figure 2: Read-Only Project

4. Select **Save Project As** and specify a project name and location.

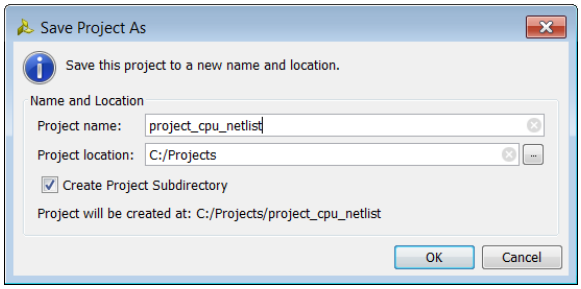


Figure 3: Save Project As...

The project saves to the specified location.

The Vivado IDE displays project information in the Project Summary window. Because this is a netlist project, there are no options for synthesis in the Flow Navigator. The Project Summary window shows that the next step in the design flow is Vivado implementation.

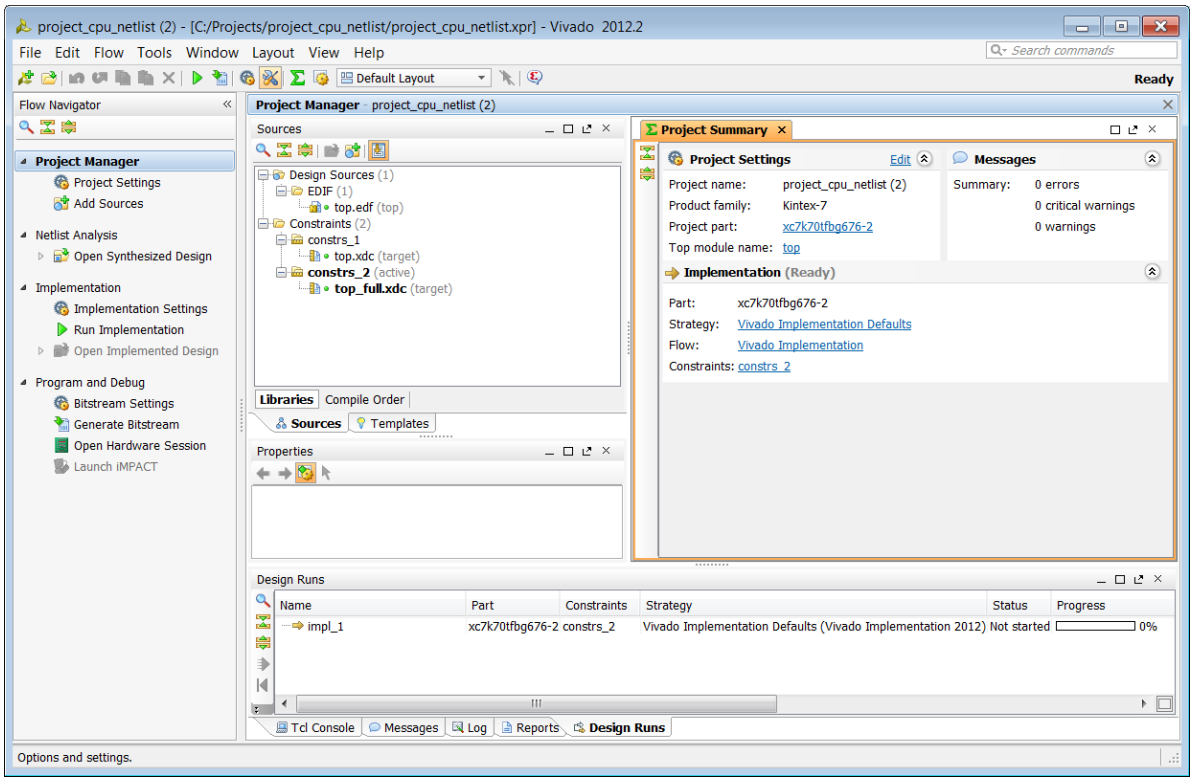


Figure 4: Project Summary window

Step 2: Defining Constraint Sets and Files

Important! *The Vivado Design Suite does not support the UCF format. For information on migrating UCF constraints to XDC commands refer to the Vivado Design Suite Migration Methodology Guide (UG912).*

Start by creating a new constraint set and adding an empty XDC constraints file to it. The sample design already contains two constraint sets, but you do not use them for this lab.

1. From the Flow Navigator, select **Add Sources** in the Project Manager section.
2. From the list displayed in the Add Sources dialog box, select **Add or Create Constraints** and click **Next**.

3. From the Add or Create Constraints dialog box, use the Specify Constraint Set: drop down menu, and select **Create Constraint Set** as shown below.

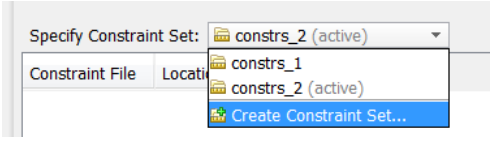


Figure 5: Create Constraint Set

4. In the Create Constraint Set Name dialog box, specify the constraint set name as **`lab1`** and click **OK**.
5. Enable the **Make active** checkbox.
6. Select **Create File** to add a new XDC file to the project. Enter **`timing`** as the file name, leave the file as **<Local to Project>**, and click **OK**.

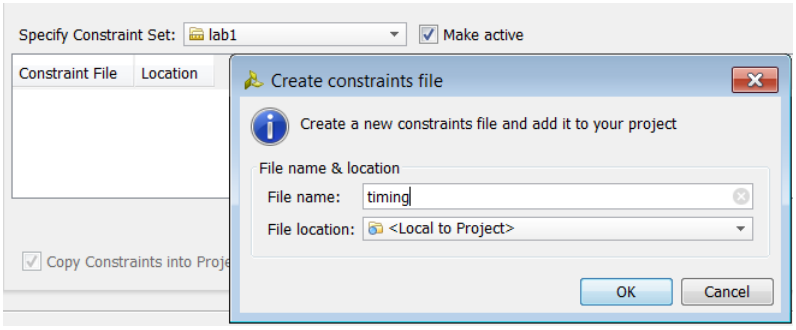


Figure 6: Constraints File Name

The `timing.xdc` file is added to the `lab1` constraint set.

7. Select **Finish** to complete the creation of the new constraint set and XDC file.

   You should see the new constraint set and XDC file in the Sources window as shown below. The constraint set is made active as you directed when you created it.

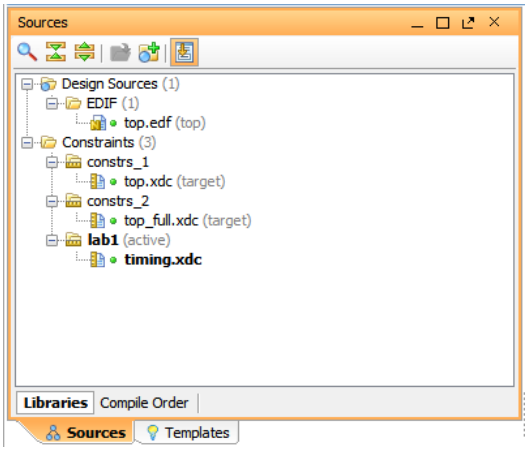


Figure 7: Sources window

Step 3: Using a Timing Constraint Wizard

Now open the synthesized design, and create some new timing constraints in the design. You will create a clock for this design, since you need one or more clocks in order to do timing analysis, and to perform timing driving place and route.

1. From the Flow Navigator, select **Open Synthesized Design**.

   The synthesized netlist opens with the Device window displayed.

2. Select **Edit Timing Constraints** from the Flow Navigator under the Netlist Analysis section. The Vivado IDE displays the Timing Constraints window.

   There are three sections to the Timing Constraints window:

   - **Constraints tree view:** Located in the upper-left of the Timing Constraints window, as shown in *Figure 8: Timing Constraints window*. This section displays standard timing constraints, grouped by category. Double-clicking on a constraint in this section opens a Constraints wizard to help you define the selected constraint.
   - **Constraints Spreadsheet:** Located in the upper right of *Figure 8*. This section displays timing constraints of the type currently selected in the Constraints tree view. You can use this to directly define or edit constraints instead of the Constraints wizard if you prefer.
   - **All Constraints:** Located at the bottom of the Timing Constraints window, this section displays all the currently defined timing constraints in the design.

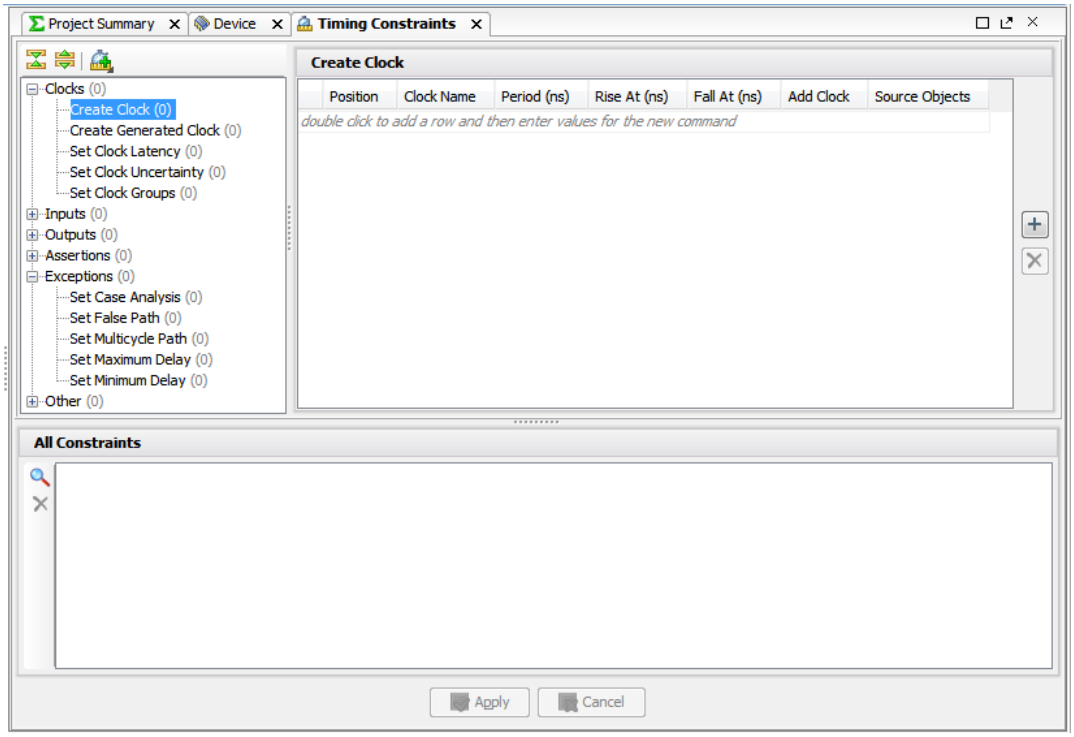


Figure 8: Timing Constraints window

3. Under the Clocks heading of the Constraints tree view, double-click **Create Clock**. This opens the Create Clock wizard as shown below.

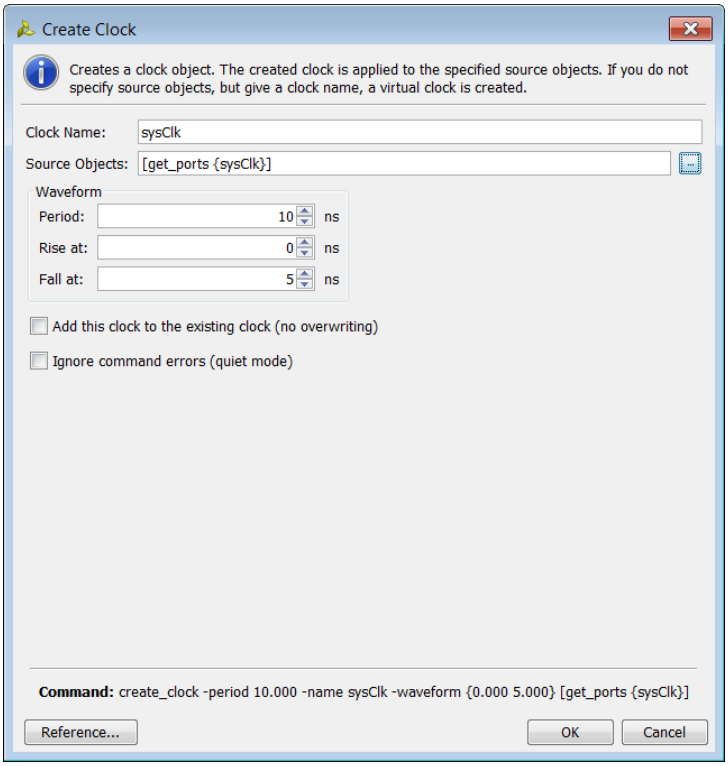


Figure 9: Create Clock wizard

a. Enter **sysClk** for the Clock name.

   The clock name can be any name, and does not have to match any element of the design (port or pin); it is just a name. However, typically the name of a primary clock matches the name of its input port.

b. For the Source Objects field select the browse button (...) to bring up the Specify Clock Sources Objects search window as shown in *Figure 10: Specify Clock Sources*.

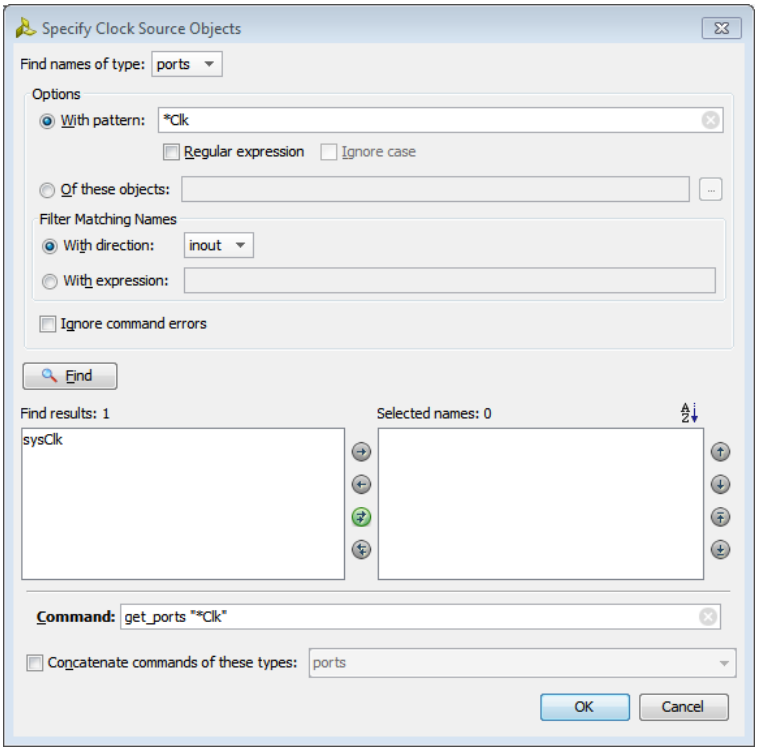


Figure 10: Specify Clock Sources

c. Check that **Find Name of type:** is set to **ports**.

d. Change the **"With pattern"** from **"*"** to **"*Clk"** and click the **Find** button.

   `sysClk` should appear under **Find results**.

e. Select **`sysClk`,** and click the directional (right) green arrow to move it under **Selected names**.

Tip: *You can also double-click `sysClk` to move it from Find results to Selected names.*

Notice that the Command field displayed at the bottom of the dialog box changes as you perform these different actions. The `get_ports` command changes to:

```
get_ports {sysClk}
```

f. Select **OK** to finish specifying the clock sources, and return to the Create Clock wizard.

   The Create Clock wizard should now look as shown in *Figure 9: Create Clock wizard*. Accept the default values of the Waveform section, a period of 10ns with a 50% duty cycle. You can change these values as needed by using the up and down arrows, or by directly typing values. Notice the **Command:** field at the bottom of the window:

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

The Vivado IDE displays the Tcl command form of all constraints created by design wizards for your review. This is useful for learning the Tcl command syntax, and for verifying the final constraint before adding it.

g. Click **OK** to close the Create Clock wizard, and create the `sysClk` clock constraint as shown in *Figure 11: The Added `sysClk` Constraint*.

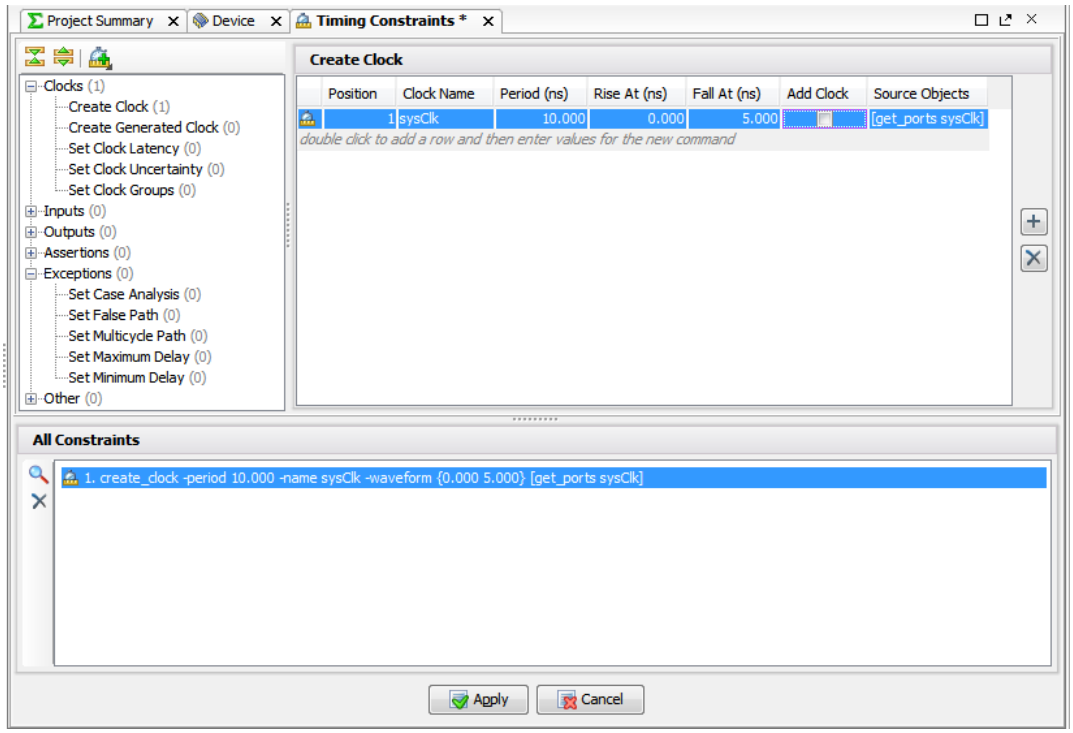


Figure 11: The Added `sysClk` Constraint

You see under the Constraints tree view that one Create Clock constraint has been added, indicating that the design has one clock. You can see the various properties of the `sysClk` you created in the other sections of the Timing Constraints window as well.

4. Click **Apply** at the bottom of the Timing Constraints window to save the `sysClk` constraint, and update the in-memory design with any new or changed constraints.

Important! *You must use the Save Constraints command to save any constraint changes to the `timing.xdc` file.*

Step 4: Reporting Clock Interactions

You can now report the interactions between the different clocks in the design, including the `sysClk` you defined in the prior step.

1. In the Netlist Analysis section of the Flow Navigator, select **Report Clock Interaction** and click **OK** in the Report clock Interaction dialog box to accept the default settings.

   The Vivado IDE generates a graphical matrix showing the relationship of the various clocks in the design. You see the `sysClk` you just created, as well as the automatically generated clocks that come from the MMCM in the design.

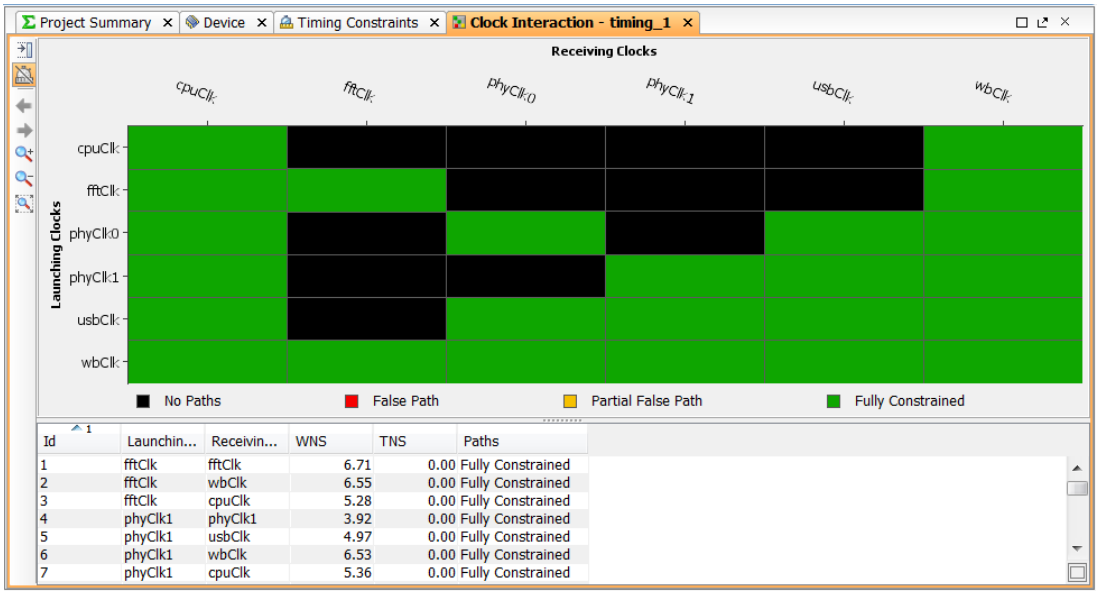
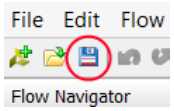


Figure 12: Clock Interaction report

Step 5: Saving Constraints

You have now created a primary clock for the design, but the constraint exists only in the Vivado Design Suite in-memory design. You have not yet saved the constraint to the `timing.xdc` file.

Notice that when you create the clock constraint, the Save Constraints icon is enabled.



1. Click the **Save Constraints** icon.

   The No Target Constraints File dialog box opens. This is because, although the `timing.xdc` file is in the `lab1` constraint set, it is not the target constraint file. You can save the constraints to an existing file, or create a new one.

2. Select the existing `timing.xdc` file as shown below, and click **OK**.

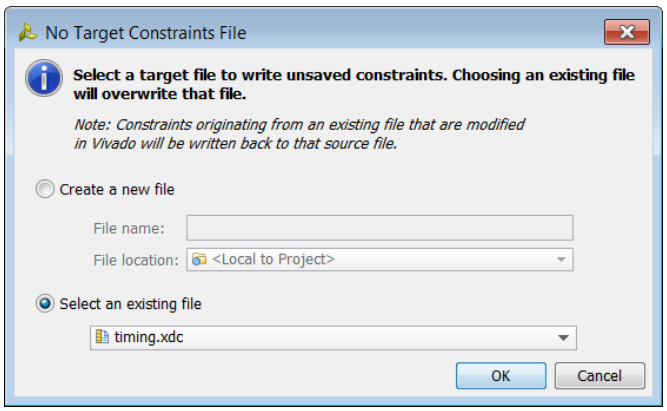


Figure 13: No Target Constraints File

Note: After you save the constraints, the Save Constraints icon becomes disabled, indicating the constraint files are up-to-date.

3. Double-click on the `timing.xdc` file in the `lab1` constraint set, in the Sources window.

   The `timing.xdc` opens in the Vivado text editor, and shows the `create_clock` command with context-sensitive text coloring.

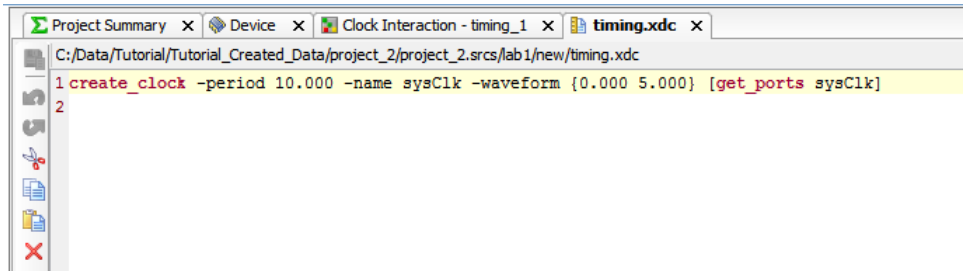


Figure 14: timing.xdc file

Tip: *The Vivado IDE can be customized to support any of a number of third party text editors. Refer to the Vivado Design Suite User Guide: Using the Integrated Design Environment (UG893) for more information.*

Step 6: Using the Timing Constraint Spreadsheet

You can create other timing constraints following the process described in *Step 3: Creating Clock Constraints*. You can also enter constraints directly into the spreadsheet of the Timing Constraints window, as shown in *Figure 11: The Added `sysClk` Constraint*.

By default, the Vivado IDE does not time paths to or from I/O ports in the design. You must first assign input/output delay constraints to the ports. In this step you assign an input delay onto the `or1200_clmode` port. Before that though, generate a custom timing report for paths starting from this port.

1. Select **Tools > Timing > Report Timing** to bring up the Report Timing dialog box.

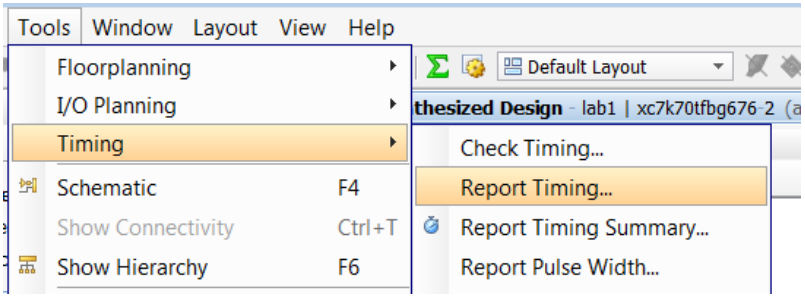


Figure 15: Report Timing command

The report Timing dialog box opens as shown in *Figure 16*. To report timing from the specific input port, set the **From** field as follows: `get_ports {or1200_clmode}`

You can type the `get_ports` command directly into the **Start Points From** field, or use the browse button (...), and search for the specific port in the Choose Start Points dialog box.

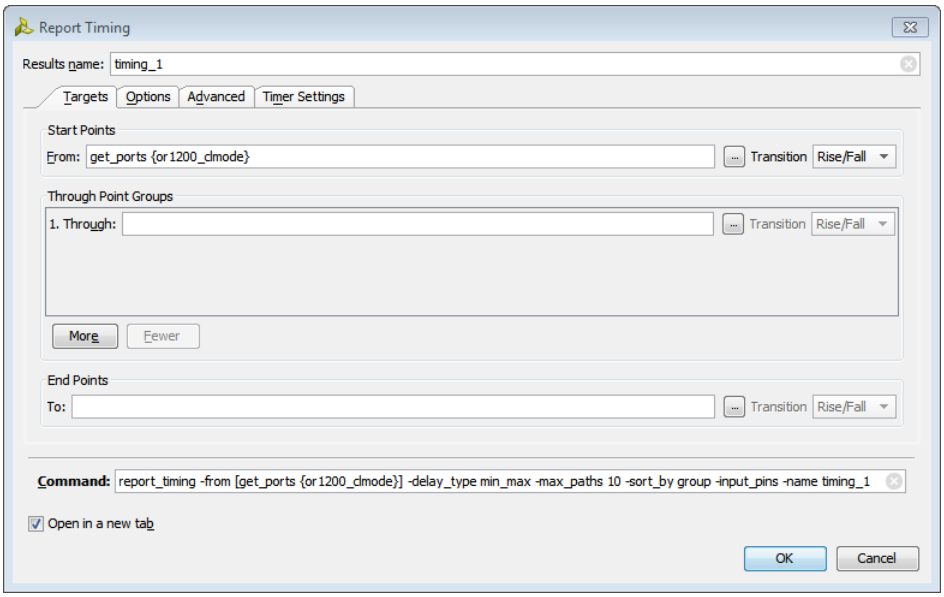


Figure 16: Report Timing dialog box

> ***Tip:*** *Find names of type port, and specify With Pattern or1200* to find the desired input port in the Chose Start Points dialog box.*

Notice the complete Tcl command in the Command field at the bottom of the Report Timing dialog box.

2. Click **OK** to generate the timing report.

   The Timing tab displays at the bottom of the Vivado IDE, showing the 10 worst Setup violations, and 10 worst Hold violations. All of the reported violations have an infinite slack. In the Source Clock column you can see "input port clock", as shown in *Figure 17*. Timing for these unconstrained paths is not considered.

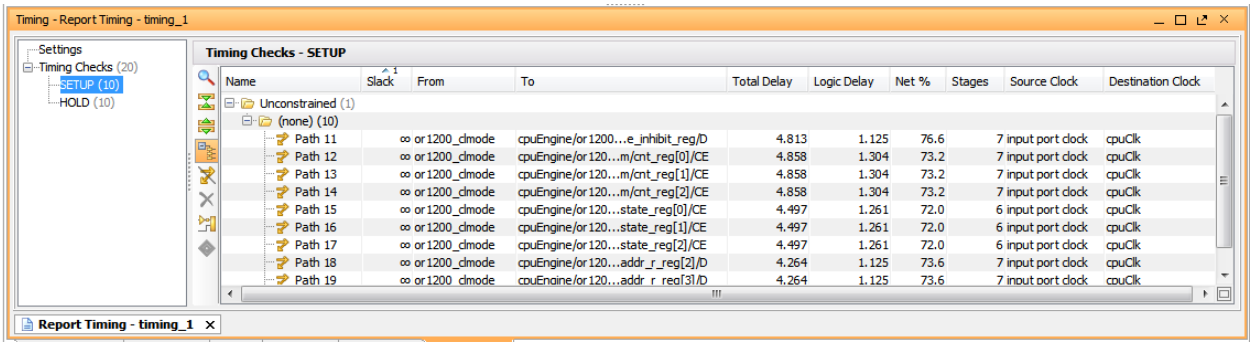


Figure 17: Report Timing Results

Look in the Tcl Console to see the run details of the `report_timing` command.

3. Use the **Window > Timing Constraints** command to open the Timing Constraints window if it is not currently open.

4. In the Timing Constraints window, select **Set Input Delay** from the Inputs heading of the Constraints tree view.

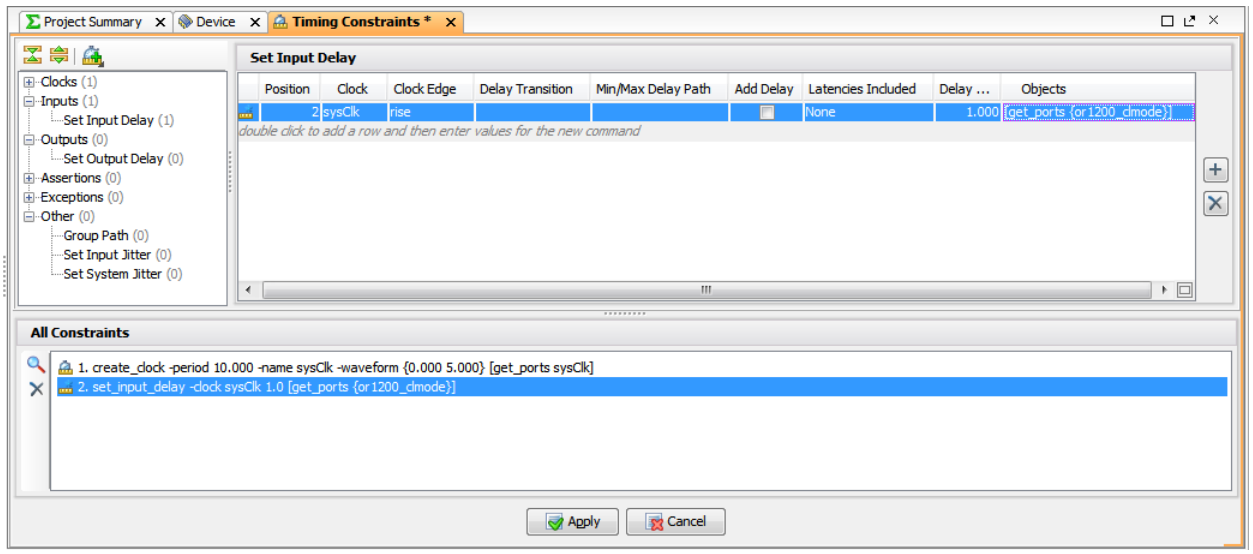


Figure 18: Set Input Delay

Notice that the Constraints spreadsheet displays different columns for Set Input Delay than for the Create Clock constraint as shown in *Figure 8*.

5. Double-click in the Constraint spreadsheet to manually enter values for the `set_input_delay` constraint.

   Enter the following values under the specified columns:

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

   You can directly type the `get_ports` command in the **Objects** column, or you can use the browse button ( ) to open the Specify Delay Objects dialog box to search for the `or1200_clmode` port.

Tip: *Find names of type port, and specify With Pattern or1200* to find the desired input port in the Specify Delay Objects dialog box.*

The Constraint spreadsheet should look like *Figure 18: Set Input Delay* when complete. The icon next to the constraint, which is also displayed in the All Constraints section,. indicates the command has not yet been applied to the design.

6. To apply the constraint to the in-memory design click the **Apply** button at the bottom of the Timing Constraints window.
7. Go to the Tcl Console and press the up-arrow key to scroll through the transcripted Tcl commands to find the `report_timing` command you previously ran, and re-run it here.

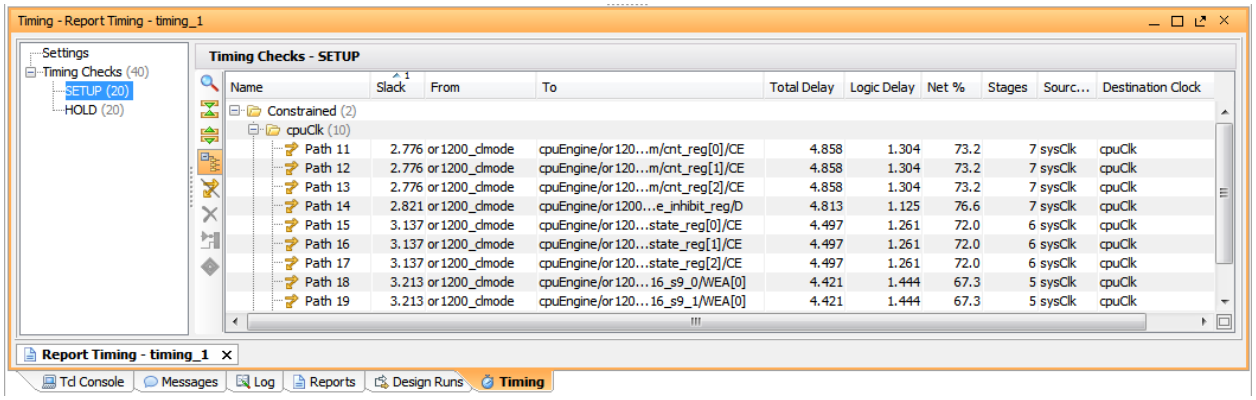


Figure 19: Report Timing Success

Notice that you now see values in the Slack column, and have Source and Destination Clock values for each path. During implementation, these paths are considered.

Also, notice that the Save Constraints icon is enabled again since there is a new constraint that you have not yet saved to a file. Pressing this will write the `set_input_delay`

command to the end of the `timing.xdc`. This file was set as the target when you saved the `create_clock` command earlier.

8. **Exit** the Vivado IDE, or keep it open and continue to **Lab #2**: Using the Tcl Console.

Summary

You have learned how to add timing constraints to a design using a constraint wizard and the Constraints spreadsheet from the Timing Constraints window in the Vivado IDE.

You can also use the Tcl Console to interactively add and apply constraints to the design as Tcl commands.

Still another approach is to work directly with the XDC file to create design constraints.

Lab #2: Setting Physical Constraints

In this lab, you will create physical constraints for the CPU Netlist sample design, to demonstrate that actions in the GUI produce Tcl commands. The interactive capabilities of the Tcl Console allow exploration of a design, and experimentation and analysis. Complex operations are easily scripted for repeated use, even for inclusion at various stages of the flow.

Step 1: Opening the Example Project

Tip: *If you are continuing from Lab #1, and your example project is still open, you can skip ahead to Step 2.*

1. Start by loading Vivado IDE.
   - Launch Vivado IDE from an icon on the Windows desktop, or
   - Type **vivado** from a command terminal.
2. From the Getting Started page, click **Open Example Project** and select the **CPU (Synthesized)** design.

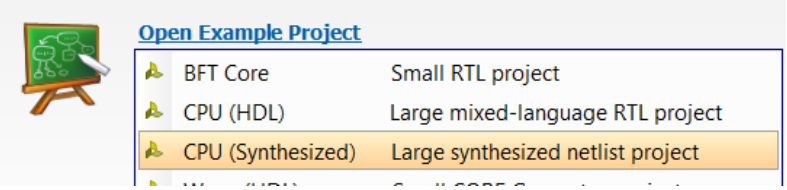


Figure 20: Open Example Design

3. The design project opens in the Vivado IDE, and a dialog box opens stating that the project is read-only and prompting you to save the project to a new location.

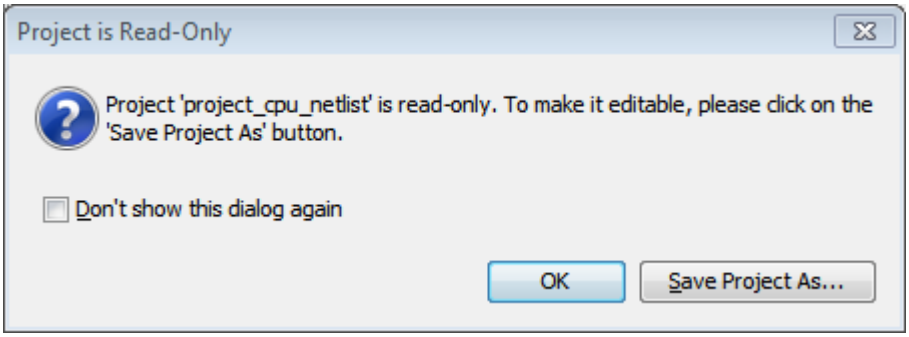


Figure 21: Read-Only Project

4. Select **Save Project As...** and specify a project name and location.

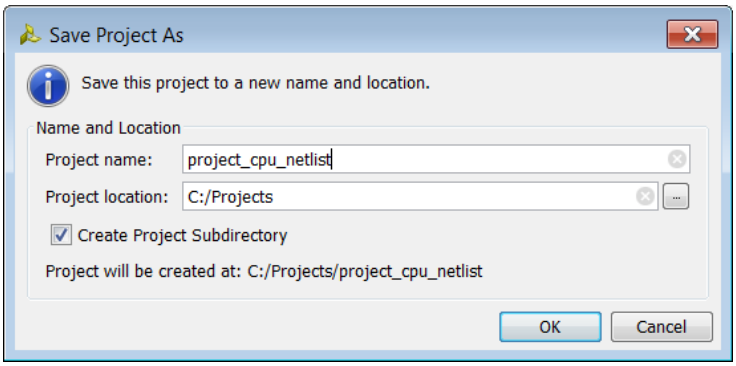


Figure 22: Save Project As...

The project saves to the specified location.

Step 2: Adding Placement Constraints

Open the Synthesized Design, explore some of the design hierarchy, and begin placing logic elements to create physical constraints.

1. From the Flow Navigator, select **Open Synthesized Design**.

   The synthesized netlist opens with the Device window displayed.

2. Select the **Netlist** window and expand the **clkgen** hierarchy.
3. Expand the **Primitives** folder and select the **mmcm_adv_inst** (MMCME2_ADV) cell.

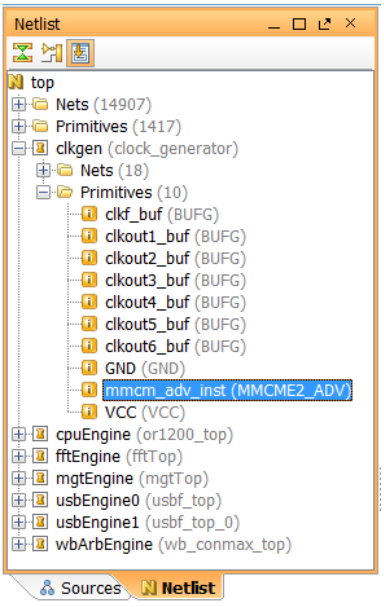


Figure 23: Netlist window

4. Look in the Properties view, under the Attributes tab, and notice that there are no `IS_LOC_FIXED` or `IS_BEL_FIXED` attributes shown.
5. Check this in the Tcl Console by typing:

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

   This returns a zero, indicating the object is not fixed to a location.
6. Zoom into the bottom right of the Device view, to display the lower half of **Clock Region X1Y0**, to prepare for placing the selected object.
7. In the Netlist window, click on the **mmcm_adv_inst** and drag it into the Device window to place it into the bottom right MMCME2_ADV.

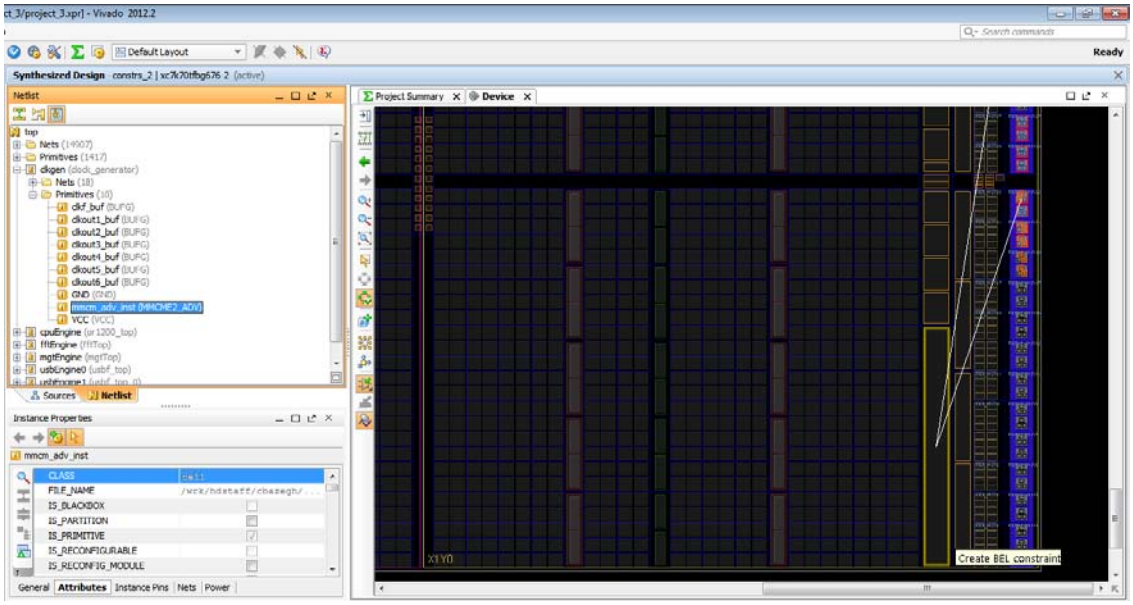


Figure 24:Placing the MMCM

Look in the Tcl Console. You should see these three commands:

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

The `startgroup` and `endgroup` Tcl commands bracket sequences of commands to support the undo function in the Vivado tools. If you make a mistake, you can use the `undo` command in the Tcl Console. This will undo the placement and allow you to redo it. For more information on `startgroup`, `endgroup`, and `undo`, refer to the *Vivado Design Suite Tcl Command Reference Guide (UG835)*.

8. Look again at the Attributes tab in the Properties window for the MMCM cell you placed.

   Notice the `IS_BEL_FIXED` and `IS_LOC_FIXED` attributes, reflecting that the object has been placed.

Tip: *The Instance Drag and Drop mode in the Device window determines whether only* `IS_LOC_FIXED` *is set, or* `IS_BEL_FIXED` *is also set, when placing objects. Refer to the Vivado Design Suite User Guide: Using the Integrated Design Environment (UG893) for more information on using the Device window.*

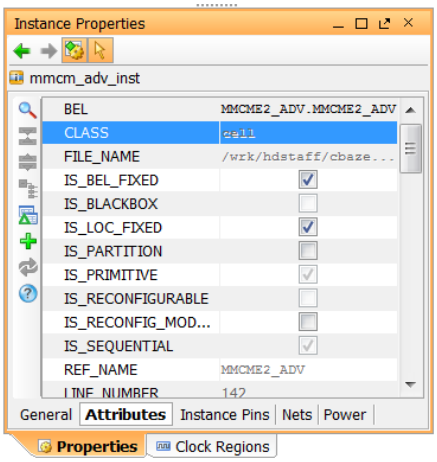
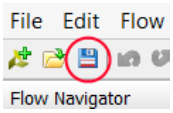


Figure 25:BEL and LOC Placement Constraints

The `IS_BEL_FIXED` and `IS_LOC_FIXED` attributes on the object are physical constraints reflecting the placement of the object. These constraint are used by Vivado implementation, and will not be changed by the tool. However, if the attributes are invalid, they will cause errors downstream in the design flow.



Notice that when you place the `mmcm_adv_inst` in the Device window, the Save Constraints icon is enabled. The physical constraints are added to the Vivado tool in-memory design, but are not yet saved to the target constraint file.

Step 3: Defining Additional Physical Constraints

In this step you will define additional physical constraints to the design, such as the `PACKAGE_PIN`, and `PROHIBIT` constraints.

1. Select the **I/O Planning** view layout from the Layout Selector in the tool bar menu.

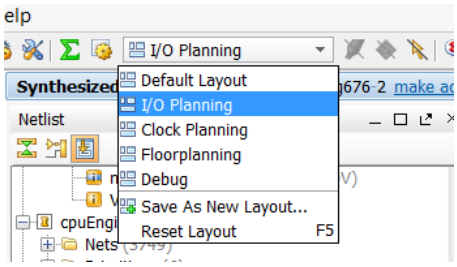


Figure 26: Layout Selector

The I/O Planning view layout displays the Package window, as well as the I/O Ports and Package Pins windows, to facilitate planning the I/O port assignment for the design.

For the purposes of this tutorial, assume the PCB layout has been completed, and therefore certain pins are not accessible on the FPGA package. You can prohibit the Vivado tool from using these pins during placement and routing (assuming you have not already specified all of your I/O assignments).

2. Select the **AA8** pin in the Package window.

   Use the X and Y-axis values, on the edge of the Package window, to help you locate this pin on the package.

3. With the pin selected, right-click and select **Set Prohibit**.

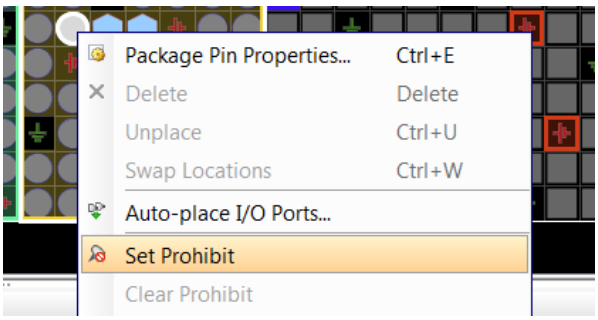


Figure 27: Set Prohibit

When you unselect the pin, you will notice the site now has a red circle with a diagonal line through it, indicating it is unusable.

4. Look in the Tcl Console and review the TCL command produced by the Vivado IDE:

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

Step 4: Defining Constraints with Cell Properties

You can create timing and placement constraints as you have seen in this tutorial, but you can also change the properties of cells, to control how they are handled by Vivado implementation. Many physical constraints are defined as properties on a cell object.

For example, if you discover a timing issue with a RAM in the design. You can change a property of the RAM cell to add in pipeline registers. After confirming with the designer and validation teams that this is an acceptable approach, you can change the design.

Because it might be too costly to go back to the RTL after synthesis, you can make the change in the netlist as follows.

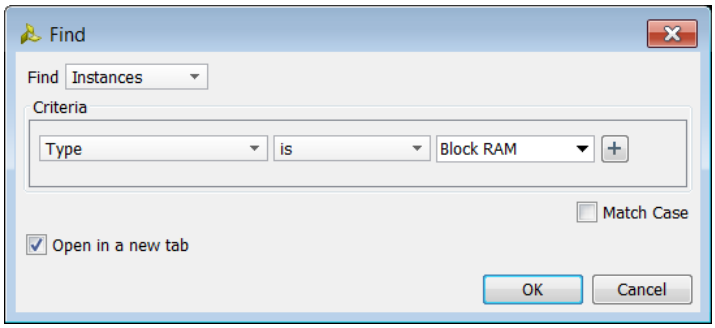


Figure 28: Find dialog box

1. Select **Edit > Find** to open the Find dialog box, as shown in *Figure 28*, and search for Block RAMs.

    a. Specify Find **Instances.**

    b. Under Criteria, specify **Type is Block Ram**

    c. Click **OK**

    The Find Results window displays.

2. Select the first reported cell, which should be the RAMB36E1 cell:

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

    In the Attributes tab of the Instance Properties window, you can see the DOA_REG and DOB_REG are set to zero, indicating that the output registers are disabled.

3. Generate a custom timing report from this cell, either from **Tools > Timing > Report Timing** or directly from the Tcl Console.

    The Tcl command is:

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

Tip: *You can copy and paste the cell name from the General tab of the Instance Properties window into the Tcl console.*

    In the data path section of the report, 1.800ns is added for this RAM.

```
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. In the Attributes tab of the Instance Properties window, select the **DOA_REG** and **DOB_REG** properties for this cell, and change their values from "0" to "1".

5. Click the **Apply** button in the Instance Properties window.

    You can see the two `set_property` commands run in the Tcl Console.

Because these are properties on objects, and not directly defined as timing constraints, you must update the timing data in the Vivado tools, and rerun the timing report, to see these property changes have an effect.

6. Update the timing by typing the following in the Tcl Console:

   ```
   update_timing -full
   ```

7. Rerun the timing report from the selected cell. The Tcl command will be:

   ```
   report_timing -from [get_selected_objects]
   ```

   Notice that the data path delay for the RAM is now 0.622ns.

Next, set the configuration mode on the design. This is another property that is a physical constraint, in this case of the design rather than of a cell. To begin, list all of the properties on the current design.

1. List the properties of the design in the Tcl Console:

   ```
   list_property [current_design]
   ```

   This command returns the list of all properties in the current design. To make the list more readable, you can use the standard Tcl `join` command to combine the properties output with "\n" newline character, resulting in each property displaying on a separate line.

   ```
   join [list_property [current_design]] \n
   ```

2. The specific property of interest is `CONFIG_MODE`. To see what values this particular property can accept, use the `list_property_value` Tcl command:

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

3. Set the CONFIG_MODE property to M_SERIAL for this project:

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

   The configuration mode has now been set.

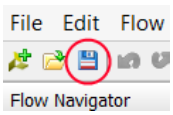
4. Use the get_property command to check that the CONFIG_MODE property was correctly set:

   ```
   get_property CONFIG_MODE [current_design]
   ```

   The property value `M_SERIAL` is returned by the Vivado tool.

Step 5: Saving Constraints

Notice that the Save Constraints icon is enabled because there are new design constraints. The cell and design properties you modified in Lab #2 have been added to the Vivado tool in-memory design, but are not yet saved to the target constraint file.



1. Click the **Save Constraints** button.

   The physical constraints you defined over the course of this tutorial are saved to the target constraint file.

2. Select the target XDC from the active constraint set in the Sources window, to open the file in the Vivado IDE text editor.

   Note: The specific target constraint file you open depends on whether you continued from Lab #1, or restarted the Vivado IDE at Lab #2.

   Notice that only the five `set_property` commands are saved to the files. Only constraints are written to the XDC, not the object query or reporting commands.

   - `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. **Exit** the Vivado IDE.

Summary

In this lab, you learned how to use both the Vivado IDE and the Tcl Console to create and verify physical constraints. Most actions performed in the IDE result in Tcl commands being run in the Tcl Console. The Vivado IDE provides powerful interactive capabilities for developing physical and timing constraints, which can then be saved to constraint files and reused as needed.