Vivado Design Suite Tutorial

Design Analysis and Closure Techniques

UG938 (v2014.1) April 2, 2014

XILINX
ALL PROGRAMMABLE™

Notice of Disclaimer

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx’s limited warranty, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx’s Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

©Copyright 2012-2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, Vivado, ISE, Kintex, Spartan, UltraScale, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

Date	Version	Revision
04/02/2014	2014.1	Validated with release.

Table of Contents

Revision History 2
Table of Contents 3
Vivado Design Analysis and Closure Techniques Tutorial 5
Overview 5
Tutorial Objectives 5
Tutorial Design Description 6
Hardware and Software Requirements 6
Preparing the Tutorial Design Files 7
Lab 1: Design Analysis 8
Step 1: Opening the Example Project 8
Step 2: Viewing the Device Resources and Clock Regions 10
Examining Sites and BELs 13
Examining Clock Regions 14
Step 3: Exploring the Logical Hierarchy 17
Cross probing RTL Source Files 18
Step 4: Displaying Design Resources Statistics 19
Viewing the Resource Estimates of the Entire Design 19
Examining Resources in More Detail 20
Step 5: Performing Pre-Placement Timing Analysis 21
Reporting the Timing Summary 21
Step 6: Implementing the Design 24
Examining the Constraints File 24
Defining Implementation Runs 25
Launching Runs 27
Conclusion 30
Lab 2: Timing Closure Techniques 31
Step 1: Analyzing Post-Implementation Timing 31

Viewing Timing Paths 32
Viewing the Routed Path 34
Viewing Path Properties 35
Step 2: Exploring the Hierarchy of the Timing Paths 36
Step 3: Highlighting Module Placement and Connectivity 40
Visualizing the Connectivity 41
Step 4: Floorplanning the Design 43
Automatically Creating Pblocks 43
Placing Pblocks 44
Viewing Pblock Connections 46
Finding a Global Clock 47
Moving Pblocks 48
Clearing Placement LOCs 50
Resizing Pblocks 52
Step 5: Running Implementation with the Floorplan 56
Conclusion 58

Vivado Design Analysis and Closure Techniques Tutorial

IMPORTANT: *This tutorial requires the use of the Kintex®-7 family of devices. You will need to update your Vivado tools installation if you do not have this device family installed. Refer to the Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973) for more information on Adding Design Tools or Devices.*

Overview

This tutorial introduces some of the capabilities and benefits of using the Xilinx® Vivado® Design Suite for designing high-end FPGAs. The document contains a series of step-by-step exercises that highlight methods to achieve and maintain better performing designs in less time. The steps detail the following:

- Pre-implementation design and analysis capabilities
- Implementation exploration features
- Floorplanning implementation results

VIDEO: *You can also learn more about design analysis in the Vivado Design Suite by viewing the quick take video at http://www.xilinx.com/training/vivado/analyzing-implementation-results.htm.*

Tutorial Objectives

This tutorial takes you through the various analysis, floorplanning, and implementation features of the Vivado Design Suite. When using this tutorial, you should focus on the demonstrated techniques and how they relate to your own designs. The exercises cover the following topics:

- Analyze device utilization to target alternate devices and choose the optimal device.
- Run Design Rule Checks (DRC) to quickly resolve constraint conflicts and design errors.
- Use the Netlist, Logic Hierarchy, and Schematic windows to explore logic.
- Perform a quick estimation of timing performance to assess design feasibility and identify potential problem areas.
- View, modify, or create constraints in the design.

- Analyze the design hierarchical connectivity and data flow, as well as identify critical logic connectivity and clock domains.
- Floorplan timing-critical logic to improve timing and data flow.
- View Routing Resources in the detailed Device window

Tutorial Design Description

The sample design used throughout this tutorial consists of a small design called `project_cpu_hdl`, which targets an xc7k70t device. The small sample design used in this tutorial includes:

- A RISC processor CPU core
- A pseudo FFT
- Four gigabit transceivers (GTs)
- Two USB interfaces

A small design is used to enable timely completion of the synthesis and implementation runs with minimum hardware requirements.

Hardware and Software Requirements

This tutorial requires that the 2014.1 Vivado Design Suite software release or later is installed. The following partial list describes the operating systems that the Vivado Design Suite supports on x86 and x86-64 processor architectures:

Microsoft Windows Support:

- Windows 8.1 Professional (32-bit and 64-bit), English/Japanese
- Windows 7 and 7 SP1 Professional (32-bit and 64-bit), English/Japanese

Linux Support:

- Red Hat Enterprise Workstation 6.4 and 6.5 (32-bit and 64-bit)
- SUSE Linux Enterprise 11 (32-bit and 64-bit)
- Cent OS 6.4 and 6.5 (64-bit)

Refer to the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973)* for a complete list and description of the system and software requirements.

Preparing the Tutorial Design Files

This tutorial uses the example projects provided with the Vivado Design Suite, accessible from the Getting Started page of the Vivado IDE, as seen in Figure 1.

The example projects are read-only by default, and must be saved to a new location for your use in working through this tutorial. The location you save the project to is referred to throughout this tutorial as <*Extract_Dir*>.

You can also find the files for the example projects in the Vivado Design Suite examples directory at the following location:

- **<Vivado_install_area>/Vivado/<version>/examples/Vivado_Tutorial**

You can extract the provided `Vivado_Tutorial.zip` file, at any time, to write the tutorial files to a local directory.

Lab 1: Design Analysis

Step 1: Opening the Example Project

On Linux,

1. Change to the directory where the lab materials are stored:

   cd <Extract_Dir>/Vivado_Tutorial

2. Launch the Vivado IDE: **vivado**

On Windows,

1. Launch the Vivado Design Suite IDE:

   Start > All Programs > Xilinx Design Tools > Vivado 2014.x > Vivado 2014.x[1]

2. As an alternative, click the **Vivado 2014.x** Desktop icon to start the Vivado IDE.

3. From the Getting Started page, click **Open Example Project** and select the **7-Series Kintex > CPU (HDL)** design.

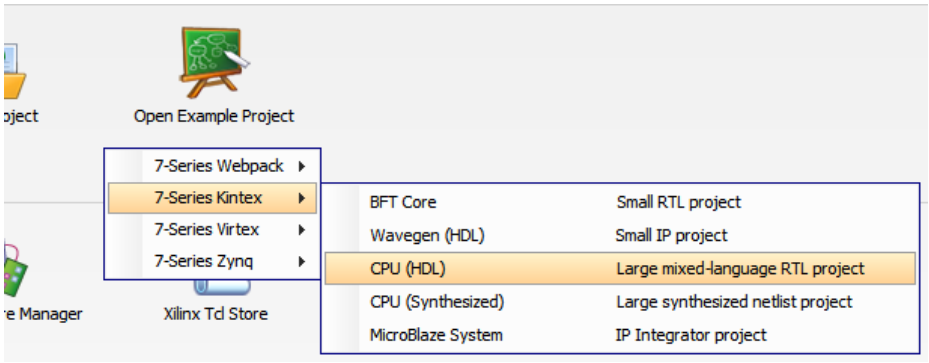


Figure 1: Open Example Design

   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.

4. Select **Save Project As** and specify:
   - Project name: **project_cpu_hdl**
   - Project location: ***<Extract_Dir>***

[1] Your Vivado Design Suite installation may be called something other than **Xilinx Design Tools** on the **Start** menu.

The project is saved to the specified location. If the specified directory does not exist, the Vivado Design Suite will create the directory as needed.

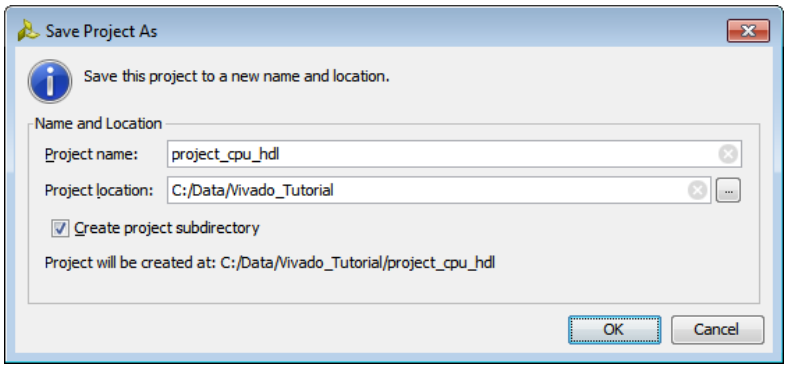


Figure 2: Save Project As...

The Vivado IDE displays project information in the Project Summary window. The Project Summary shows the project settings, and the run details for both the synthesis and implementation runs.

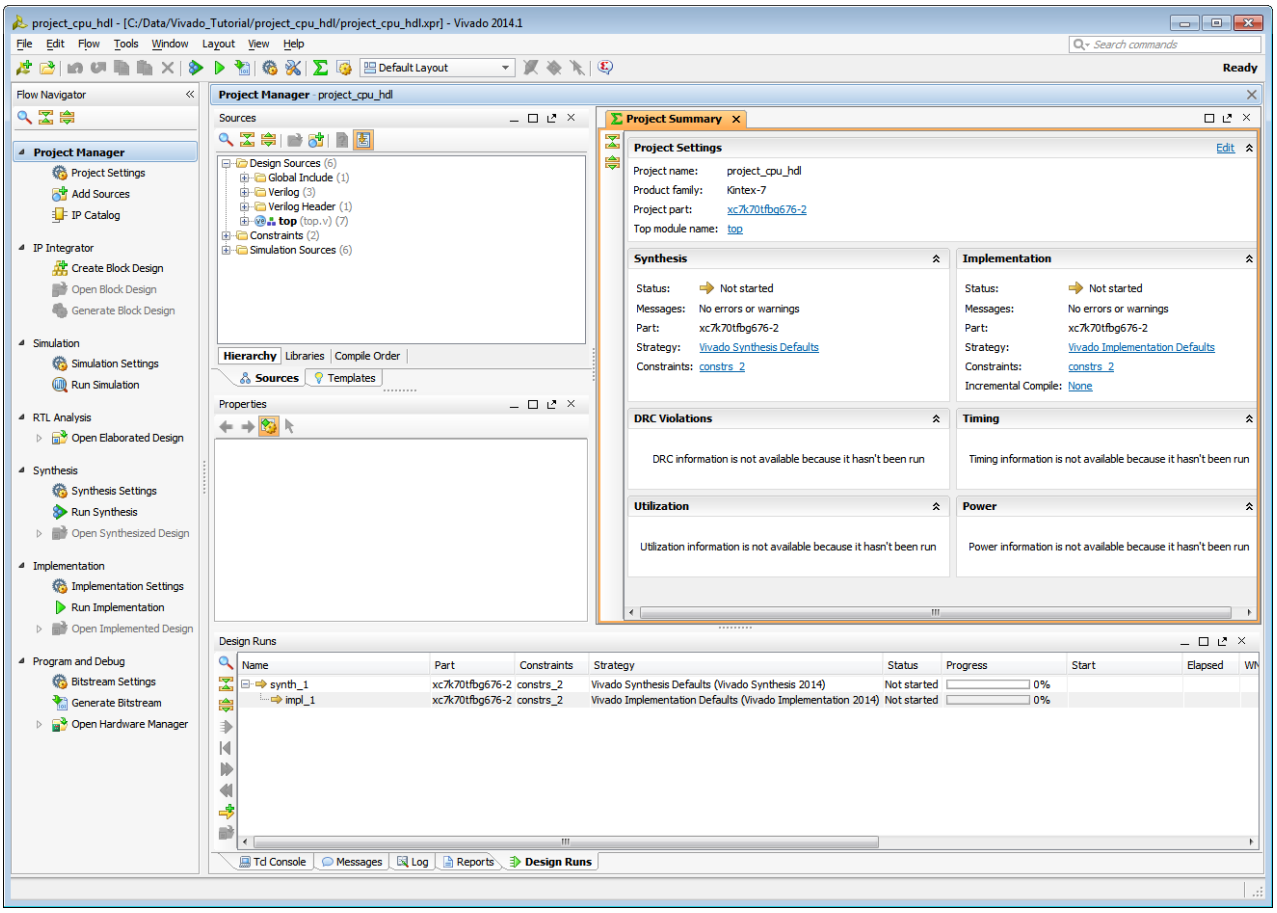


Figure 3: Vivado Environment

Step 2: Viewing the Device Resources and Clock Regions

1. In the Sources window, **expand** the **Constraints** folder to ensure constrs_2 is active, as shown in Figure 4.
2. If needed, select **constrs_2**, right-click and select **Make Active** from the popup menu.

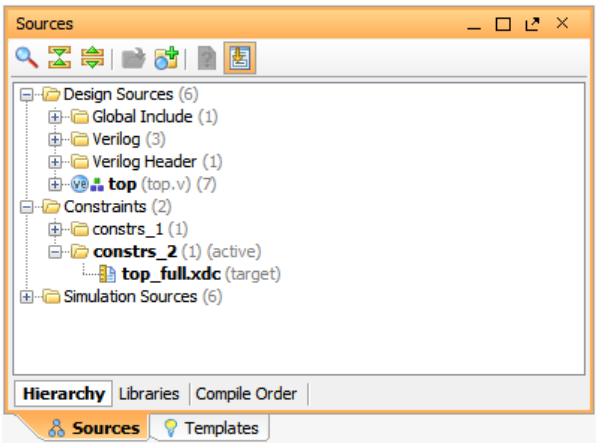


Figure 4: Sources window

3. **Click** the **Design Runs** window and examine the runs information.

   The Design Runs window offers commands to manage, launch, and reset runs for both synthesis and implementation. You can manage multiple runs in the Design Runs window.

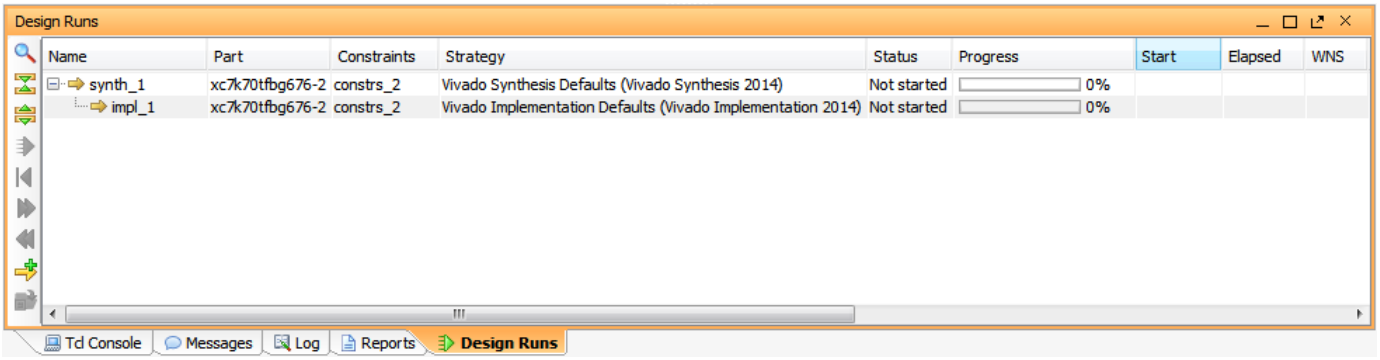


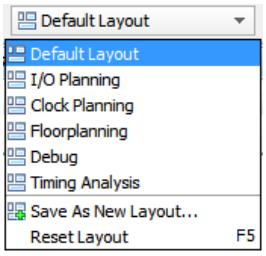
Figure 5: Design Runs window

4. In the Flow Navigator, select **Run Synthesis** to synthesize the design.

   Watch the Log window for run status. It takes several minutes for the run to complete.

5. After synthesis completes, select **Open Synthesized Design** from the Synthesis Completed dialog box and click **OK**.

   The Synthesized Design opens with the netlist window as seen in Figure 6.



6. In the Layout Selector, in the main toolbar, change the window layout to the **Default Layout** if that is not the current window layout.
7. **Click** on the **Device** window to make it active.

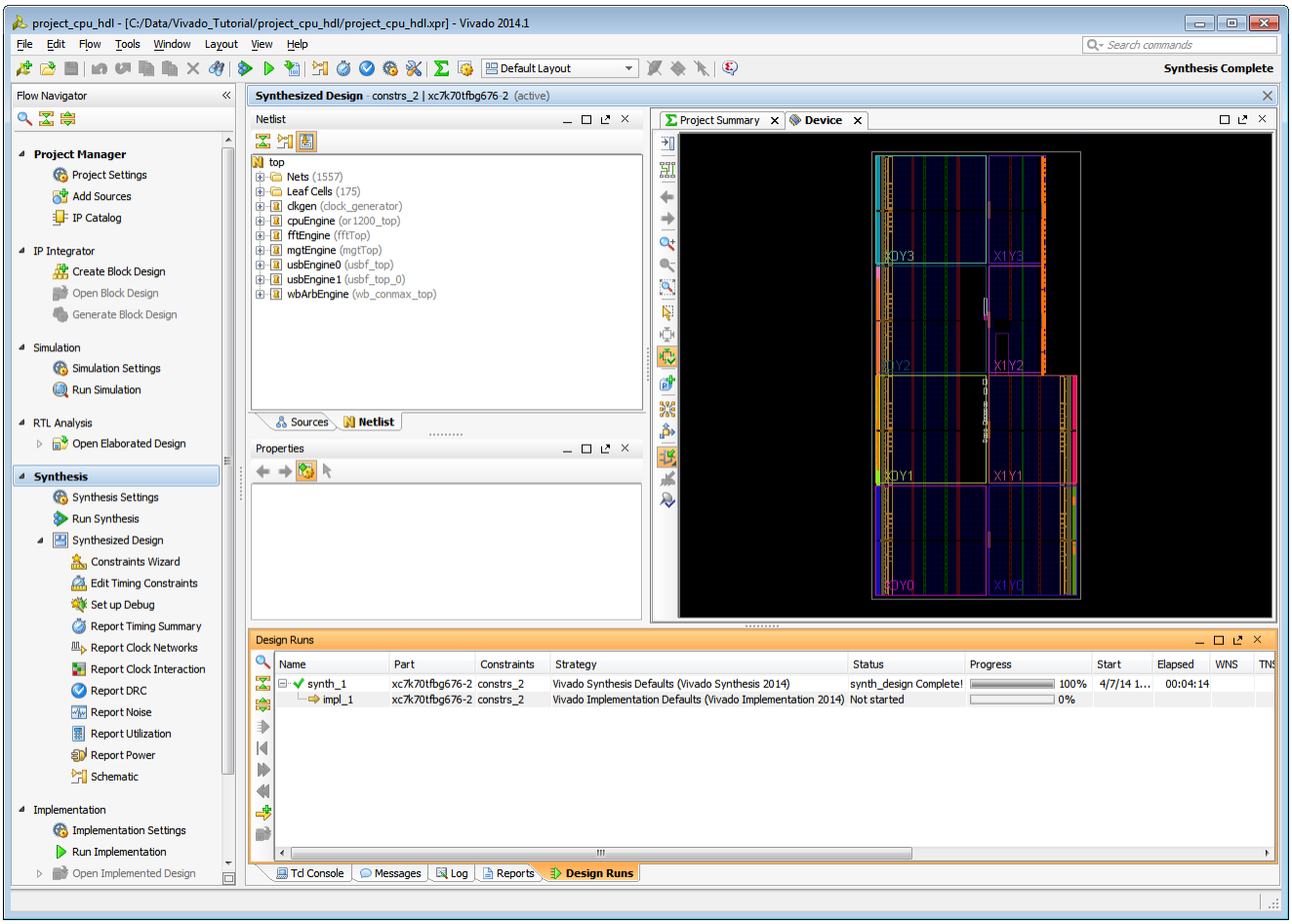


Figure 6: Open the Synthesized Design

8. In the Device window, use **Zoom Area** by clicking and holding the left mouse button, while dragging down and to the right.

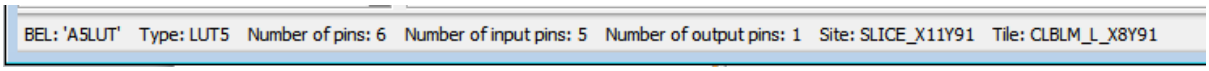
   This will draw a Zoom Area rectangle. Repeat as needed to zoom into the device close enough to see the different device resources.

In the Device window, the Vivado IDE displays the FPGA device resources that can be used to implement the design, as a matrix of tiles. Sites are available within a tile for placement of netlist instances. The available sites include, for example, SLICEs, RAMs, MULTs, and DSPs, and vary in shape and color in the Device window to differentiate the object types contained in the site. Slices are fundamental building blocks of Xilinx FPGAs, containing LUTs and registers, though the specific contents of a slice may vary with the device family.

"LOC" and "BEL" placement constraints are used to assign logic to either a site or a slice. Leaf cells can be assigned specifically to a site, or generally to a slice with a "LOC" placement constraint, or can be assigned to specific logic resources within a slice using both "LOC" and "BEL" constraints.

Figure 7: Viewing Sites, Slices and BELs in the Device window

9. Move the cursor slowly over the various device resources to view the tooltips, and take note of the following:
    - Hover the mouse over a site to show a tooltip.
    - I/O port locations and I/O buffer assignments appear inside I/O banks.
    - Tall dark red[2] tiles indicate a RAMB36 site that can also hold two RAMB18s or a FIFO.
    - Tall green tiles indicate sites for DSP48s.
    - Square blue tiles are a CLB containing SLICEs.
    - SLICE coordinates appear in the status bar at the bottom left of the Vivado IDE main window.



Examining Sites and BELs

Below the Netlist window, the Site Properties or BEL Properties window is displayed. These windows appear when a site or bel is selected within the Device window, or some other window.

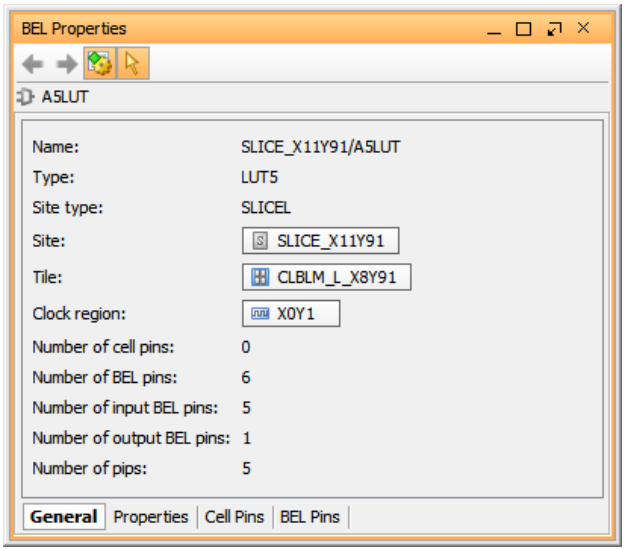


Figure 8 : BEL Properties window

1. Select several different types of resources within the Device window, and examine the Site and BEL Property information.

[2] The colors mentioned here are for the Vivado IDE default color scheme. Your colors may be different.

2. Use the **Zoom Fit** command from the Device window toolbar menu, or click and drag the cursor up and to the left in the Device window, to see the whole device.
3. Click the **Auto-fit Selection** button on the Device window tool bar menu, to zoom and center any objects selected in the next section.

Examining Clock Regions

The Device window shows the clock regions on the device, which is useful to help guide floorplanning and placement. The Clock Region window lists all the clock regions in the device. The Clock Region Properties window displays clock region information to indicate potential clock contentions prior to implementation. Viewing clock domains is covered later in the tutorial.

1. Select **Window > Clock Regions** to display the Clock Regions window.
2. **Select** Clock Region **X0Y2** listed in the Clock Regions window.

   The clock region is highlighted in the Device window, and the Clock Regions Properties window is displayed.

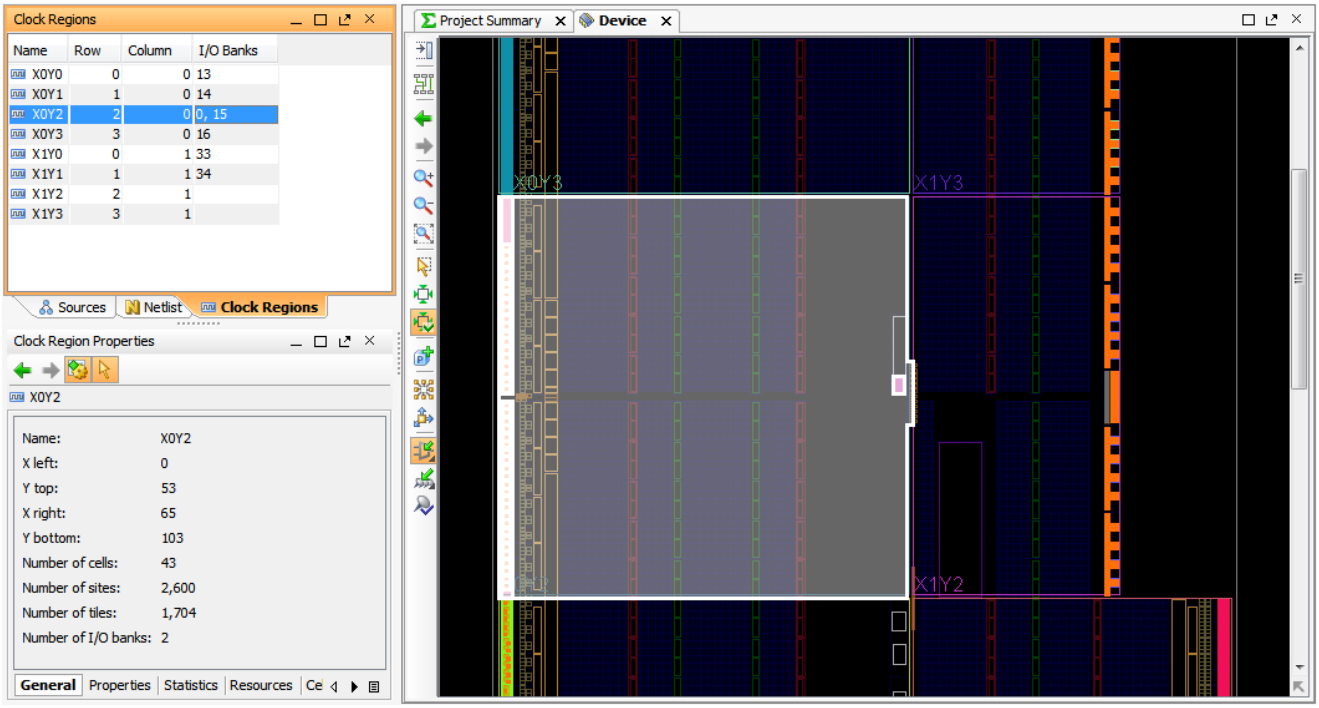


Figure 9: Viewing Clock Region Properties

3. Click the Properties tab to examine the **Clock Region Properties** window.
4. Click the **Statistics** tab of the Clock Region Properties, and scroll to examine the available device resources of the selected clock region.

5. Click the **Resources** tab of the Clock Region Properties, and select and examine the locations of the BUFR and IDELAYCTRL sites.
6. **Select** one of the **BUFRs**, and notice it is highlighted in the Device window.
7. In either the Device window or the Properties window, right-click and select **Mark** to mark the selected BUFR site.
8. On the sidebar menu of the Device window, click **Zoom Fit**, , to view the whole device.

   Notice the mark on the BUFR allows you to quickly locate the site in the zoomed out Device window.
9. Select the **Unmark All** button on the main tool bar menu to remove the mark.
10. In the Clock Region Properties window, click the **I/O Banks** tab to examine the I/O Banks in the selected clock region.
11. In the Clock Region Properties window, **select** one of the **I/O banks**.

TIP: *To locate the I/O Banks tab, you may need to scroll the Clock Region Properties window using the displayed arrows.*

The selected I/O bank is highlighted in any open windows, such as the Device window, the I/O Ports window, and the Package and Package Pins windows. You can open and switch between these different windows to see the various representations of the selected I/O bank.

12. Select **Window > Clock Resources** to open the Clock Resources window, seen in Figure 10.

    In the Clock Resources window, the Vivado IDE displays a simplified view of the I/O and clocking resources in a spreadsheet-like interface. This window shows the relationship and interactions between regional and global clocking resources on the device: BUFRs, BUFIOs, BUFGs, MMCMs and GTs, while maintaining proper relative positioning between these resources. You can use the Clock Resources window to plan and place critical clocking and I/O logic prior to placing and routing the rest of the design.

    In the target part for this design, the Clock Resources window shows eight clock regions, in a 2x4 matrix, numbered from X0Y0 in the lower left to X1Y3 in the upper right. The device is divided into a top and a bottom half, which can be expanded and collapsed as needed to further simplify the display. Each clock region displays I/O banks containing clock-capable pins (CCIOs), BUFIOs, and BUFRs. In the center column of the device are global clock buffers (BUFGs) for managing global clocks on the device. The clock regions on the right side of the device have either an I/O bank or a column of gigabit transceivers (GTs).
13. **Select** a global clock buffer (**BUFG**) in the Clock Resources window and see it highlighted in the other open windows.

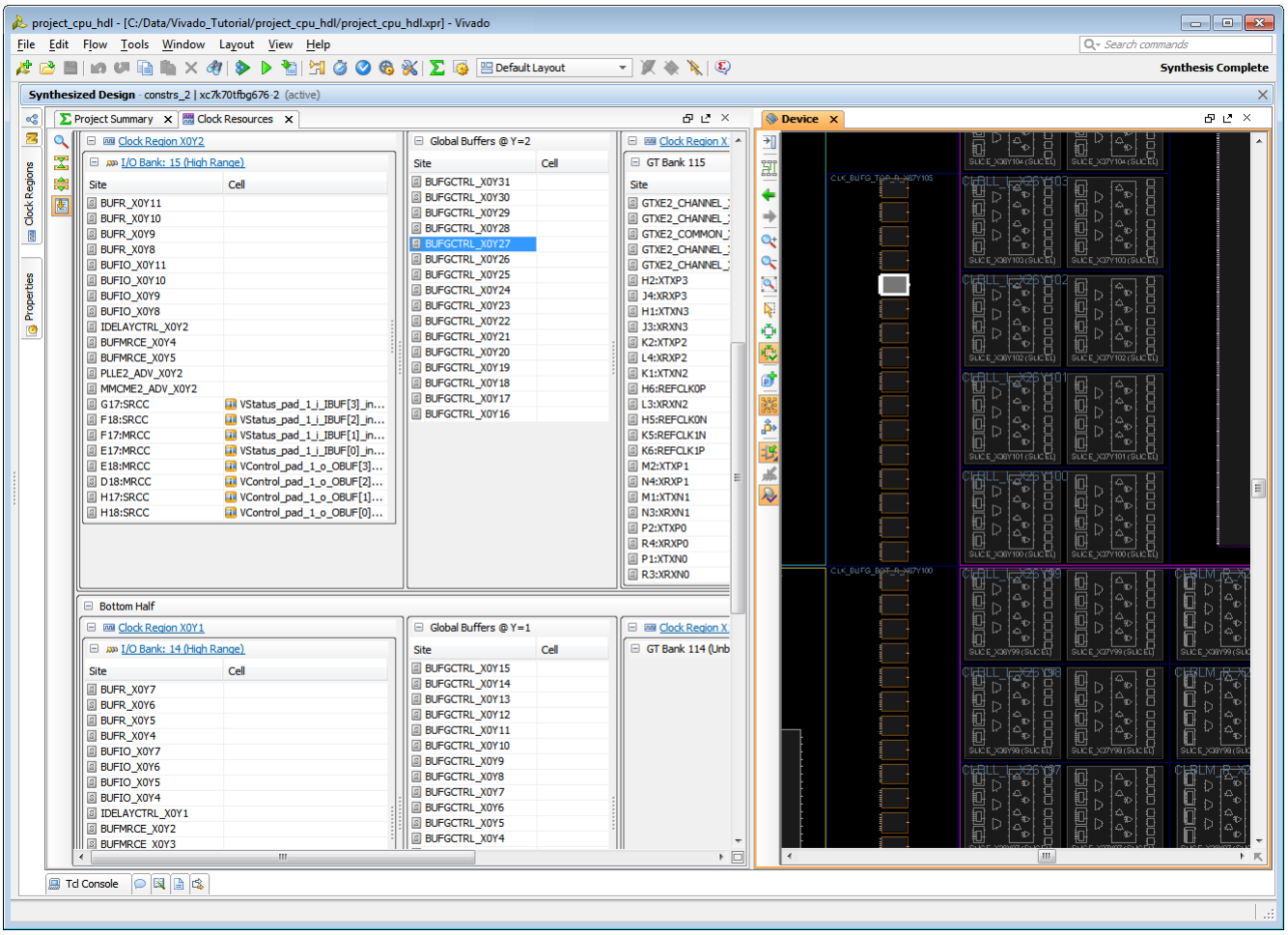
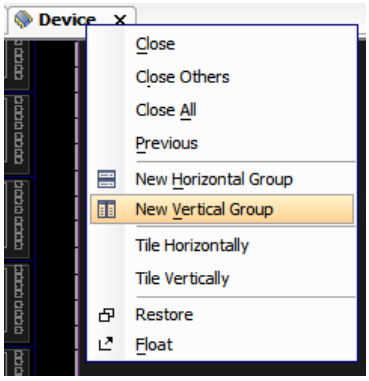


Figure 10: Clock Resources and Device windows

In Figure 10, the Clock Resources window is maximized in the Vivado IDE, and the Device window is open in a new vertical window group.

You can arrange windows in this manner using the Tile Vertically, Tile Horizontally, or the New Vertical Group, New Horizontal Group commands. These commands can be found by right clicking, to open the popup menu, in the window tab of a graphical window, such as the Device window or Clock Resources window. See *Vivado Design Suite User Guide: Using the Vivado IDE (UG893)* for more information.



14. **Close** the **Clock Resource** window and **Clock Regions** window by clicking the **X** in the tab.

Step 3: Exploring the Logical Hierarchy

1. Click the Netlist window tab, and if needed, click the **Collapse All** button .
2. Click the plus sign next to the `usbEngine0` module to examine the hierarchy.

   The Netlist window should now look like Figure 11.

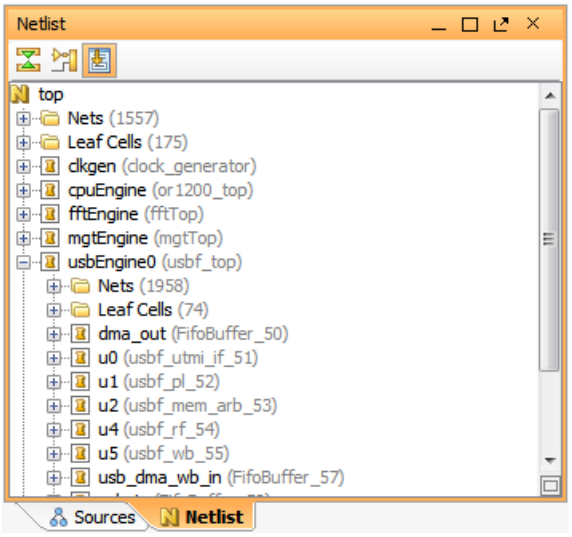


Figure 11: Expanding the Netlist Window

3. Click the plus sign next to the **Nets** folder of the `usbEngine0`.

   These are the nets inside the `usbEngine0` module. You can collapse the Nets folder by clicking the '**-**' next to it.
4. Click the plus sign next to the **Leaf Cells** folder of `usbEngine0` to expand and examine the contents.

   These are the leaf cells inside the top-level of the `usbEngine0` module, not inside any of the sub-modules. There is also a Leaf Cells folder at the top level of the Netlist window that contains the leaf cells in the top-level of the design.
5. **Select** the **u4** module.
6. Right-click and select **Show Hierarchy**, or press **F6**.

   The Hierarchy window opens in the Workspace. The Hierarchy window is used primarily during design analysis and floorplanning. The Hierarchy window displays the hierarchical relationship of the modules. The width of the blocks in the Hierarchy window are relative to the device resources consumed by each block. It lets you see how a timing path traverses the logic hierarchy, or gauge how big a module is before floorplanning the module.

   You can select modules directly from the Hierarchy window for floorplanning. Logic selected in other windows is highlighted in the Hierarchy window.

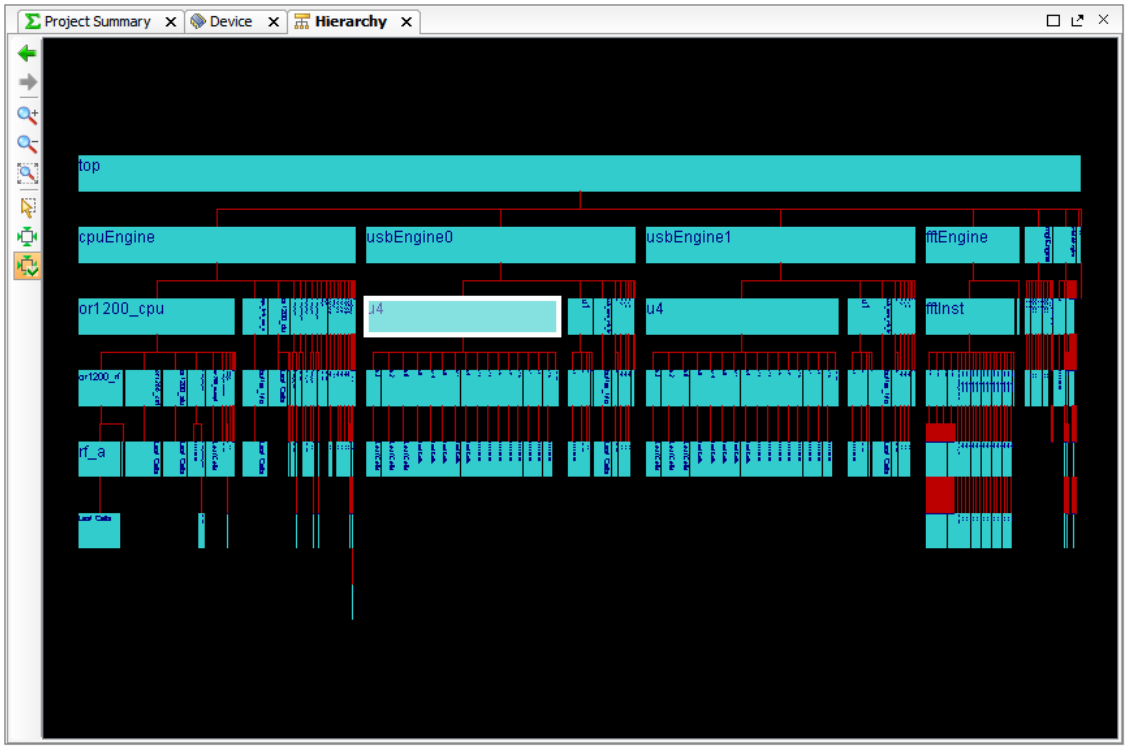


Figure 12: The Netlist and the Hierarchy Windows

7. **Select** any **module** in the **Hierarchy** window, and notice that the module is selected in the Netlist and other windows, as well. Selection works across windows.
8. Click the **Unselect All** main toolbar button, or press **F12**.

Cross probing RTL Source Files

1. **Select** the **u4** block in the Hierarchy window, or in the Netlist window.
2. Right-click and select **Go to Source** from the popup menu, or press **F7**.

   Notice the RTL source file is now open in the Text Editor, with the line number highlighted where the logic instance was derived.

   You can use this feature, and the Go to Instantiation command, to quickly locate the definition of a module, or a specific instantiation of a module in the RTL source files.

TIP: *The Text Editor displays some file formats with context-sensitive highlighting to make it easy to identify keywords and comment lines within the file.*

3. **Select** several other **logic instances** in either the Netlist or Hierarchy window and view the RTL source files.
4. **Close** all open RTL source files in the **Text Editor**, and **close** the **Hierarchy window**.
5. Click the **Unselect All** main toolbar button, or press **F12**.

Step 4: Displaying Design Resources Statistics

The Vivado Design Suite utilization analysis provides design statistics to help you determine the optimal device to fit the design. The tool helps you see how the logic resources are split between modules. It is easy to explore multiple device types to determine the best overall utilization and performance estimations.

Viewing the Resource Estimates of the Entire Design

1. In the Flow Navigator, under Synthesized Design, select **Report Utilization** and click **OK** in the Report Utilization dialog box to accept the defaults.
2. Scroll down the **Utilization** window.

   The Utilization window shows device resource utilization sorted by logic type on the left side, and displayed according to the design hierarchy on the right side.

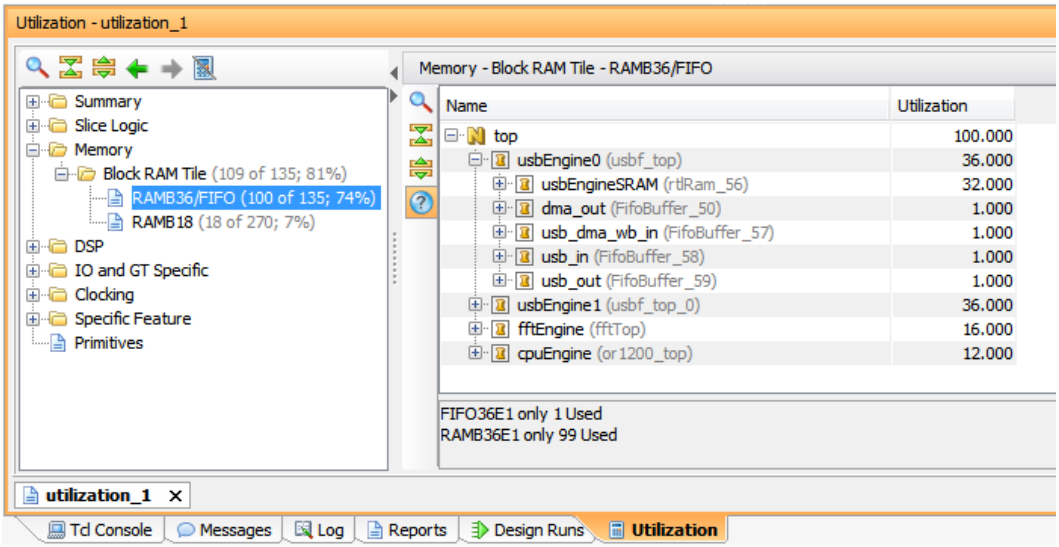


Figure 13: Resource Utilization window

3. In the Utilization window, click **Memory > Block RAM Tile > RAMB36/FIFO**

   In the left side of the window you will notice that the overall utilization of Block RAM 36 is 74%, or 100 of 135 available units. In the right side you can see that the usbEngine blocks consume 36 each, or 72 out of the 100 utilized Block RAMs.
4. Click the plus sign next to the `usbEngine0` to expand and examine the hierarchy.
5. Select **usbEngine0** in the right side of the Utilization window.

   Notice that `usbEngine0` is also selected in other open windows, such as the Netlist window.
6. **Close** the **Utilization** window by clicking the X button in the upper right of the window.

Examining Resources in More Detail

The Pblock Properties window offers you another way to view the device resources, and the resource utilization by the design.

1. Select **Window > Physical Constraints** to open the Physical Constraints window.
2. In the Physical Constraints window, **select** the **ROOT** design.

   This opens the Pblock Properties window. The Vivado Design Suite provides the ability to hierarchically divide the design into smaller, more manageable physical blocks (Pblocks). Pblocks can include logic modules and cells from anywhere in the design hierarchy.

   The Vivado tool also creates a single Pblock from the root design, or the top-level design, even when there are no other Pblocks defined. In this section you will examine the resources available in the root Pblock.
3. In the **Pblock Properties** window, select the **Statistics** tab.

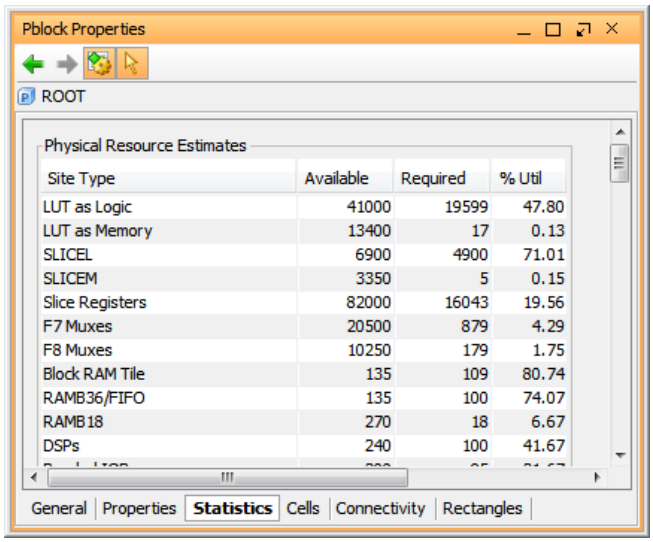


Figure 14: Viewing Design Resource Statistics

4. View the **Physical Resource Estimates** displayed in the Pblock Properties window, as shown in Figure 14.

   - **Available**: The available device sites covered by the Pblock, as drawn.
   - **Required**: The number of resources required by the logic assigned to the Pblock.
   - **% Util**: The estimated utilization of each site type. Red text indicates that utilization exceeds 100%, and the Pblock lacks sufficient resources for the assigned logic.

   The ROOT Pblock includes the entire design placed onto the target Xilinx FPGA. In this case the Pblock Properties statistics reflect the logic requirements of the whole design, and reflect the resources available on the target part.

IMPORTANT: *If the Utilization % is exceeded for the ROOT, you will need to select a different target part.*

5. **Scroll** through the **design statistics** displayed in the Pblock Properties window.

   The Statistics tab displays the device resource utilization for each type of logic element, carry chain count and longest length, clock report, I/O utilization, and leaf cells and I/O statistics. If the design has any Relatively Placed Macros (RPMs), the RPM count and maximum size information are also shown.

6. **Close** the **Physical Constraints** window.

Step 5: Performing Pre-Placement Timing Analysis

In this step, you will investigate timing before implementing the design. Timing analysis is available anywhere in the design flow after synthesis.

At this point, prior to implementation, the Vivado IDE timing engine provides an estimate of what the route delays will be, and does not report on whether the design has met timing. Before running implementation it can be helpful to perform an early Static Timing Analysis, including estimated route delays, to determine the achievability of the timing constraints in the design.

Reporting the Timing Summary

The Timing Summary shows the timing slack on each path. ….

This can assist you in determining your next steps if the design is not meeting performance requirements; helping you focus on critical timing paths with negative or low slack values. Endpoints with negative slack are expected to have timing problems. Endpoints with low slack margins in preliminary timing analysis may not meet timing targets during implementation.

TIP: *The Slack Histogram does not show the complete timing path. You must use the Report Timing command to see the timing of a complete path.*

1. From the Synthesized Design menu of the Flow Navigator **select Report Timing Summary**.

   This opens the Report Timing Summary dialog box, as seen in Figure 15.

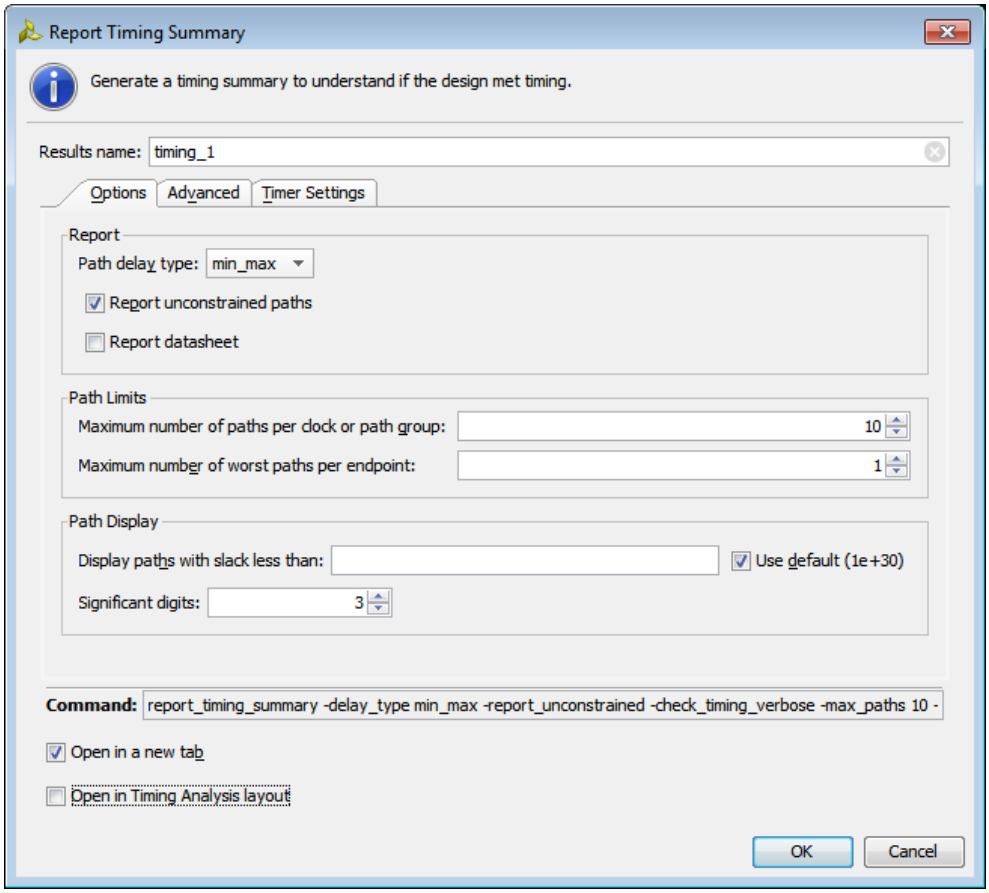


Figure 15: Report Timing Summary

2. **Examine** the options on the **Advanced** and **Timer Settings** tabs.

   You can specify various delay parameters used by the timing engine, to determine how the Vivado tool calculates timing.

   At this point in the design flow, prior to implementation, the Interconnect pull-down in the Timer Settings tab has two values: estimated, and none. Prior to running the Vivado implementation tools, the timer cannot give you actual interconnect delays. The routing delay can be estimated based on a routing delay model, or can be left out of the analysis.

3. Click **OK** to display the Timing Summary report.

   The Timing Summary report consists of two windows as seen in Figure 16: the left hand window groups the returned information into specific types of reports, and the right-hand window displays the selected report. The default view is the Design Timing Summary, which displays the slack information for the Setup, Hold, and Pulse Width timing. See the *Vivado Design Suite User Guide: Design Analysis and Closure Techniques (UG906)* for more information on the Timing Summary Report.

   Notice the Worst reported value for Setup, Hold, and Pulse Width is a link that can be clicked.

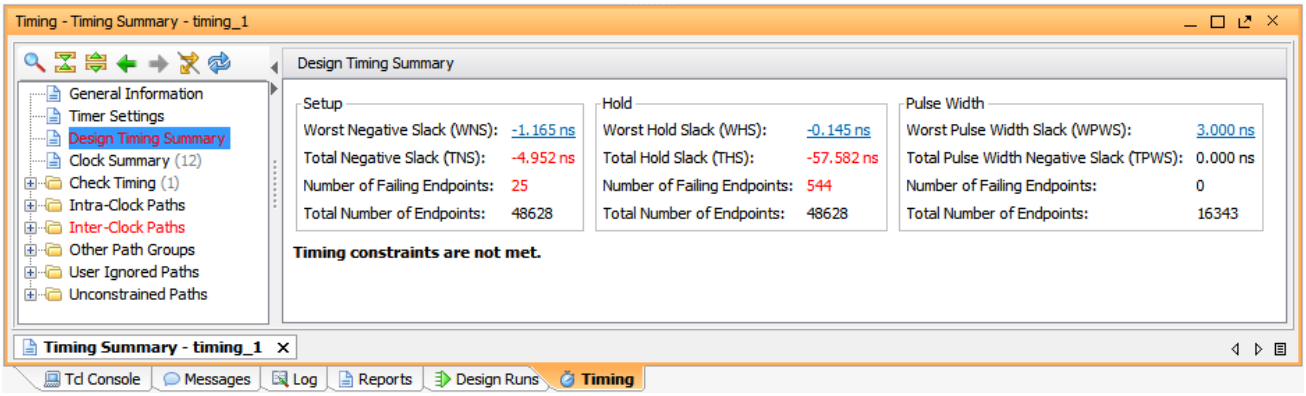


Figure 16: Design Timing Summary

4. **Click** on the **link** for **Worst Negative Slack** (WNS).

   The timing path with the WNS, and other paths in the same path group are displayed as seen below. The report includes columns for the calculated timing slack, start point, endpoint, delay values, and the source and destination clocks. Paths that fail timing appear in red in the report.

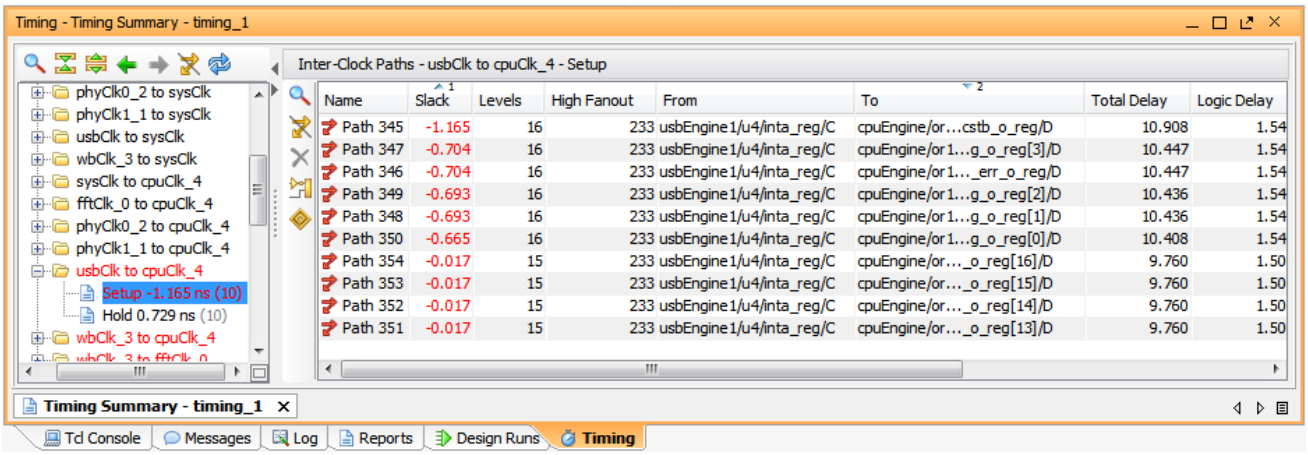


Figure 17: Worst Negative Slack

Scroll down the list of reported timing paths. Notice that most of the timing paths associated with the WNS have start points in the `usbEngine1` module.

5. **Click** the **To** column header twice to sort the list by destination in descending order.

TIP: *Any of the table style windows in the Vivado IDE can be sorted by clicking on the column header; once to sort in ascending order, twice to sort in descending order. To sort multiple columns press* ***Ctrl*** *and select additional column headers.*

6. **Close** the Timing Summary report by clicking the **X** button.

Step 6: Implementing the Design

Vivado implementation is a timing-driven flow with native support of industry standard Synopsys Design Constraints (SDC) commands, and Xilinx Design Constraints format (XDC), to specify design requirements and restrictions. Vivado implementation encompasses all of the design steps required to place and route the netlist onto the FPGA device resources while meeting the logical, physical, and timing constraints of the design.

Important! *The Vivado Design Suite does not support the UCF format. For information on migrating UCF constraints to XDC commands, refer to the ISE to Vivado Design Suite Migration Guide (UG911).*

Examining the Constraints File

1. If the Sources window is hidden from view, display it by selecting **Window > Sources**, or selecting the Sources tab as shown in Figure 18.
2. In the Sources window, click the **Collapse All** button, then expand `constrs_1` and `constrs_2`.

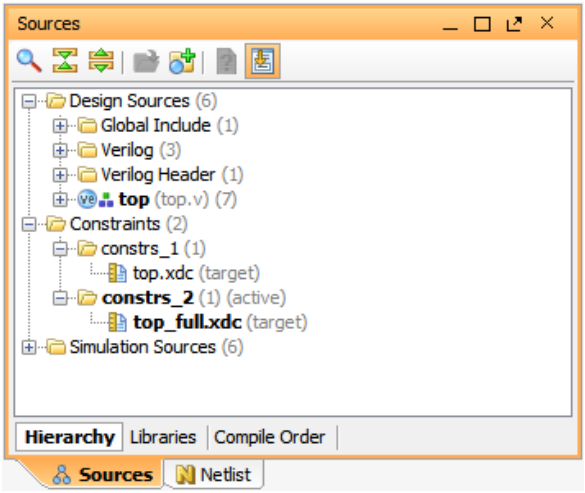


Figure 18: Sources Window

   When running the implementation tools, the Vivado Design Suite uses the active constraint set by default.

   Make sure that `constrs_2`, with `top_full.xdc`, is the active constraint set.

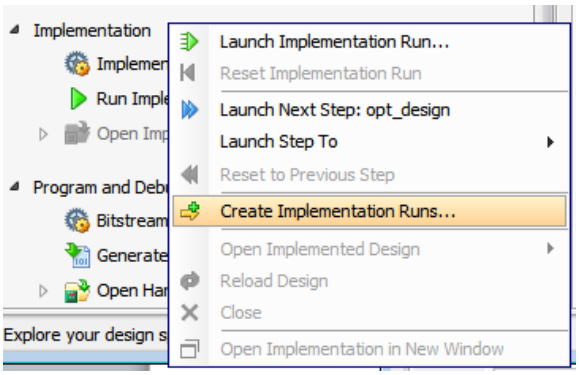
3. Double-click **top_full.xdc** to open the file, and examine the contents of the constraint file.
4. **Close** the XDC file.

   Do not save any changes if prompted.

Defining Implementation Runs

The Vivado IDE allows you to configure and concurrently run multiple synthesis and implementation runs. This tutorial will demonstrate how to do that, but you will only launch one implementation run in order to conserve time.

1. In the Flow Navigator, right-click on Implementation, and click **Create Implementation Runs** in the popup menu.



2. Click **Next** in the Create New Runs confirmation dialog.

   The Configure Implementation Runs dialog box opens, letting you create and configure one or more implementation runs, selecting strategies to use during the run.

   The first implementation run listed should be `impl_2`. The strategy listed will depend on whether you have created any user-defined implementation strategies. See the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on standard available strategies, and on defining new strategies.

3. Click the **Strategy** pull-down menu to select the **Flow_RuntimeOptimized** strategy listed under Vivado Strategies.

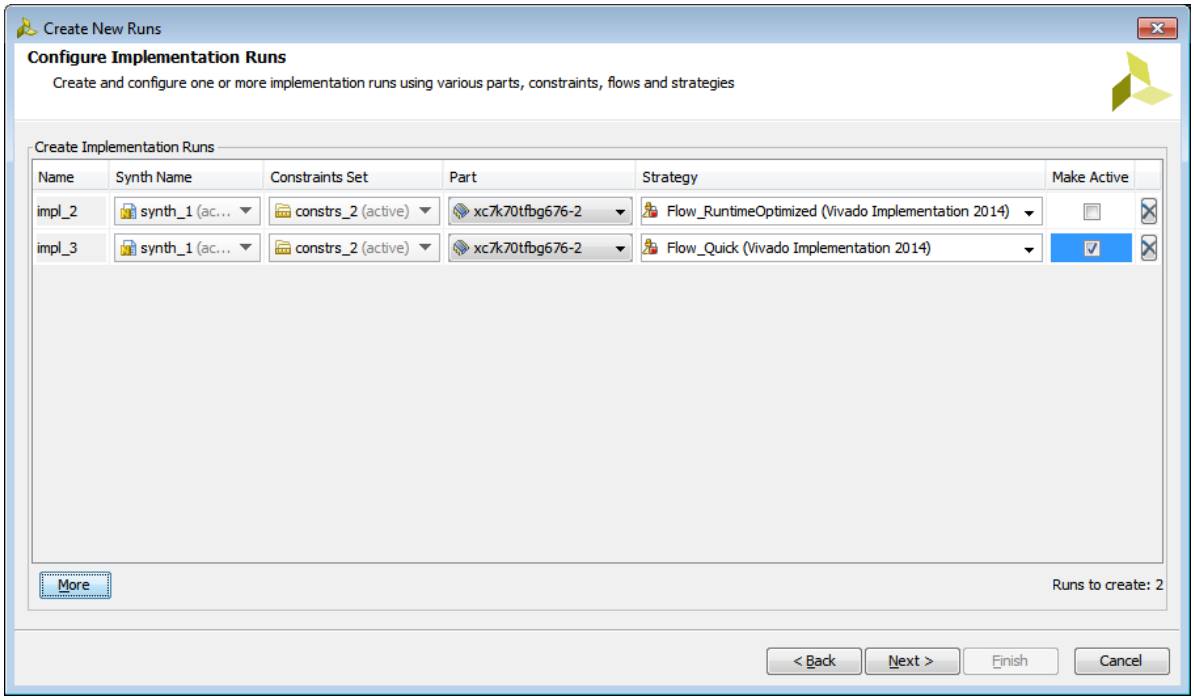


Figure 19: Configure Implementation Runs

4. Click **More** to add a new run, called `impl_3`.
5. Click the **Strategy** pull-down menu to select the **Flow_Quick** strategy.

   This is the implementation run you will be using later in the tutorial. You can also configure Synth Name, Constraints Set, and Part when defining the run. However, at this time, you will use the default settings.
6. **Click** the **Make Active** checkbox for **`impl_3`**, as seen in Figure 19, to make it the active run.

   Only one synthesis and one implementation run can be active in the Vivado tool. By default, the Vivado IDE directs commands to, and displays reports and messages from the active run.
7. Click **Next** to open the Launch Options dialog box.
8. **Review** the available **launch options**. You can:
   - Launch the runs locally to run implementation on the local computer.
   - Launch the runs remotely (on Linux only), to run implementation on remote hosts.
   - Compile run scripts for later use, to create batch scripts that can be run either locally or remotely.
   - Save the runs without launching implementation, which is what you will choose now.
9. Select **Do not launch now**, and click **Next**.
10. **Review** the Create New Runs Summary, and click **Finish**.

    Your Design Runs window should look as follows:

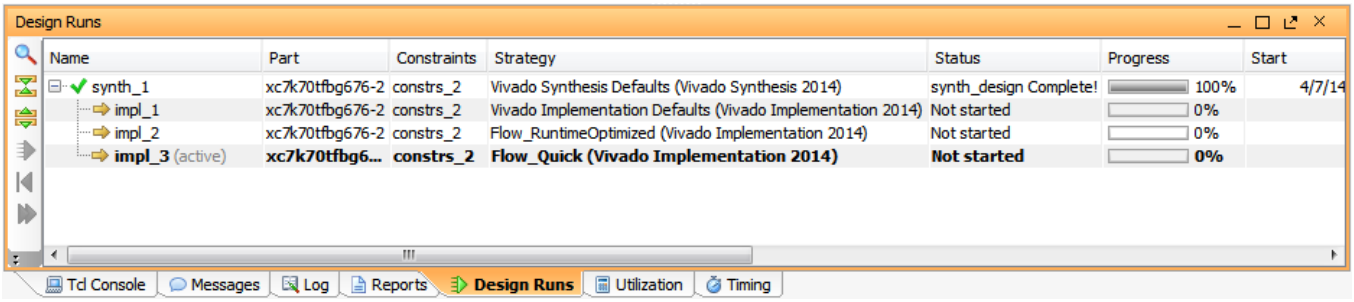


Figure 20: Design Runs window

Launching Runs

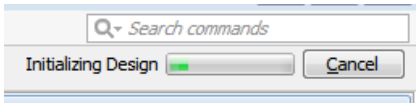
The next step in the flow is implementing the design through the Vivado IDE implementation tools. The Vivado implementation process includes logical as well as physical transformations of the design, and consists of the following sub-processes:

- **Opt Design**: Optimize the logical design to make it easier to fit into the target Xilinx FPGA.
- **Power Opt Design**: Optionally optimize elements of the design to reduce power consumption of the implemented FPGA.
- **Place Design**: Place the design onto the target Xilinx device.
- **Phys Opt Design**: Optionally optimize the timing of the design by replicating drivers of high-fanout nets to distribute the loads.
- **Route Design**: Route the design onto the target Xilinx device.

See the *Vivado Design Suite User Guide: Implementation (UG904)* for more information on these sub-processes of implementation, or the *Vivado Design Suite Tcl Command Reference Guide (UG835)* for information on the specific Tcl commands

1. Select **Run Implementation** in the Flow Navigator.

   Implementation takes several minutes to complete. The status of the implementation run is displayed in the project status bar at the upper right of the Vivado IDE.



   While you are waiting for implementation to complete, you can experiment with some of the analysis features available for use with the synthesized design. Commands such as the Report Clock Networks and Report Clock Interaction can be run from the Flow Navigator on the open synthesized design while implementation is running.

   The Implementation Completed dialog box displays when the run is finished.

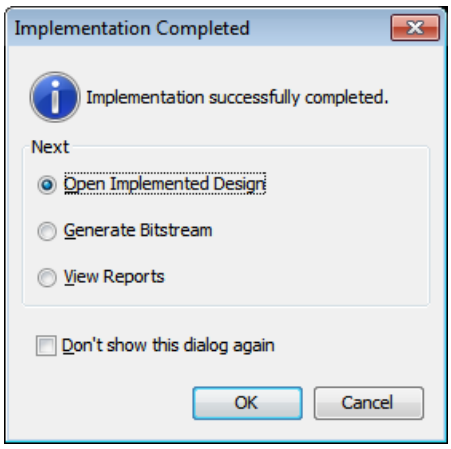


Figure 21: Implementation Completed

2. Select **Open Implemented Design** in the Implementation Completed dialog, and click **OK**.
3. Select **Yes** if prompted to close the synthesized design before opening the implemented design, and **Don't Save**, if prompted.

TIP: *This prompt might have been turned off if you elected to never show this dialog again. To restore this prompt, go to* ***Tools > Options > Window Behavior*** *and check the* ***Show dialog before switching to a different design*** *checkbox.*

The results from the implemented design are imported into the Vivado IDE as shown in Figure 22. The Device window shows the placement of all logic instances onto the available device resources.

Notice the status of the Implemented Design in the upper-right corner of the Vivado IDE is **Implementation Complete, Failed Timing!** In Lab #2 you will explore the timing failures.

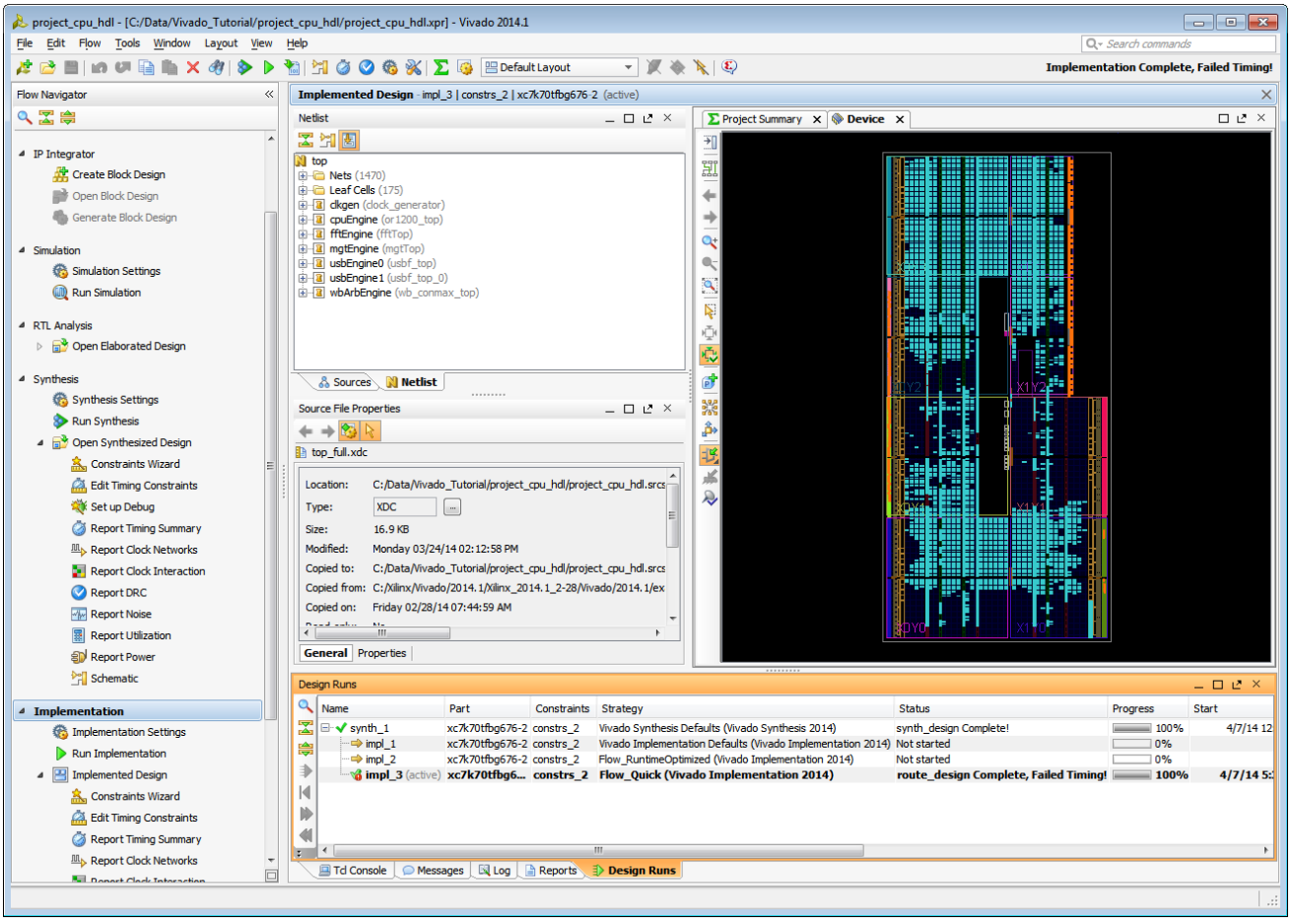


Figure 22: Implemented Design

4. Click the **Messages** window tab at the bottom of the Vivado IDE, to display the Messages window.

   Note: The Messages window collects all information, warning, and error messages from the various tools of the Vivado Design Suite.

5. Click the **Collapse All** toolbar button to display the various message categories .

6. Expand the **Implementation** messages.

7. Expand **Route_Design** folder to see the messages generated at this step.

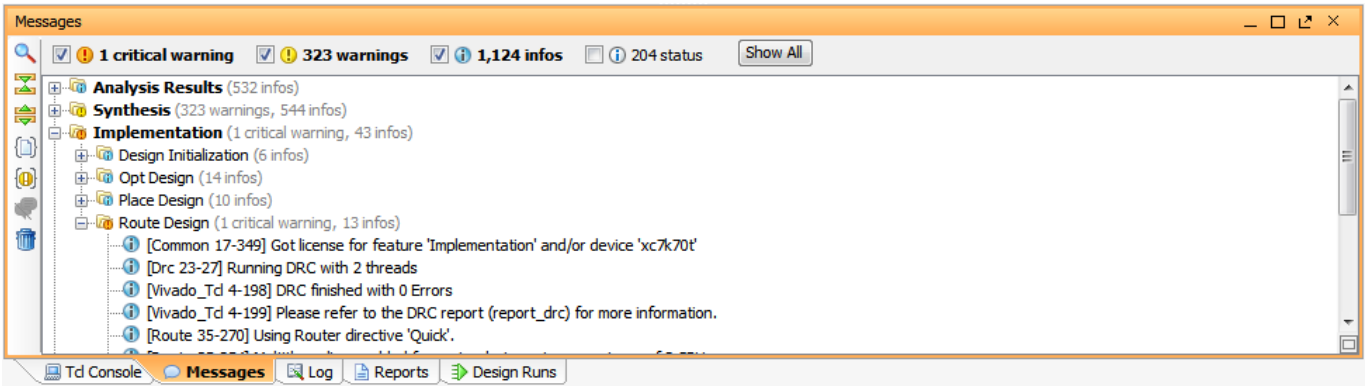


Figure 23: Implementation Messages

8. Click the **Reports** window tab to display the various reports generated by different tools in the Vivado Design Suite.

   If the Reports window tab is not displayed, select **Window > Reports** to display the Reports.

9. **Double-click** any of the **reports** listed in the Reports window to open those reports in the Vivado IDE text editor.

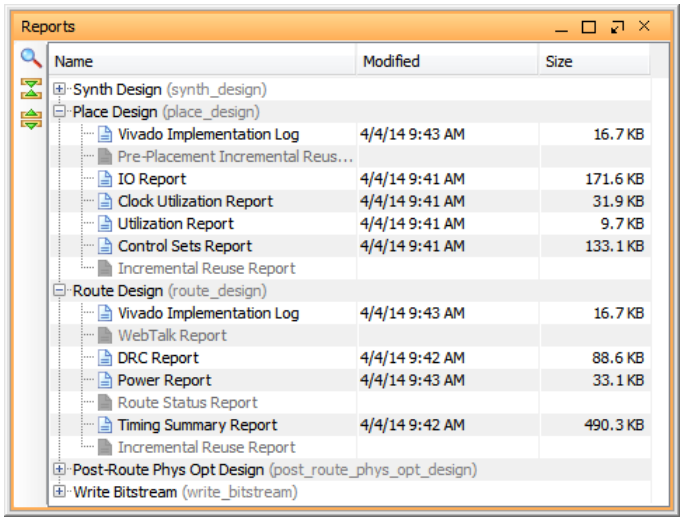


Figure 24: Implementation Reports

Conclusion

In this lab, you have opened the CPU HDL project, and examined various aspects of the design using tools available within the Vivado IDE. You have examined the hierarchy of the design, analyzed the timing slack histogram, and the timing report prior to implementation, and synthesized and implemented the design. Finally, you examined the messages and reports generated by the Vivado implementation tools.

This concludes Lab #1. You can proceed at this time to Lab #2, or use the following steps to exit the tool:

1. Click **File > Close Project** to close the CPU HDL design.
2. Click **File > Exit** to exit the Vivado Design Suite.

Lab 2: Timing Closure Techniques

This lab continues from the end of Lab #1 in this tutorial. You must complete Lab #1 prior to beginning Lab #2. If you closed the tool, or closed the tutorial project at the end of Lab #1, you will need to open them again.

1. Start by loading the Vivado Integrated Design Environment (IDE).
   - Launch Vivado IDE from the icon on the Windows desktop, or
   - Type **vivado** from a command terminal.

   The Vivado IDE opens.
2. From the **Recent Projects** list of the Getting Started page, **click** once to select and **open** the **`project_cpu_hdl`** project you created in Lab #1.

Figure 25: Open Recent Projects

3. After the project has finished opening, from the Flow Navigator click **Open Implemented Design**.

Step 1: Analyzing Post-Implementation Timing

With the design implemented onto the target device, you can begin to explore the design and constraints to resolve any challenges with meeting the timing requirements of the design. The Vivado Design Suite offers an array of tools and techniques to help you achieve timing closure.

The Report Timing Summary command creates a set of timing reports intended to provide sign-off level timing analysis for the post-implemented design. You can analyze timing results from the implementation process to drive any floorplanning that might be required to meet timing requirements.

IMPORTANT: *Only the Vivado timing analysis Report Timing Summary command provides a sign-off level timing report.*

The Vivado tool runs timing analysis after implementation, and generates a report file you can open in the Text Editor for review. However, to view the report interactively in the Vivado IDE, and cross-select and analyze the design, you must regenerate the report.

Viewing Timing Paths

1. From the Flow Navigator, under Implemented Design, select **Report Timing Summary** and click **OK** to run Vivado timing analysis with the default settings.

   The Timing Summary window opens, as shown in Figure 26. In the figure below, the Design Timing Summary is selected. You will recall that in Lab 1 you ran the Report Timing Summary prior to placement, so the interconnect delays were only estimated. This time the interconnect delays are based on the actual route.

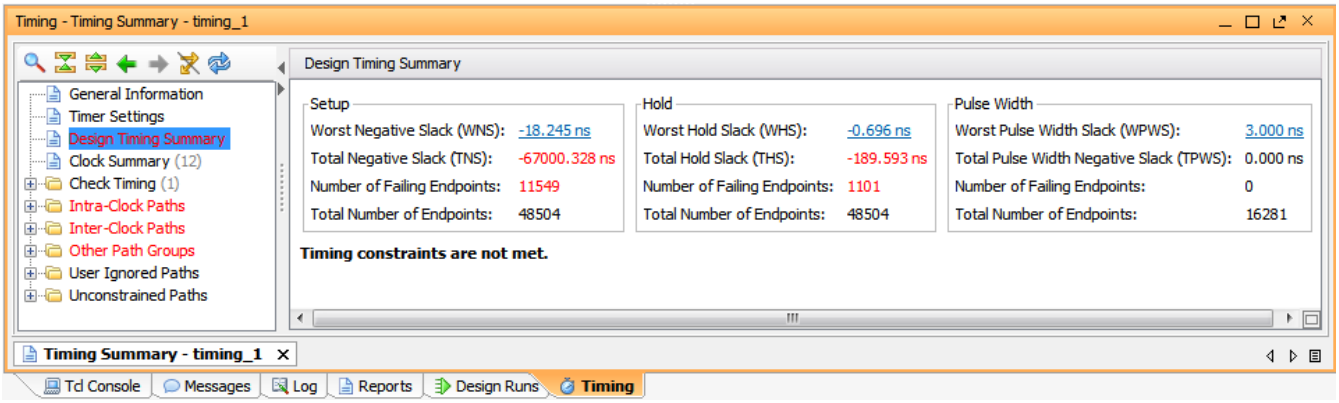


Figure 26: Report Timing Summary

   The Design Timing Summary reports timing slack values, such as Worst Negative Slack (WNS), showing whether the design satisfied the timing constraints. Any negative slack values, or a number of failing endpoints, are an indication of timing problems.

   The Flow_Quick strategy employed by the implementation run, `impl_3`, is intended to perform a quick placement and routing of the design to assess resource utilization and see if the design will fit onto the target part. This strategy ignores timing constraints, and makes no effort to meet timing, as is seen clearly in the results In this design, the Design Timing Summary reports many paths which fail timing.

2. **Click** the hyperlinked value for the **Worst Negative Slack** (WNS) to select and display the timing path with the worst slack as shown in Figure 27.

   Notice that the worst timing paths in the report connect to the `usbEngine1` module. This information will guide your floorplanning strategy later in this tutorial.

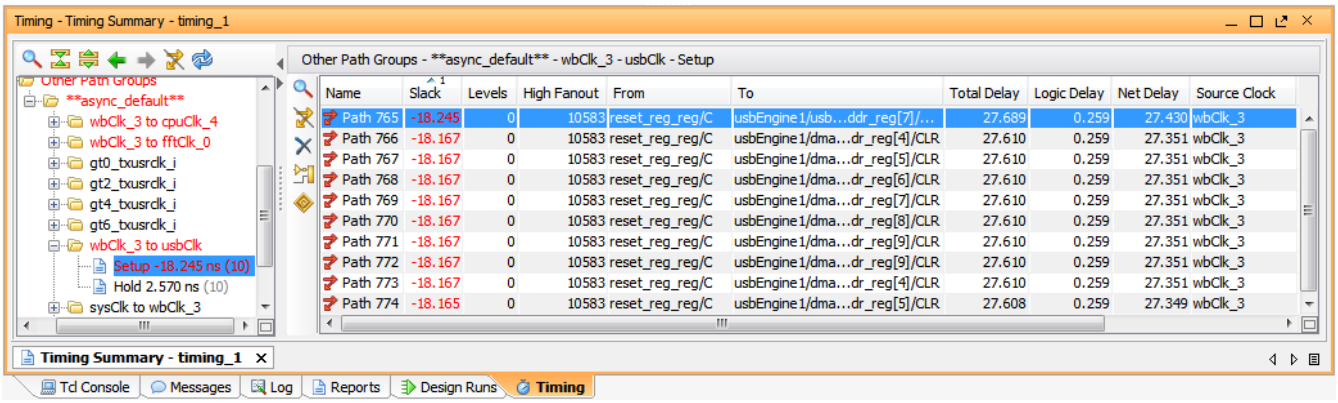


Figure 27: Worst Negative Slack

3. Right-click and select **Mark** in the popup menu.

   In an implemented design, selecting the timing path in the report also selects the path in the Device window. The Device window can make it easier to see timing challenges due to placement and routing, and help you visualize how floorplanning could improve timing.

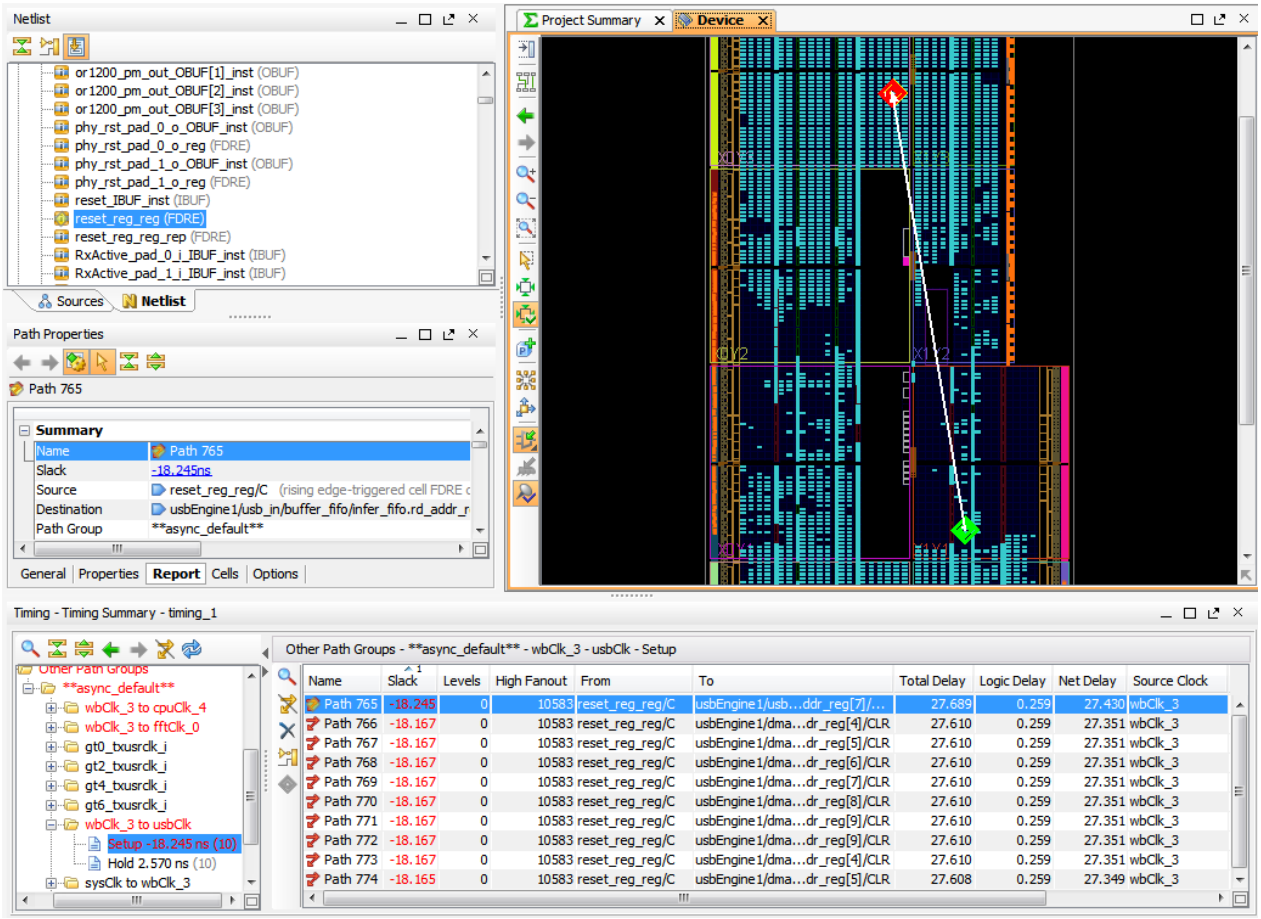


Figure 28: Marking the Timing Path

The selected path is marked with a green diamond showing the timing path *start* point, and a red diamond showing the timing path *end* point. Yellow diamonds mark any *through* points in the path.

Viewing the Routed Path

1. In the Device window, select the **Routing Resources** toolbar button to enable the display of routing resources.

   When enabled, the Routing Resources command displays a graphical representation of the device components, and the FPGA interconnections between the components. This provides a detailed view of the available routing resources on the device, and shows the specific resources used by a selected path, as seen in Figure 29.

2. In the Device window, select the **Auto-fit Selection** toolbar button to enable automatic zoom for each newly selected object.

   Select various paths in the Timing Summary window, and notice the detailed routing displayed for each timing path in the Device window. This routing resources view can provide you a better understanding of the interconnect portion of the path delay, and help you resolve timing problems. Refer to the *Vivado Design Suite Tutorial: Implementation (UG986)* for more information.

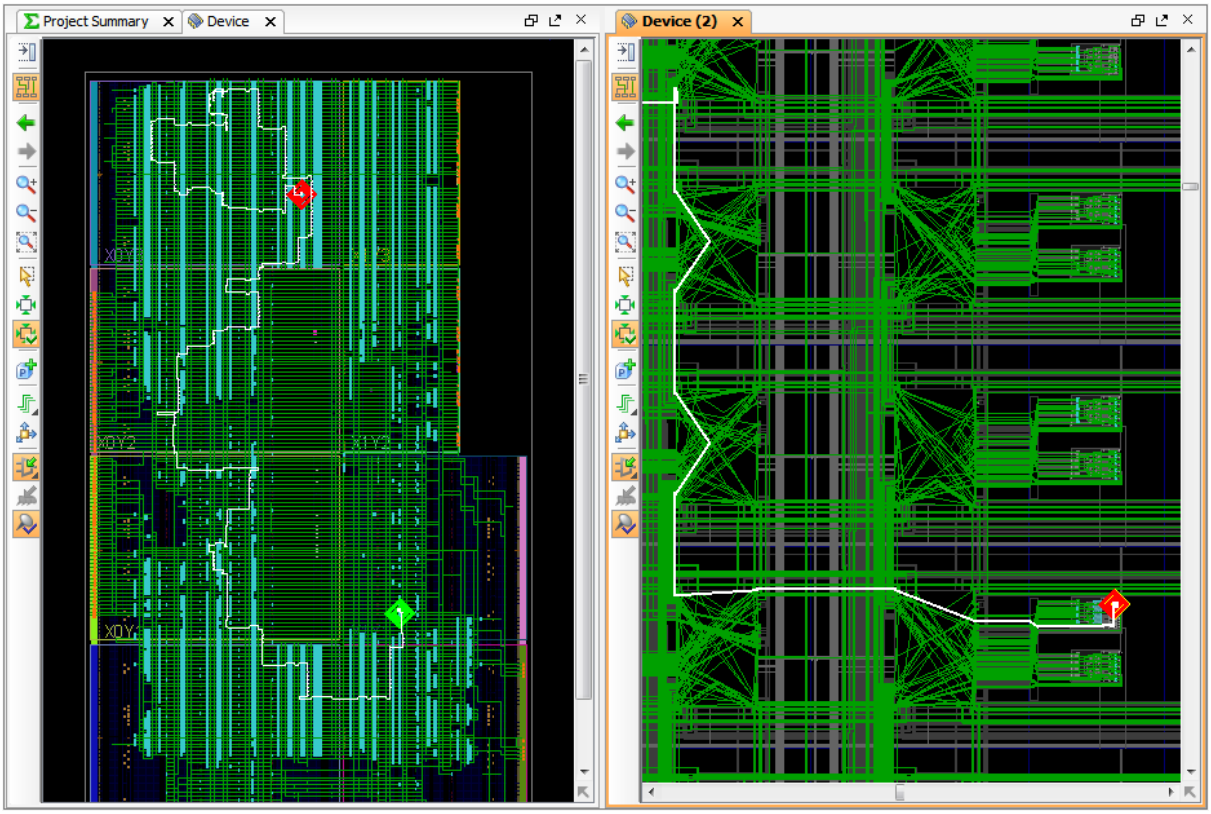


Figure 29: Routing Resources in the Device window

3. **Zoom** closely into a timing path displayed in the Device window to see the details of the routing resources.

   When looking at the routed path, it may seem to take an indirect route between connection points. This is because the Vivado router considers many different aspects of a design. If the design is meeting timing, other considerations such as routing density and power analysis, or timing requirements on other paths may affect the route solution.

4. Right-click and, from the popup menu, select **Unmark**, or select the **Unmark All** button from the main toolbar to clear the marks.

5. Click the **Unselect All** button on the main toolbar menu, or press **F12,** to unselect all currently selected objects.

6. In the Device window toolbar menu, select the **Routing Resources** toolbar button again to disable the detailed routing view.

   Notice the placed cells are now displayed in the Device window again, without the added details of the routing resources.

Viewing Path Properties

When you select paths in the Timing Summary window, the Path Properties window opens to show a summary and details of the timing path.

1. In the Timing Summary window, **reselect** the first **path**.
2. Select the Path Properties window, and **Maximize**, or **Float**, the window for the selected path, and examine the information provided as shown in Figure 30.
3. **Restore** or **dock** the Path Properties window to its original size and location.

TIP: *You can maximize or restore a window by double-clicking in the window banner.*

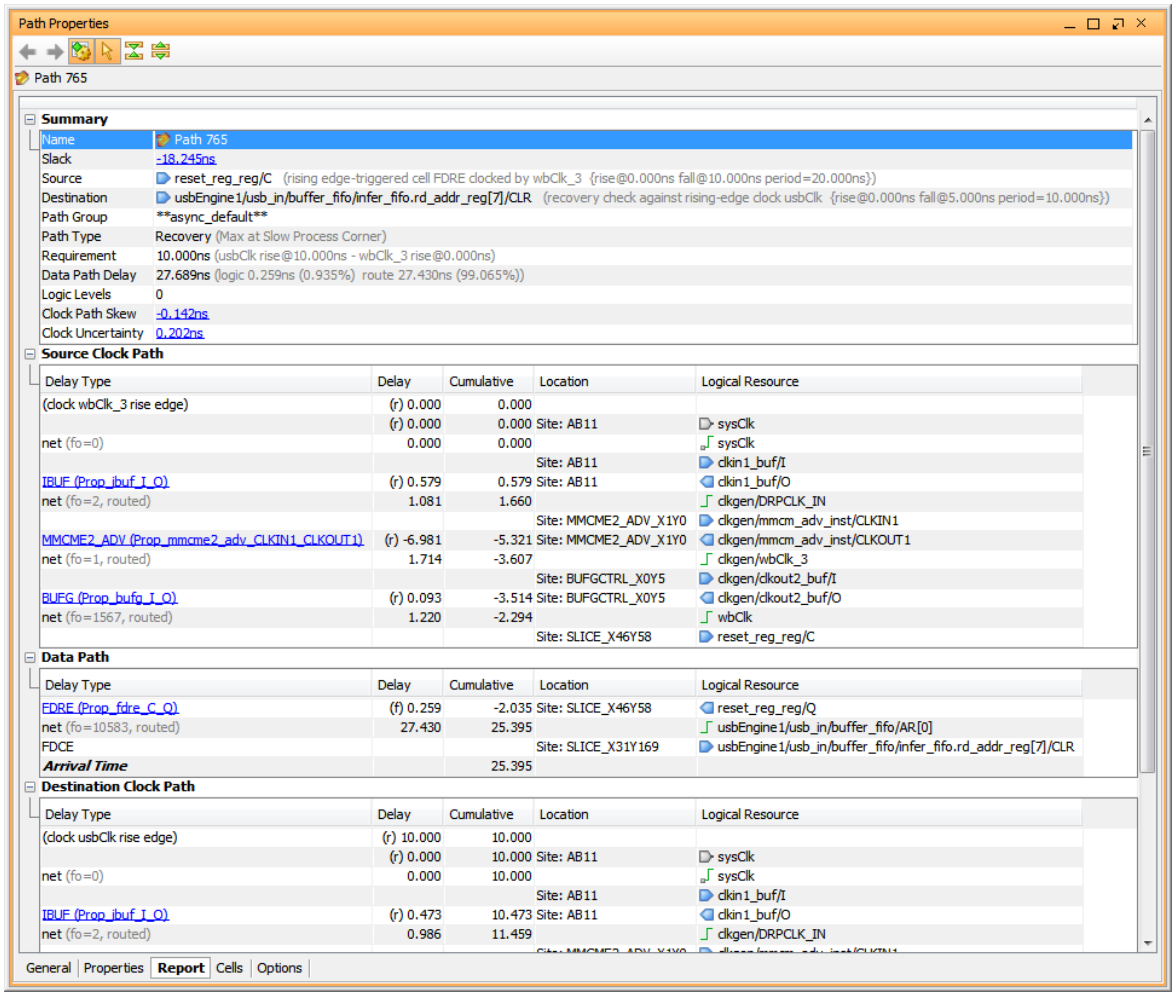


Figure 30: Path Properties window

Step 2: Exploring the Hierarchy of the Timing Paths

The Vivado IDE offers many tools to help you locate logic associated with the reported timing paths. In this step of the tutorial, you will use the Schematic and Hierarchy windows to visualize the logic associated with the timing paths with the worst delay.

1. On the left side of the Timing Summary window, expand the **Intra-Clock Paths**.
2. Select and expand the **usbClk** path by double-clicking the name, and select **Setup**.
3. On the right side of the Timing Summary window, **select** the first path listed.
4. Right click, and select **Schematic** to create a schematic for the selected path.

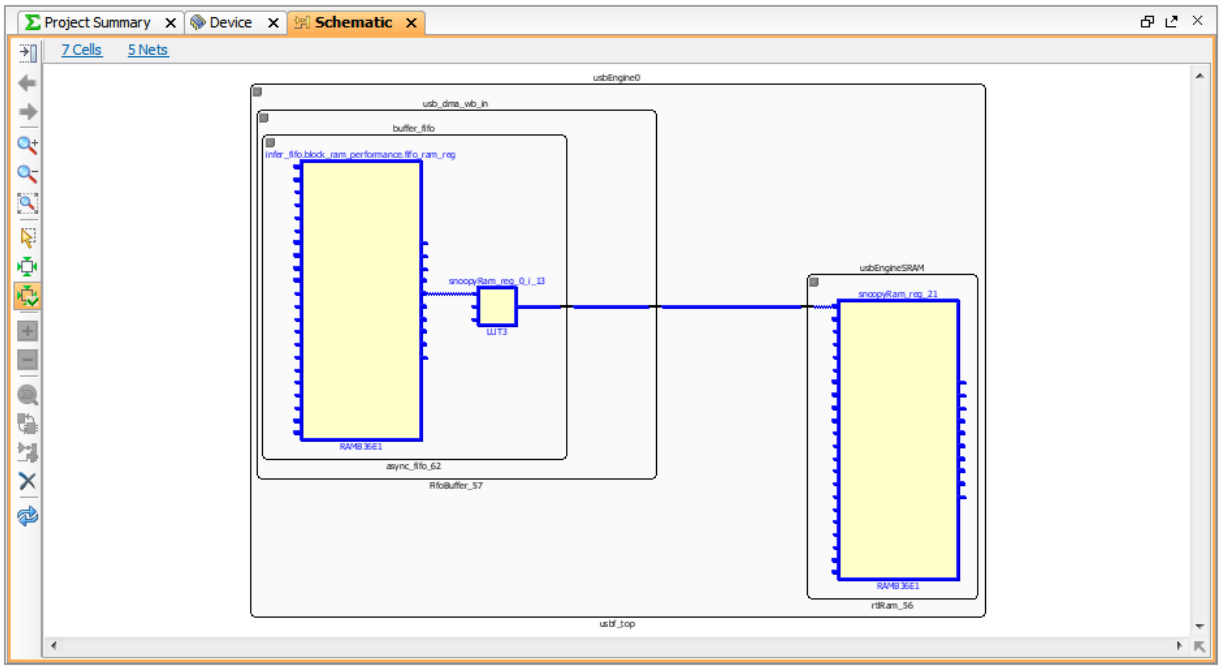


Figure 31: Timing Path Schematic

When you select a timing path in the Timing Summary window, the Vivado tool also selects the cells associated with the start points, end points, and through points on the timing paths. The Schematic window shows all of the logic cells on the selected timing paths.

5. In the Netlist window, click the **Collapse All** button.

   This simplifies the information displayed in the Netlist window, so that the items selected in the next task are more visible, and easier to see.

TIP: *Sometimes, because of the volume of information presented in the Netlist window, it is beneficial to collapse the tree view, so you can more clearly see newly selected objects.*

---

6. In the Schematic window, right-click and select **Select Leaf Cell Parents**.

   The Select Leaf Cell Parents command selects the hierarchical parent of any currently selected leaf cells, or modules, working up through the hierarchy until it gets to the top-level of the design. The parent modules are also selected in the Netlist window.

7. In the Netlist window, right-click and select the **Show Hierarchy** command from the popup menu, or press **F6**.

   The selected logic, which is associated with the timing paths with the worst delays, is all contained in the `usbEngine0` module. There is a duplicate module, `usbEngine1`, with identical logic, which will have many of the same problems. You will explore these two `usbEngine` blocks further for floorplanning.

8. **Close** the **Hierarchy and Schematic** windows, by clicking the X button on the window tabs.

9. **Select** the first **path** in the list in the Timing Summary window.
10. **Right-click** and select **Schematic** from the popup menu.

    The timing path logic is shown in the Schematic window as seen in Figure 31. The nested blocks shown in the Schematic window, represent the hierarchical structure of the design. With this view, it is easy to identify the various logic modules associated with the critical path logic. For the selected timing path, you can see the logic is contained within the `usbEngine0` block.
11. Click the **Unselect All** button on the main toolbar menu, or press **F12**.
12. Select the **RAMB36E1** in the `buffer_fifo` block of the opened Schematic window.
13. Right-click and select **Expand/Collapse > Collapse Outside** from the popup menu.

    This will collapse all of the modules outside of the selected cell, so that the schematic is only displaying the hierarchy of the selected block RAM.
14. Double-click the outside of the **ADDRARDADDR[15:0]** pin, in the top left of the block RAM, and notice that the logic connected to it is expanded outward.

TIP: *Double-clicking outside the pin is the same as selecting the pin and using the* ***Expand Cone > To Leaf Cells*** *command from the popup menu.*

15. Select the **infer_fifo.wr_addr_reg[0]** cell, displayed at the top of the newly added logic.
16. Right-click and select **Expand Cone > To Leaf Cells** from the popup menu, or double click the specified cell.

    This expands the logic connected to the selected cell outward to the next connected leaf cell, as shown in Figure 32. The logic associated with this expansion includes a reset net, a global clock buffer, and logic in the `usbEngine1` bock. You can see the block ram is relatively well insulated from the other logic.

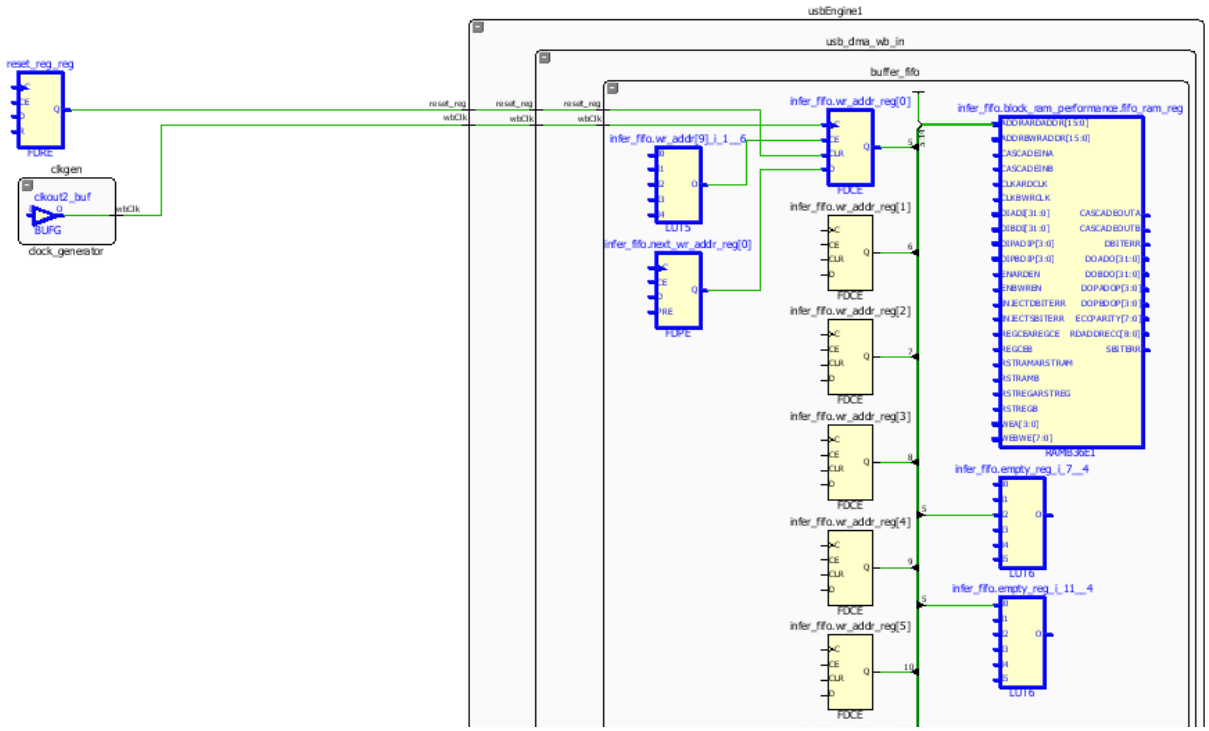


Figure 32: Interactively Expanding Logic

17. In the Schematic window, select the **Previous schematic** button .

    You can use the **Previous schematic** and **Next schematic** toolbar buttons to switch between prior views displayed in the Schematic window.

18. In the Schematic window tab, **click** the **X** button to **close** the **Schematic** window.

    The input to this block RAM is contained in the `usbEngine` block. You do not have to consider other hierarchy for floorplanning.

19. Click the **Unselect All** button, or press **F12**.

20. In the Netlist window, click the **Collapse All** button.

Step 3: Highlighting Module Placement and Connectivity

At this point, you have determined that the worst timing delays are found in the `usbEngine` blocks, and examined the logic in the schematic. You will now evaluate the current module placement to understand how the logic was implemented without floorplanning, to see how placement can be improved. This will lead you to a placement strategy you can implement through floorplanning.

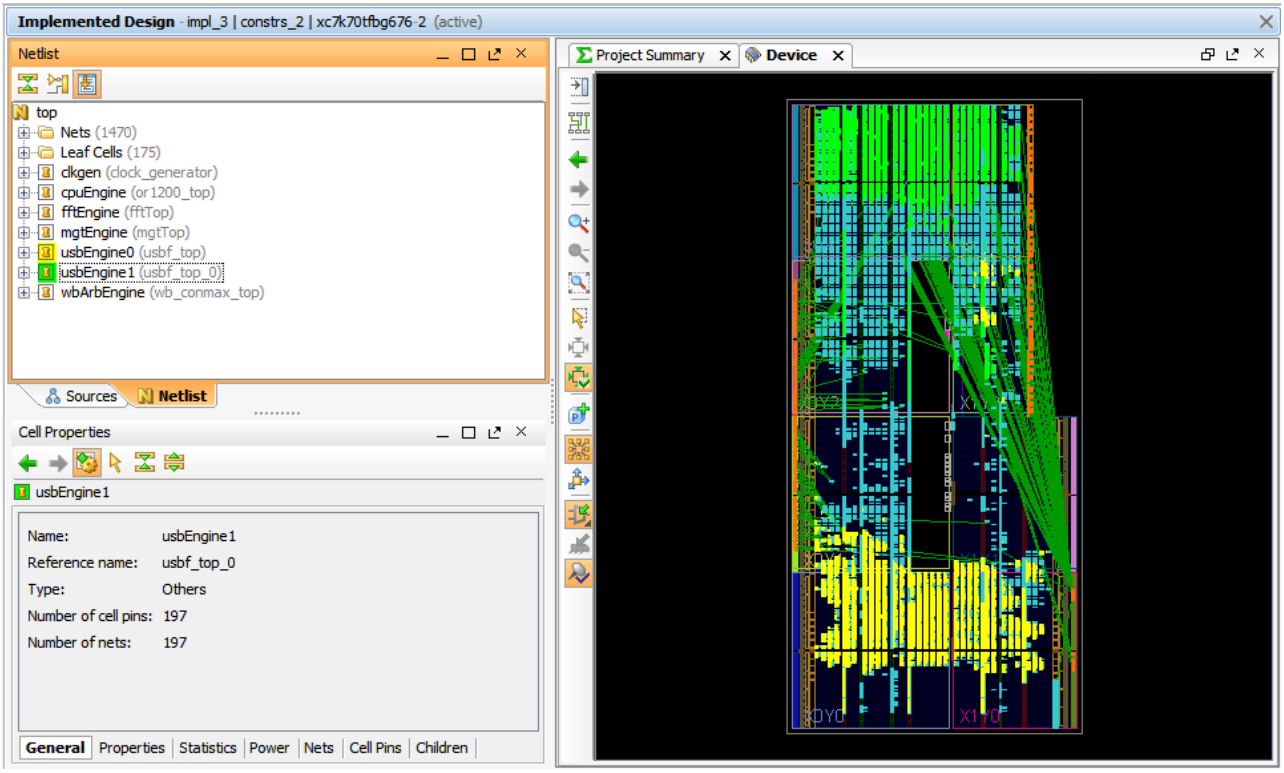


Figure 33: Highlight Module Placement

1. In the Netlist window, select **usbEngine0** and **usbEngine1**.
2. Right-click and select **Highlight Leaf Cells > Cycle Colors** from the popup menu.

   The leaf cells in each module are highlighted in a different color as shown in Figure 33. Notice the placement of the cells within each module, with `usbEngine0` highlighted in one color, and `usbEngine1` highlighted in a different color. Notice the logic is spread out across the device.
3. In the Device window, click the **Device View Options** button .
4. Uncheck the box next to **Cells** to disable the display of the placed cells in the Device window.

   Various elements of the Device window can be hidden, to make it easier to see important details of the design or device.

Zoom into the Device window for a closer look, and pan around the window to examine the placement. With logic spread out across the device, the usbEngine blocks could benefit from floorplanning to improve timing.

5. Enable Cells in the **Device View Options** menu.
6. Click the **Device View Options** button to hide the menu.
7. From the main toolbar, click the **Unhighlight All** button.

Visualizing the Connectivity

You can alleviate routing congestion and timing issues by floorplanning to prevent logic from being placed into critical or congested areas during implementation. Evaluating the design connectivity will help you determine which modules are suitable for floorplanning. For example, logic modules that connect to logic throughout the device may not be suitable for floorplanning, while tightly grouped and self-contained modules are suitable.

1. In the Device window, if it is not enabled, select the **Show I/O Nets** toolbar button.

   The wires in the Device window in Figure 34 show the connectivity between the I/O ports and placed instances on the device.

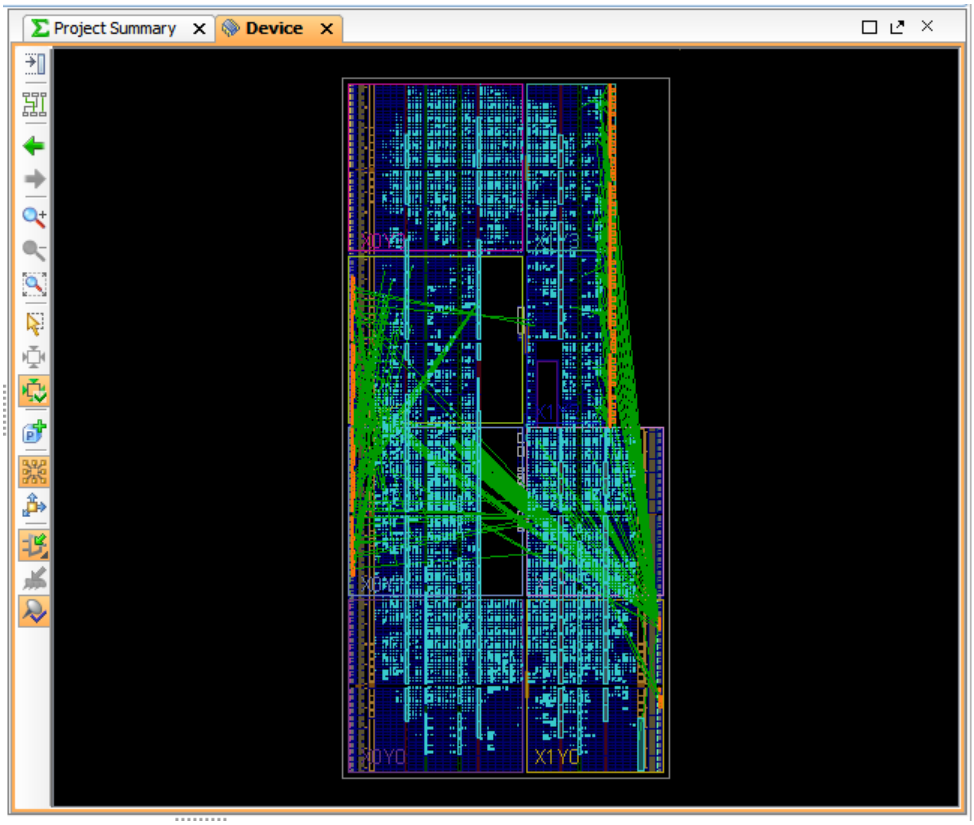


Figure 34: Show I/O Connectivity

2. Click to select one of the I/O Nets.
3. Inspect the net name in the Netlist window.
4. In the Netlist window, click the **Collapse All** button .
5. From the Netlist window, select **usbEngine0** and **usbEngine1**.
6. Right-click and select the **Show Connectivity** command from the popup menu, or press **Ctrl-T**.

   The interface nets that connect `usbEngine0` and `usbEngine1` to the rest of the design are highlighted, as shown in the following figure. You can see that most of the connections go to the I/O Blocks on the left side of the device. This will help guide your placement during floorplanning.

   You will floorplan the `usbEngine` blocks to keep the I/O logic near the I/O blocks, not specifically to improve timing. However, this could improve timing if the design was failing timing in some way.

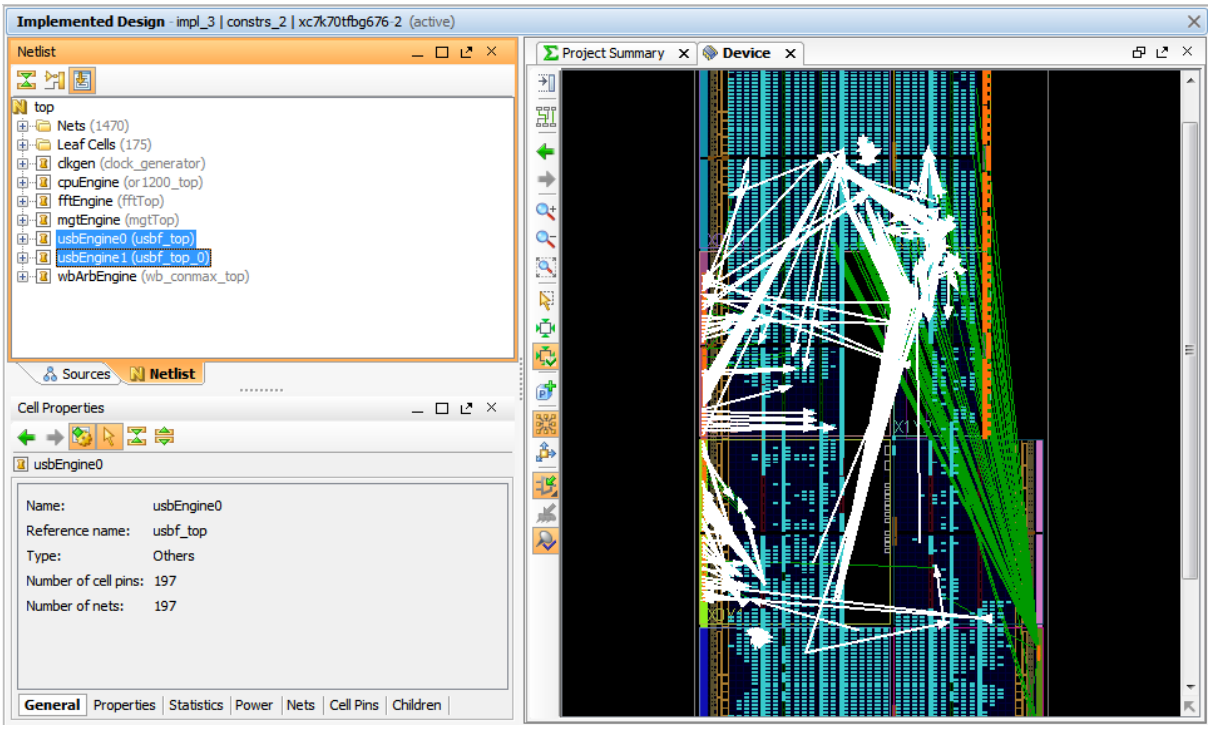


Figure 35: Show usbEngine Connectivity

   You can use the Show Connectivity command to highlight or select a cone of logic from any source net or logic object.

7. Click the **Unselect All** button or press **F12**.

Step 4: Floorplanning the Design

With an understanding of the design connectivity, and the problems of the prior placement, you can now begin floorplanning the design. The process of floorplanning begins by grouping some or all of the logic in the design into physical blocks, called Pblocks, and constraining that logic with common constraints. Grouping associated logic together into Pblocks helps you analyze the data flow of the design, limits interconnect lengths, prevents logic migration, and reduces timing delays.

Defining a Pblock results in a `create_pblock` Tcl command that is written to the target constraint file. The constraint defines the assigned logic, specified site ranges, and defined properties of the Pblock.

Automatically Creating Pblocks

1. Select **Tools > Floorplanning > Auto-create Pblocks**.

   This opens the Auto-create Pblocks dialog box.

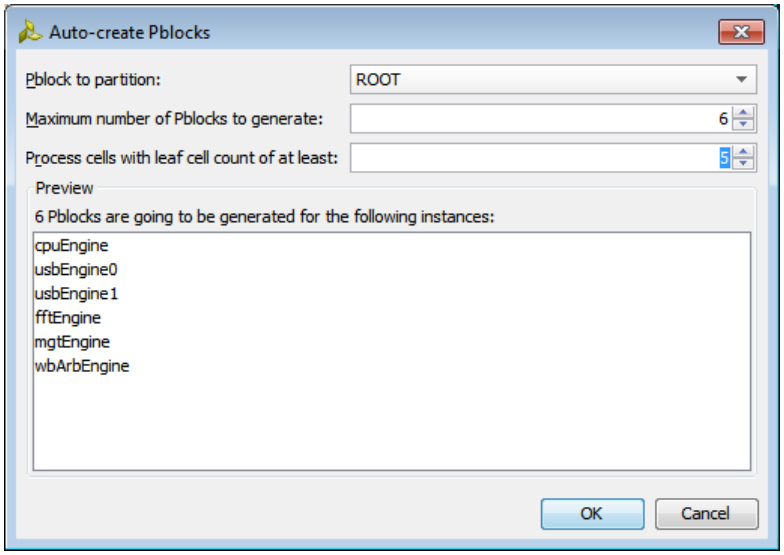


Figure 36: Auto-Create Pblocks

   In the dialog box, review the options to define the maximum number of Pblocks to automatically create, and to limit the size of the modules to create Pblocks from.

2. Change the **Maximum number** of Pblocks to generate to **6**.

   If the design has more modules than the maximum number of Pblocks, the Vivado tool creates Pblocks starting with the largest modules first.

3. **Click OK** to create Pblocks for the listed modules.

4. **Click OK** to confirm the creation of six Pblocks in the Auto-create Pblocks dialog box.

5. **Select** the **Floorplanning layout** from the View Layout drop down menu to display the Physical Constraints window.

   The newly created Pblocks, as well as the top-level ROOT, are listed in the Physical Constraints window, as shown in Figure 37. In the Netlist view, the icon next to the six modules that have been defined as Pblocks has changed from to , to illustrate this change.

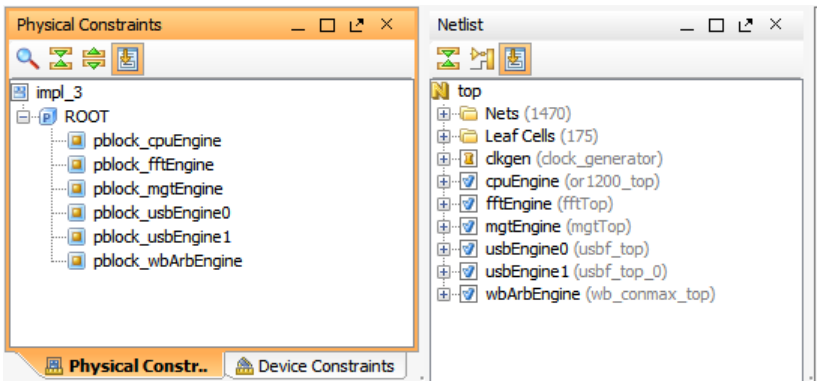


Figure 37: Physical Constraints window

Placing Pblocks

1. Select **Tools > Floorplanning > Place Pblocks**.

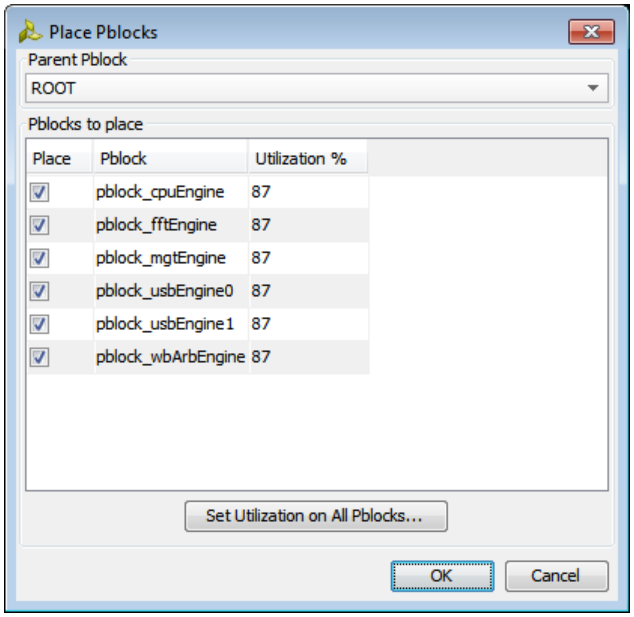


Figure 38: Place Pblocks

The Place Pblocks dialog box contains options to define which Pblocks to place, and what the target SLICE utilization for the Pblocks should be. The size of the Pblock created by the Vivado IDE depends on the Utilization % specified in the Place Pblocks dialog box.

TIP: *The Utilization % is calculated using SLICE logic only. This provides a useful estimate for determining the size of a Pblock. However, Pblocks may need to be resized prior to implementation to insure sufficient resources are available for all logic types assigned to the Pblock.*

2. Click **OK** to accept the **default values**, and place all the Pblocks.

   The specific placement of the Pblocks is preliminary, and may be different each time the Place Pblocks command is run. You can arrange the Pblocks in the Device window to untangle the connectivity, as shown in Figure 39, by selecting and moving them as described in Moving Pblocks.

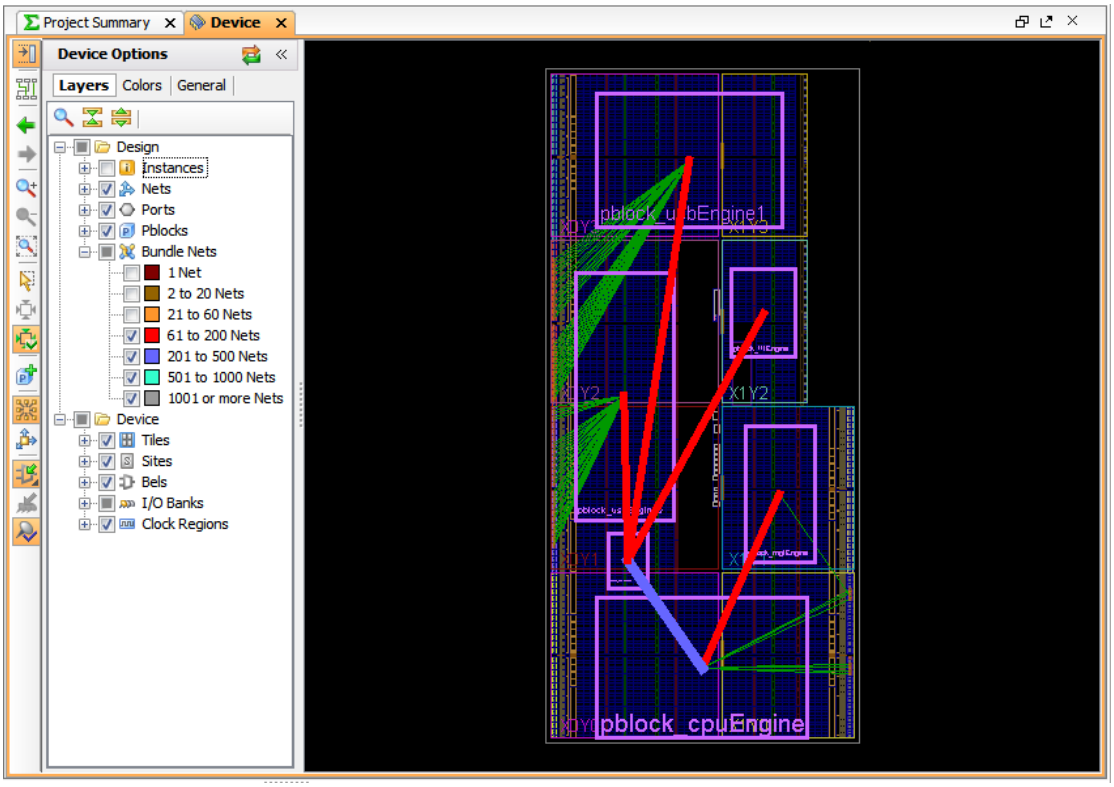


Figure 39: Pblock Data Flow with Bundled Nets

Viewing Pblock Connections

1. Select the **Show/Hide I/O Nets** toolbar button to enable the display of connections between the Pblocks and the I/O ports.
2. Open the **Device View Options** menu from the toolbar, and enable the display of Bundle Nets under the Design heading.

   Bundle nets and I/O connections help you visualize the data flow of the design, as seen in Figure 39. The preliminary floorplan provides early indications of data flow through the design, and highlights where potential routing congestion could occur. Bundles are color coded to show the number of nets connecting two Pblocks.

TIP: *To see or change the current color definitions, use the* ***Tools > Options*** *command to open the Vivado Options dialog box. Select the Colors button on the left side of the dialog box, and select the Bundle Nets category on the right. You can also view the colors defined under the Device View Options menu in the Device window, but you cannot change the colors from that location.*

---

3. In the Device window, **select** a bundle net.

   Examine the list of nets in the Bundle Net Properties window as shown in Figure 40. These are the nets contained within the selected bundle, connecting two Pblocks.

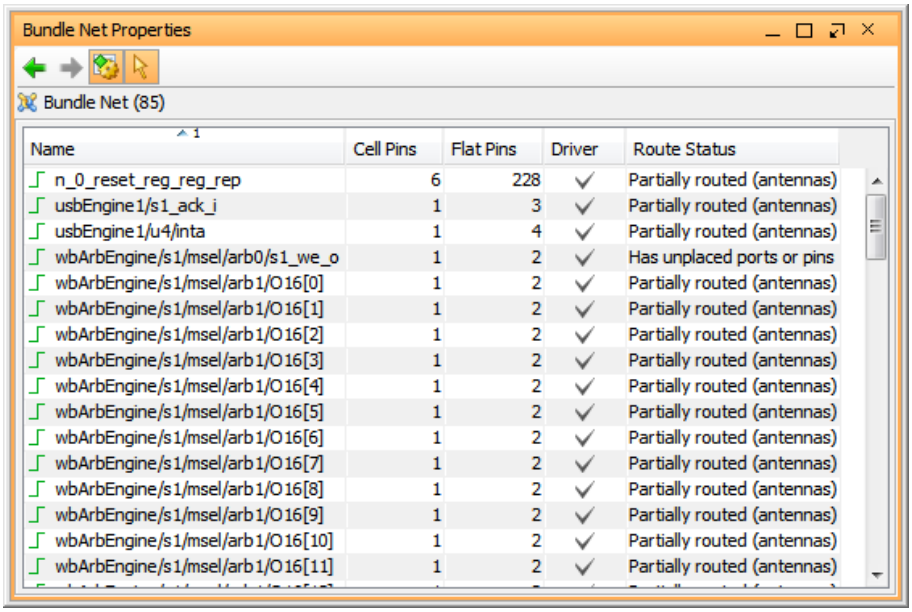


Figure 40: Bundle Net Properties

Finding a Global Clock

In this step, you will use the Find command to locate and select the USB Global Clock (`usbClk`) to highlight the clocked Pblocks.

Effective floorplanning is often dependent on proper placement of the synchronous elements from different clock domains. You can highlight clock domains to visualize connectivity and ensure that Pblocks are properly placed to effectively utilize clock resources on the device. To learn more about clock resources and clock management, refer to the Xilinx *7 Series FPGAs Clocking Resources User Guide (UG472)*.

1. Select **Edit > Find**, or type Control-F.

   This opens the Find dialog box to search through the design for specific types of objects.

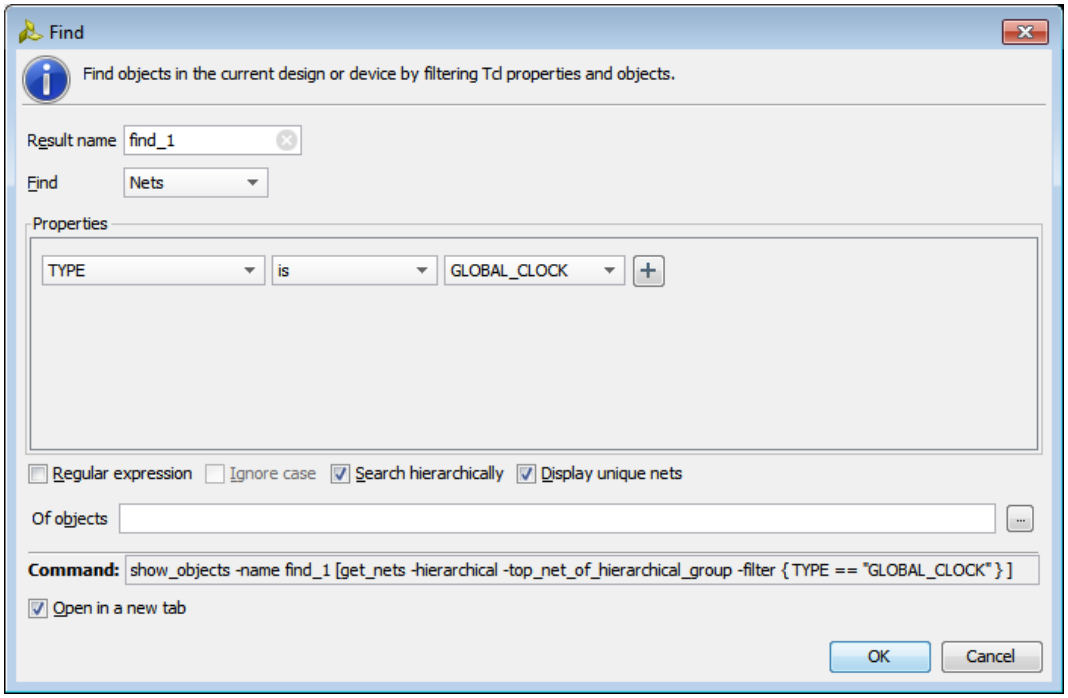


Figure 41: Find Global Clocks

2. In the Find dialog box, set the following options:
   - Find: **Nets**
   - Properties: **Type**, **is**, **Global Clock**
3. Click **OK**.

   The Find results window opens at the bottom of the Vivado IDE as shown in Figure 42.
4. In the Find Results window list, select usbClk.

   The selected net is highlighted in the Device window, connecting the two usbEngine Pblocks. This clock is limited to these two blocks.

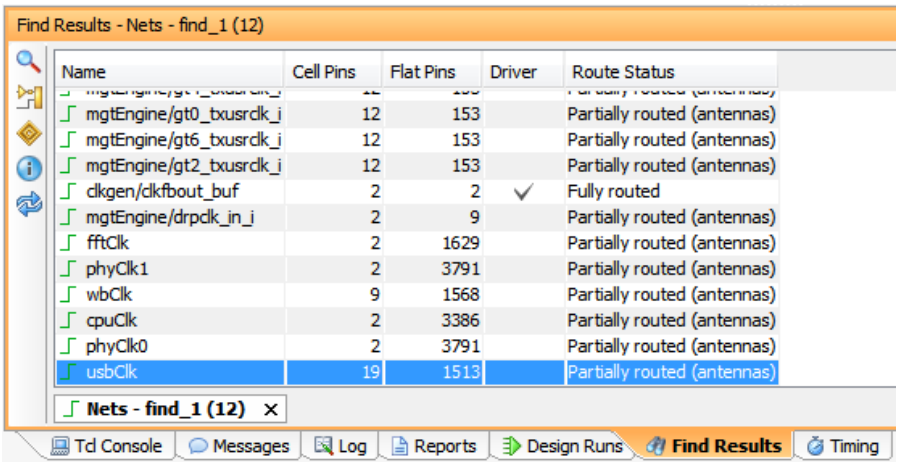
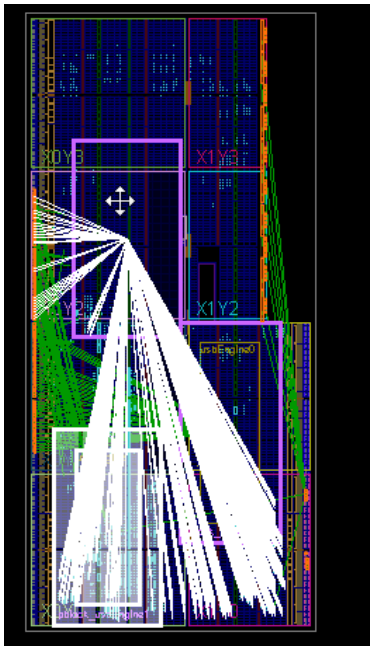


Figure 42: Find Global Clock

Moving Pblocks

With the bundle nets and the I/O connections displayed, you may need to rearrange the preliminary Pblock placement to better visualize the data flow through the design as seen in Figure 39. You can move the Pblocks in the Device window by clicking on a Pblock and dragging it to the desired location.



1. **Select** a **Pblock** in the Physical Constraints window, or select it directly in the Device window.
2. **Move** the Pblock by dragging it to a new location on the Device.

   The placement you choose should seem appropriate based on the displayed connections between the Pblocks and to the I/O ports. To reduce delay, you should place the Pblock near the I/O ports it is connected to.

   When you place the Pblock in its new location, the Move dialog box prompts you to select which resource types in the new location should be enabled for use by the Pblock.
3. Select **OK** to include all resource types.

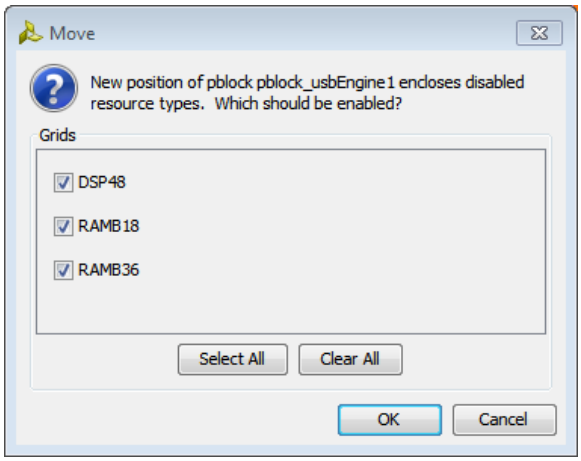


Figure 43: Move Pblock

When you move a Pblock, with logic elements that have been previously placed, the Vivado tool will open the Choose LOC Mode dialog box to prompt you to decide how to handle existing cell placement, or LOC constraints, as shown in Figure 44.

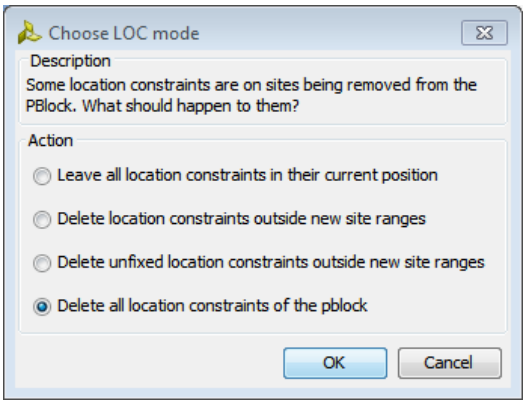


Figure 44: Choose LOC Mode

The choices are: [3]

- Leave cells where they are currently placed.
- Unplace cells that fall outside the new Pblock area.
- Unplace only unfixed cells that fall outside the new Pblock area.
- Unplace all cells, fixed and unfixed, in the Pblock.

[3] The choices offered in the Choose LOC Mode dialog box depends on the contents of the Pblock being moved, and whether there are fixed and unfixed cells inside the block.

IMPORTANT: *Both fixed and unfixed cells are placed.*

Fixed and Unfixed are terms that apply to placed cells, and describe the way the Vivado tools views placed cells in the design.

- *Fixed cells have been placed by you, or another designer, or have been imported from a constraints file. These placed cells are therefore considered fixed by the Vivado Design Suite, and will not be moved unless directed to do so.*
- *Unfixed cells have been placed by the Vivado tools in implementation, during the place_design command, or one of optimization commands. These placed cells are considered unfixed, or loosely placed, by the Vivado Design Suite, and can be moved as needed in design iterations.*

4. Select **Delete all location constraints** of the pblock and click **OK**.

   This will unplace all cells assigned to the Pblock, clearing the LOC constraint from the cells, allowing them to be relocated to the new area covered by the Pblock.

Clearing Placement LOCs

An alternative to clearing or keeping the current placement for each Pblock as you move it, is to clear the placement (LOCs) for the whole design. This lets you move Pblocks without having to respond to the Choose Loc Mode dialog box each time.

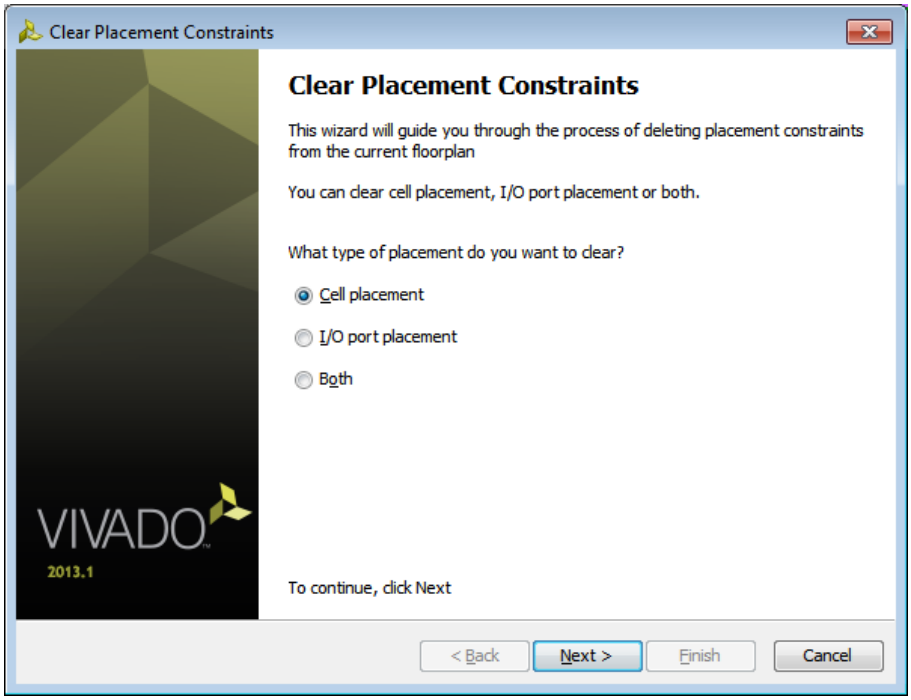


Figure 45: Clear Placement Constraints

1. Use the **Tools > Floorplanning > Clear Placement** command to clear the current placement for the whole design.

   The Clear Placement Constraints dialog box, as shown in Figure 45, lets you clear placement for either the cells or the I/O ports in the design. Or clear placement for both cells and ports.

2. Select **Cell placement** and click **Next**.

   The Cell Types to Unplace dialog box opens as shown in Figure 46.

3. **Click Next** to accept the default, and unplace everything but the I/O and the clock logic.

   The Clear Placement Summary dialog box shows what instances will be unplaced.

4. **Click Finish** to clear the specified placement LOCs from the design.

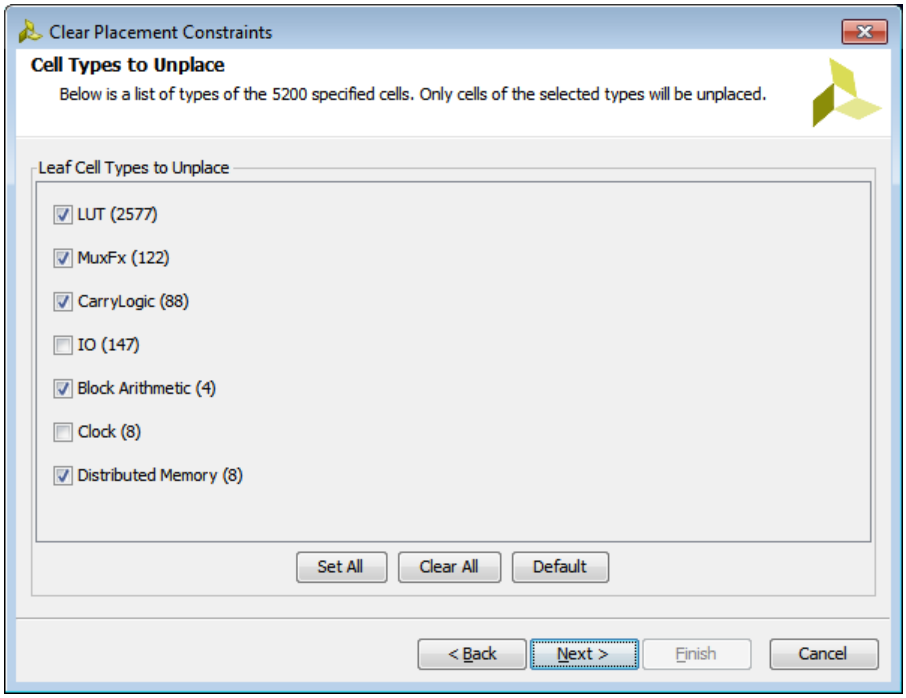


Figure 46: Clear Placement Types

You can also clear placement using Tcl commands.

5. **Enter** the following **command** in the Tcl console:

```
unplace_cell [get_cells -hier * \
-filter {((is_primitive==true && primitive_level!="INTERNAL") && \
(loc!="") && (primitive_group== FLOP_LATCH))}]
```

Resizing Pblocks

At this point, to better focus on the two `usbEngine` blocks, select and clear the other Pblocks in the Physical Constraints window.

1. In the **Physical Constraints** window, **select** `pblock_cpuEngine`, `pblock_fftEngine`, `pblock_mgtEngine`, and `pblock_wbArbEngine`.
2. **Right click** and select the **Delete** command.

   This deletes the selected Pblocks, leaving only the `usbEngine0` and `usbEngine1` Pblocks. The other Pblocks helped you understand the data flow through the different modules of the design, but are not needed for the final floorplan.

   With the `usbEngine` blocks placed approximately where you would like them, you must now resize the Pblocks to include the required resources.
3. **Select** a **Pblock**, and click the **Statistics** tab in the Pblock Properties window.

   The Statistics tab of the Pblock Properties window shows the device resources available in the area covered by the Pblock; shows the amount of resources required by the logic assigned to the Pblock; and shows the current utilization %.

   The red values, as shown in Figure 47 indicate insufficient resources available in the Pblock area to meet the requirements of the design logic assigned to the Pblock. In this case, the Pblock will need to be resized to increase its available resources.

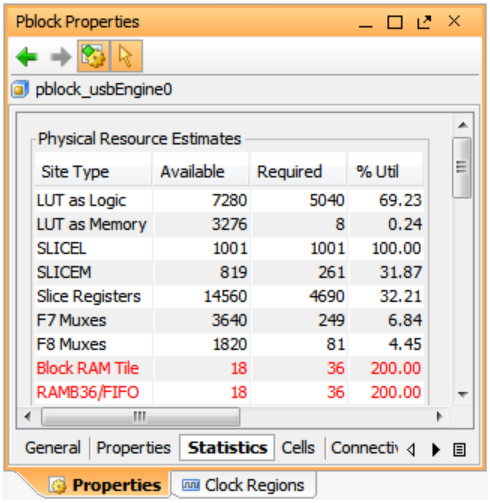


Figure 47: Pblock Properties - Statistics

4. Select **pblock_usbEngine0** in the Device window.
5. Right click and select the **Set Pblock Size** command from the popup menu.
6. **Draw a rectangle** to cover Clock Region X0Y1, on the left side the Device window.

If you size the Pblock to completely cover the clock region borders, you'll be prompted to confirm the clock region as the Pblock range. If the Pblock does not encompass the whole clock region, you'll be prompted to accept the logic types contained in the Pblock area.

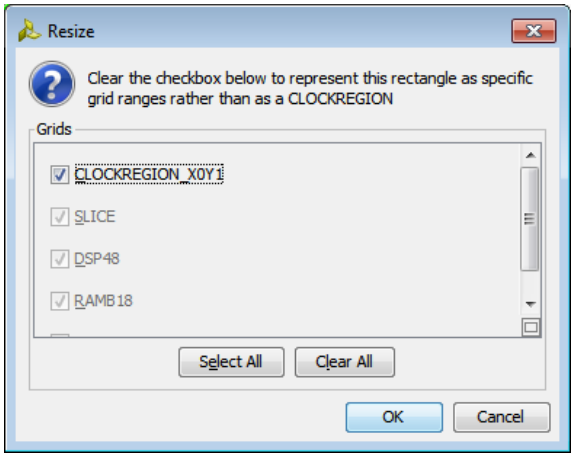


Figure 48: Set Pblock Size

7. **Click OK** to accept either the clock region, or the included logic.

   At this time you can review the Statistics of the Pblock Properties window, and see that with only a single clock region `pblock_usbEngine0` is not large enough to include the required logic resources, as shown in Figure 49.

8. **Scroll through** the **Pblock Properties** window to examine the different types of information presented.

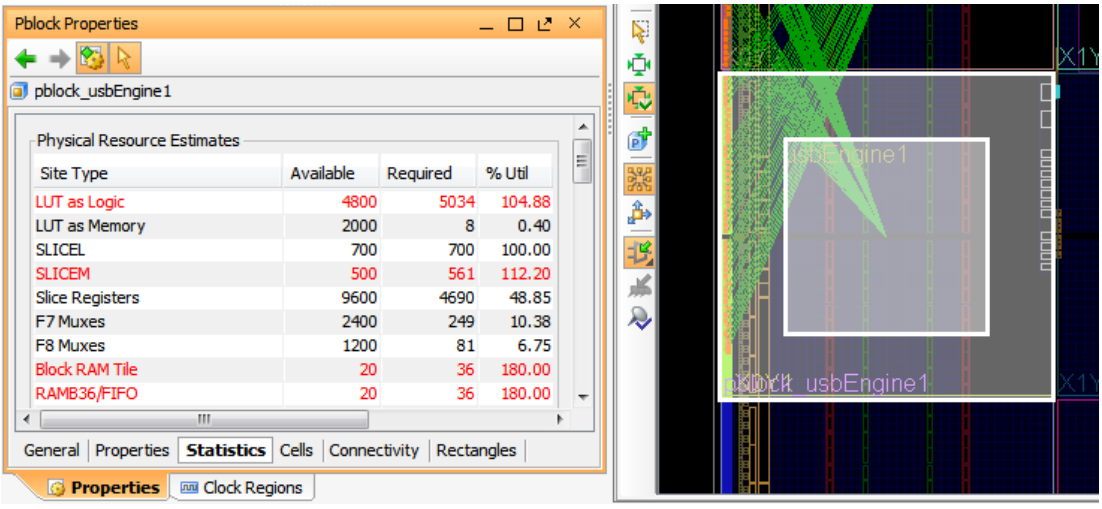


Figure 49: usbEngine0 Pblock Properties

9. **Click on** the bottom of the Pblock **and stretch** it all way to the bottom edge of the device and click **OK** to accept the Clock Region as the Pblock range.

10. **Resize** the **Pblock** rectangle to cover Clock Region X0Y0 and X0Y1 in the lower left of the Device window.
11. **Click OK** in the Resize dialog box to accept the Clock Regions as the area of the Pblock.
12. Select **pblock_usbEngine1** and resize the Pblock to cover both Clock Region X0Y2 and X0Y3 in the upper left of the Device window.
13. **Click OK** in the Resize dialog box again.

    The final Pblock placement is shown in Figure 50.

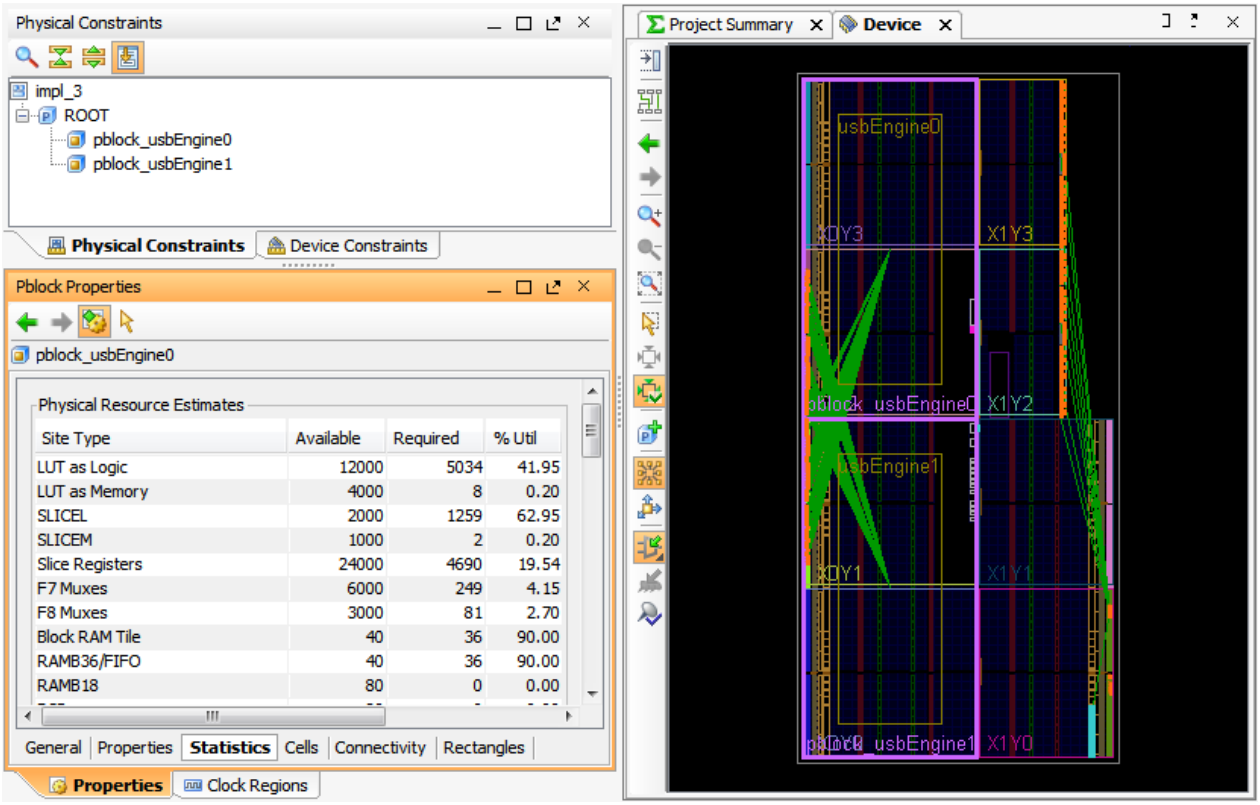


Figure 50: usbEngine0 and usbEngine1 Pblocks Placed

With new constraints created for the `usbEngine0` and `usbEngine1` Pblock placement, you must save the design constraints. It is a good idea to save the constraints into a new constraints set so that you can keep track of different design strategies. This allows you to revert to an older design strategy if needed, or develop and implement different strategies for a single design.

14. Select the **File > Save Constraints As** command.

    This opens the Save Constraints As dialog box as shown in Figure 51.

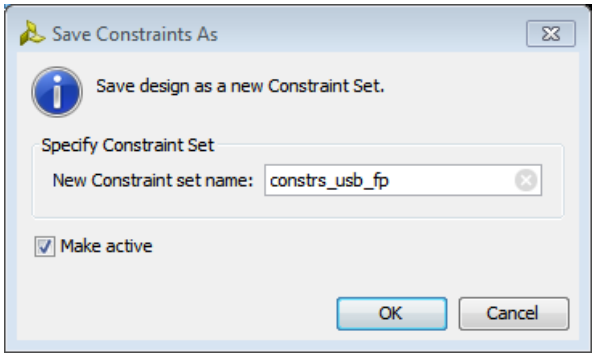


Figure 51: Saving a new Constraint Set

15. **Save** the new constraint set as **constrs_usb_fp**, for the usbEngine floorplan.
16. Make sure the **Make active** checkbox is selected, and click **OK**.

    A new constraint set, called `constrs_usb_fp`, is created to store the floorplan constraints you have created here. The `top_full.xdc` file is copied to the new constraint set, and the constraints are updated as needed to reflect the current state of the design.
17. In the Sources window, expand the **constrs_usb_fp** constraint set, and double-click the **top_full.xdc** file to open it in the Text Editor.

    Scroll through the open constraint file to examine the constraints. Toward the bottom of the file you will see the new Pblock constraints.

    When you save the constraint set, the Synthesized Netlist and Implemented Design may go out of date, as reflected in the project status bar. In some cases, the design may go out-of-date as the result of small changes that you know do not actually affect synthesis or implementation. When this happens, you can force the design into an up-to-date state, as shown in Figure 52.

    In this case, the changes you made do not affect the Synthesized Netlist, so you will not need to rerun synthesis. However, clearing the placement and floorplanning the usbEngine blocks means that the Implemented Design is out-of-date, and implementation will need to be run again. You can force the design into an up-to-date state for synthesis, and then rerun implementation.
18. Click the **more info** link in the project status bar.

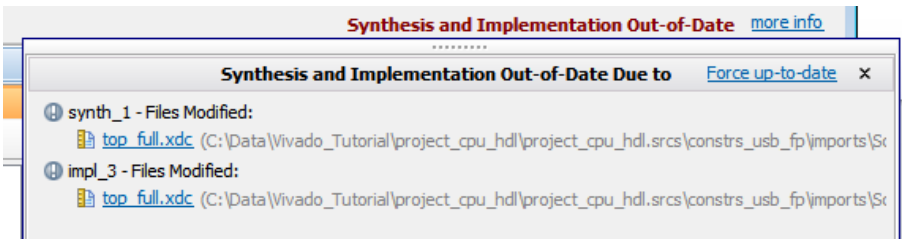


Figure 52: Force Up-to-date

This opens the Synthesis and Implementation Out-of-Date dialog box which displays some possible reasons why the design may be out-of-date.

19. Click the **Force up-to-date** link in the upper right of the dialog box.

    The project status bar is updated to reflect the up-to-date condition for the design.

Step 5: Running Implementation with the Floorplan

With the floorplan of the `usbEngine` blocks completed, you are now ready to rerun implementation.

1. Select the **Design Runs** window to make it active.
2. Right click and select the **Create Runs** command from the popup menu.
3. Select **Implementation** as the type of runs to create, and click **Next**.

   This opens the Configure Implementation Runs dialog box.
4. Select the following options, and **click Next**:
   - **impl_usb_fp** for the Name,
   - **synth_1** for the Synth Name,
   - **constrs_usb_fp** for Constraints Set,
   - **xc7k70tfbg676-2** for Part,
   - **Flow_RuntimeOptimized** for Strategy.
   - **Make Active**

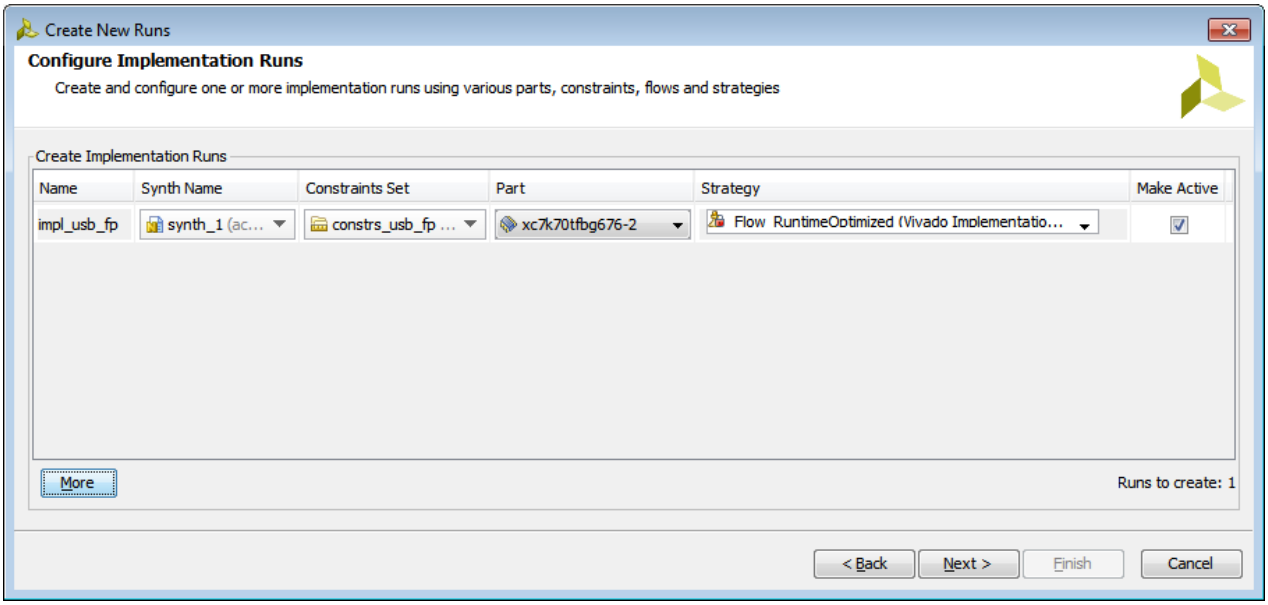


Figure 53: Floorplan Implementation Run

5. In the Launch Options dialog box, select **Do not launch now** and click **Next**.
6. On the Create New Runs Summary, check that the settings are correct, and click **Finish**.

   A new implementation run is created with the specified properties.
7. From the Flow Navigator menu, select **Run Implementation**.

   The implementation run is launched with the usbEngine blocks placed according to your floorplan. The Implementation Completed dialog box displays when the run is finished.
8. Select **Open Implemented Design**, and click **OK**.

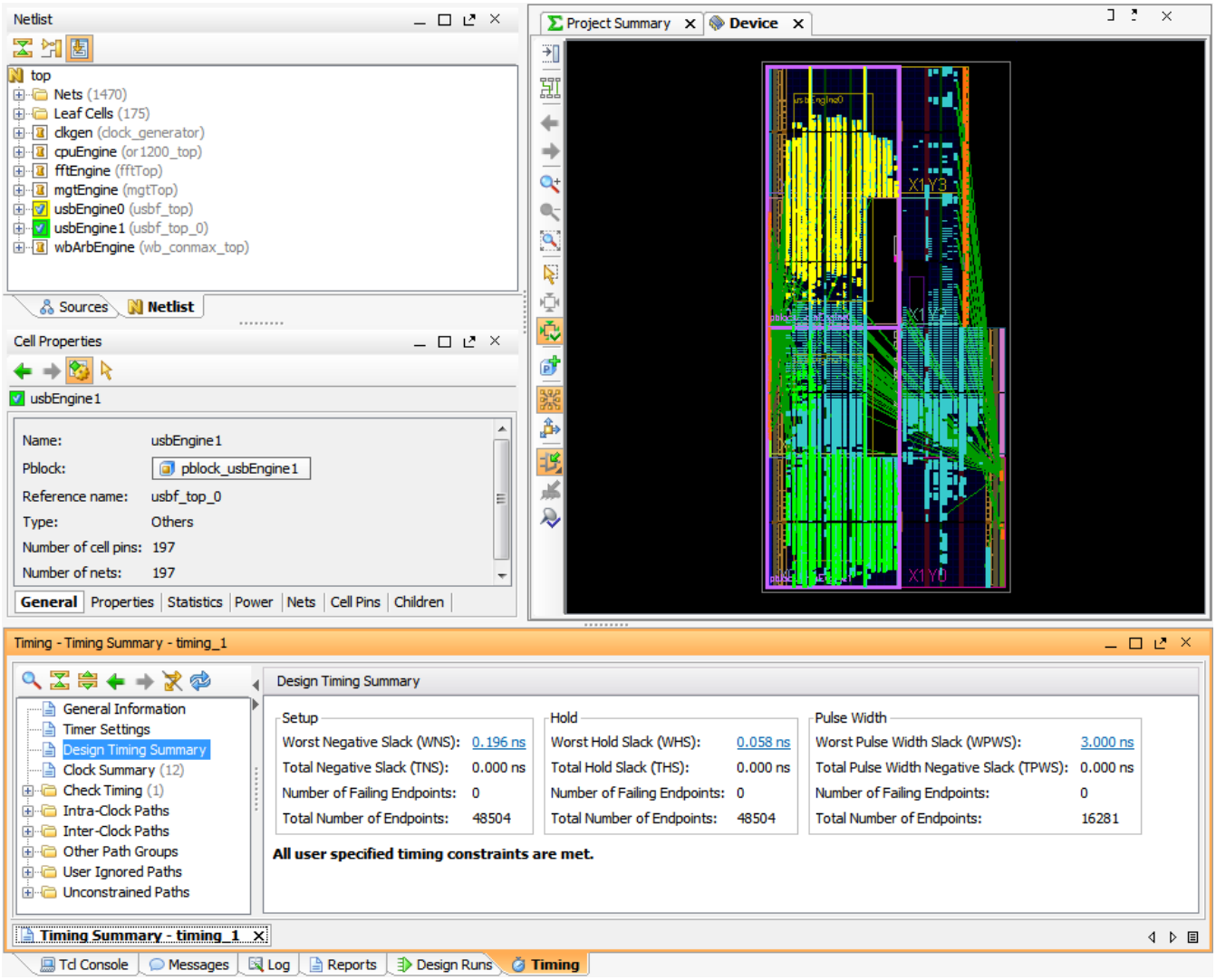


Figure 54: Implemented Floorplan

The design has been implemented with the floorplan you created for the usbEngine blocks. You can see this by highlighting the blocks as you did earlier in this tutorial.

9. **Select** the **Default Layout** from the View Layout drop down menu.
10. In the Netlist window, select **usbEngine0** and **usbEngine1**.
11. Right click and select **Highlight Leaf Cells > Cycle Colors** from the popup menu.

    The usbEngine blocks are highlighted in the Device window, and Netlist window, as seen in Figure 54.
12. In the Flow Navigator under Implemented Design, select **Report Timing Summary**.
13. Click **OK** in the Report Timing Summary dialog box to run Vivado timing analysis with the default settings.

    The Timing Summary Report now shows that all timing has been met as a result of the improved placement of your floorplan, and the improved routing results.

Conclusion

In this tutorial, you used the Vivado IDE to explore and analyze the synthesized design and targeted device prior to running the implementation tools. This enabled you to find potential design issues and errors early in the design cycle, rather than discovering issues during implementation. In addition, you used the graphical presentation of design resource estimates, design rule violations, timing estimation, constraints, and connectivity to help you understand your design and any areas with potential issues.

After running the design through the synthesis and implementation tools, you:

- Viewed implementation results and examined timing results.
- Analyzed critical path objects in the schematic, and selected the parent modules of those path objects.
- Highlighted module placement and displayed the connectivity of the modules using the Show Connectivity command.
- Analyzed placement and routing of critical timing paths.
- Floorplanned possible placements based on the hierarchy of the design.
- Implemented the floorplanned design