Vivado Design Suite User Guide

Design Analysis and Closure Techniques

UG906 (v2014.1) May 14, 2014

XILINX
ALL PROGRAMMABLE™

Revision History

Date	Version	Revision
05/14/2014	2014.1	Updates to document for Vivado® Design Suite, 2014.1 release. Updates include: • Added Validating Design Methodology Logic DRCs, page 16. • Added Clock Groups information to Report Exception, page 61.

Table of Contents

Design Analysis Within the IDE

Introduction to Design Analysis Within the IDE

This chapter provides an introduction to design analysis in the Xilinx® Vivado® Integrated Design Environment (IDE), including:

- Logic Analysis Features
- Timing Analysis Features
- Implementation Results Analysis Features

For more information about managing windows and using the Vivado IDE, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1].

Logic Analysis Features

This section discusses Logic Analysis Features, and includes:

- Using the Netlist Window
- Using the Hierarchy Window
- Using the Schematic Window
- Searching for Objects Using Find
- Analyzing Device Utilization Statistics
- Using Report DRC
- Validating Design Methodology Logic DRCs

Using the Netlist Window

The Netlist Window shows the design hierarchy as it is in the netlist, processed by the synthesis tools.

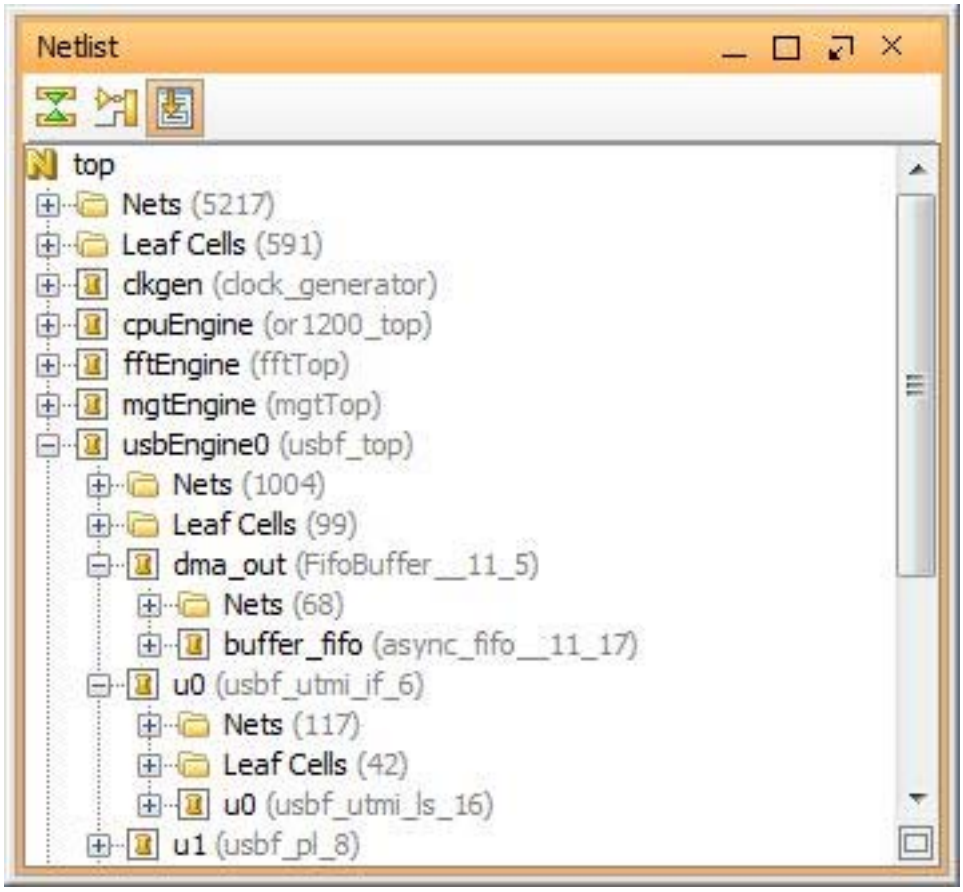


Figure 1-1: **Netlist Window**

Depending on synthesis settings, the netlist hierarchy may be a one hundred percent match for the original RTL, or there may be no hierarchy. Generally, the synthesis tools default to preserving most of the user hierarchy while optimizing the logic. This results in a smaller, faster netlist.

With the synthesis tool defaults, the netlist hierarchy is recognizable, but the interfaces to the hierarchies may be modified. Some pins and levels of hierarchy may be missing.

Each level of hierarchy shows its hierarchy tree. At each level, the tool shows:

- A nets folder for any nets at that level
- A Leaf Cells folder if there are hardware primitive instances at that level
- Any hierarchies instantiated at that level

Traversing the tree shows the whole branch. The icons next to the cells display information about the state of the design.

For more information, see *Using the Netlist Window* in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1].

The Properties Window for each level of hierarchy shows utilization statistics including:

- Primitive usage for the whole hierarchical branch, grouped in higher level buckets
- The number of nets crossing the hierarchy boundary
- Clocks used in the hierarchy

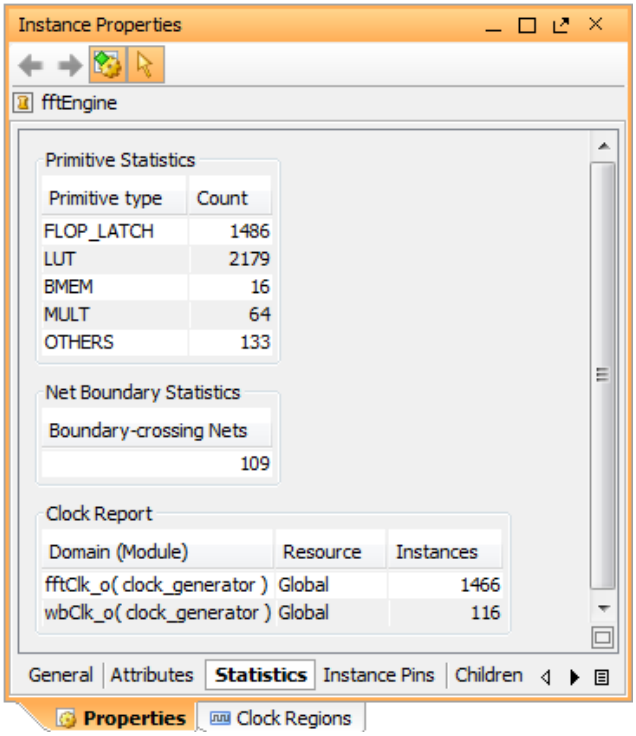


Figure 1-2: **Instance Properties Window**

If you floorplan the design, similar properties are displayed for the Pblock.

Using the Hierarchy Window

Explore the hierarchy to understand resource usage. To open the Hierarchy Window, select **Tools > Show Hierarchy**, or from the Netlist window, click **F6**.

The Hierarchy Window displays the hierarchy tree for the netlist. Each horizontal row displays a level of hierarchy inside the netlist. As you move down the rows, you move into deeper netlist hierarchy. Across the row, each level of hierarchy is sized relative to the other hierarchy at that level.

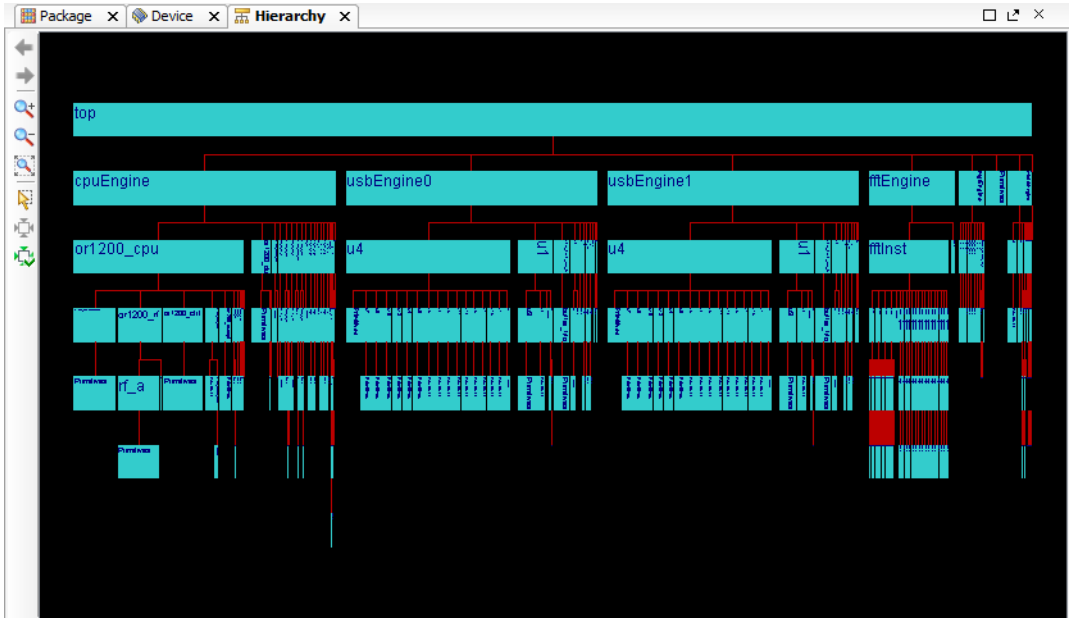


Figure 1-3: **Hierarchy Window**

Figure 1-3, Hierarchy Window, shows that `cpuEngine`, `usbEngine0`, and `usbEngine1`:

- Have most of the logic in the design.
- All use about the same number of resources.

The Utilization Report:

- Breaks apart the design based on resource type.
- Displays each resource type independently with consumption per level of hierarchy.

To view the Utilization Report, select **Tools > Report > Report Utilization**.

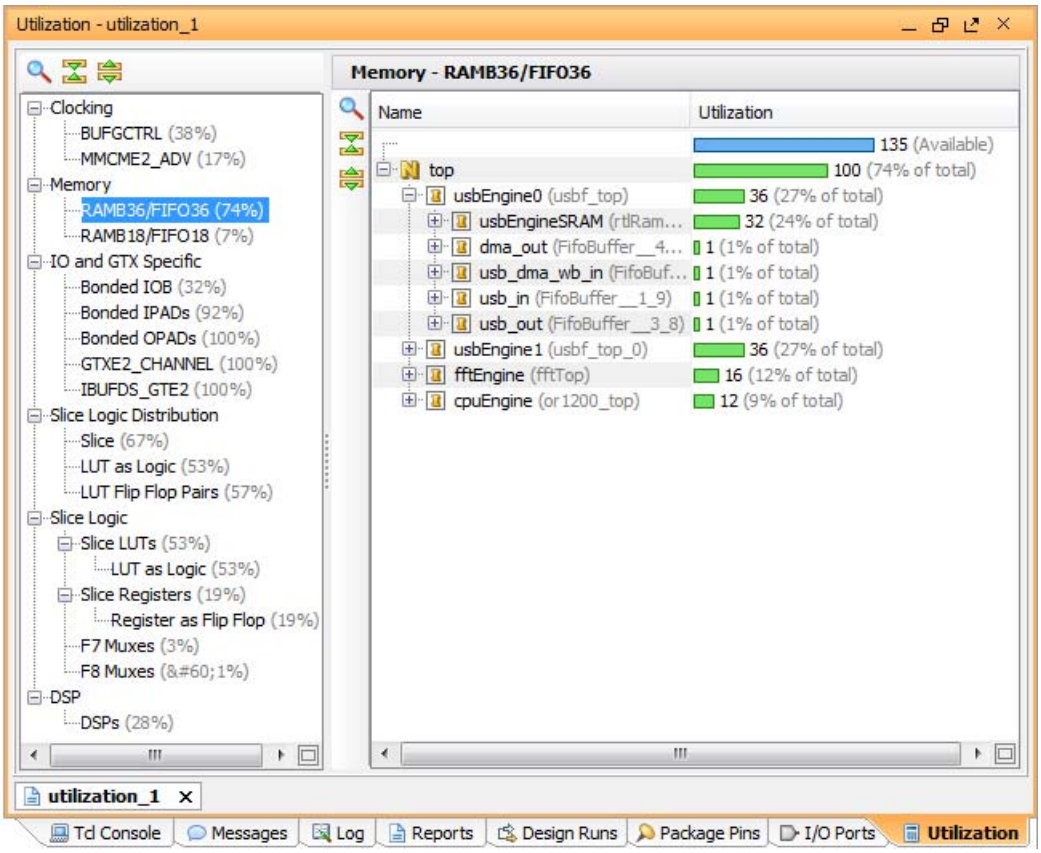


Figure 1-4: **Utilization Report**

In this design, the two `usbEngine` blocks are the two biggest consumers of the RAMB36 and FIFO36 blocks. Click the **+** (plus) icon to view the consumption at sub-hierarchies.

Using the Schematic Window

The schematic is a graphical representation of the netlist. View the schematic to:

- View a graphical representation for the netlist.
- Review the gates, hierarchies, and connectivity.
- Trace and expand cones of logic.
- Analyze the design.
- Better understand what is happening inside the design.

At the RTL level in Elaborated Design, you see how the tool has interpreted your code. In Synthesize Design and Implemented Design, you see the gates generated by the synthesis tool.

To open the schematic, select **Tools > Schematic**. If nothing is selected, the gates, hierarchy, and connectivity appear at the top level of the design.

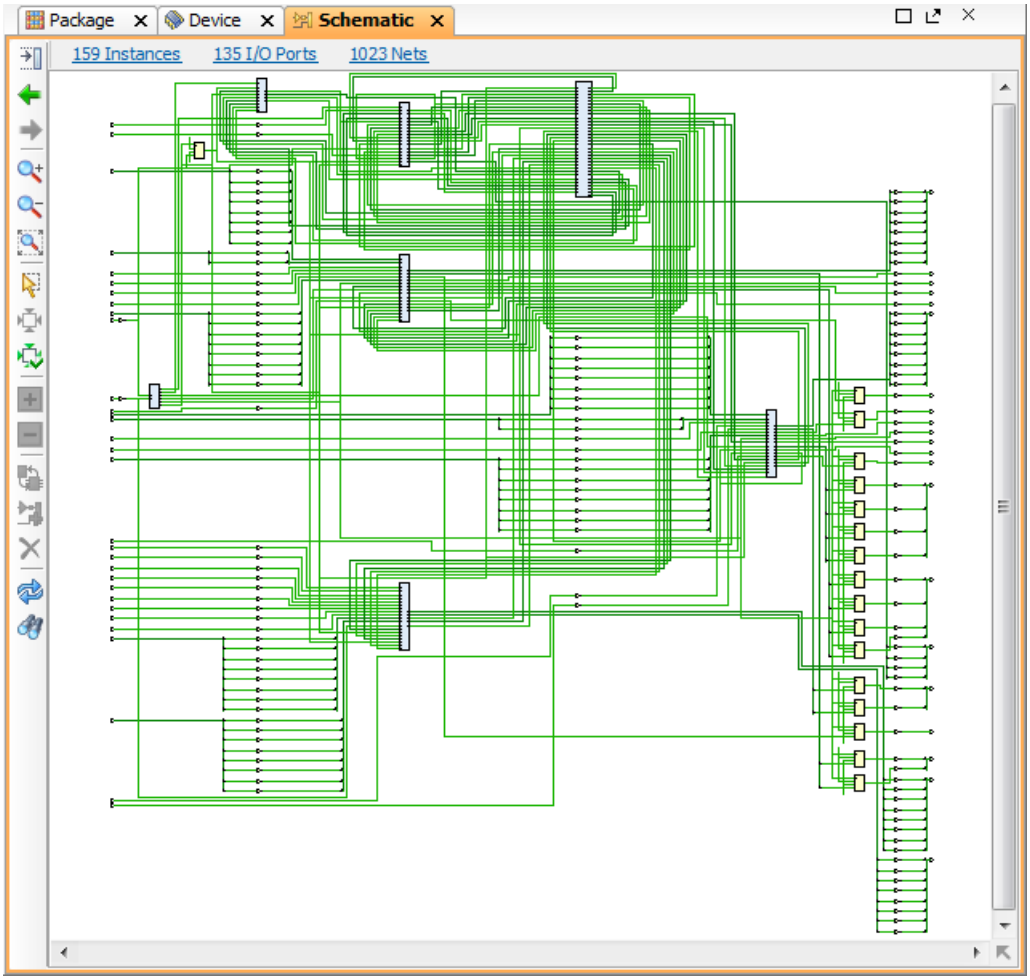


Figure 1-5: **Top Level Schematic**

For information about zooming and moving around the schematic, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1].

TIP: *The schematic is simpler if you use a single level of hierarchy only. The schematic populates with the selected element emphasized (blue). The ports for the single hierarchy display.*

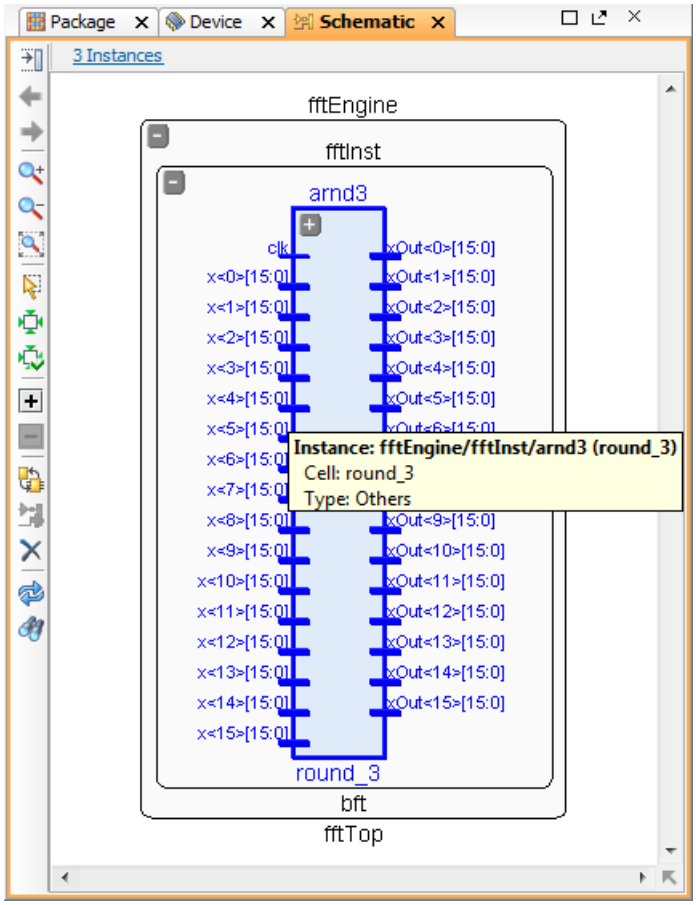


Figure 1-6: **Schematic with Single Hierarchy Selected**

You can trace the schematic in multiple ways:

- Click the **+** (plus) icon in the upper left to display the gates in the hierarchy.
- Double click a port or element to expand it.
- Use the schematic popup.

  For more information, see *Using the Schematic Window* in the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1].

- Click the **<- ->** arrows to switch between the previous and next schematic views.
- Select **Expand All** to display more logic and connectivity.
- Select **Collapse All** to simplify the schematic.

After implementation, the schematic is the easiest way to visualize the gates in a timing path. Select the path, then open the schematic with the gates and nets from that path.

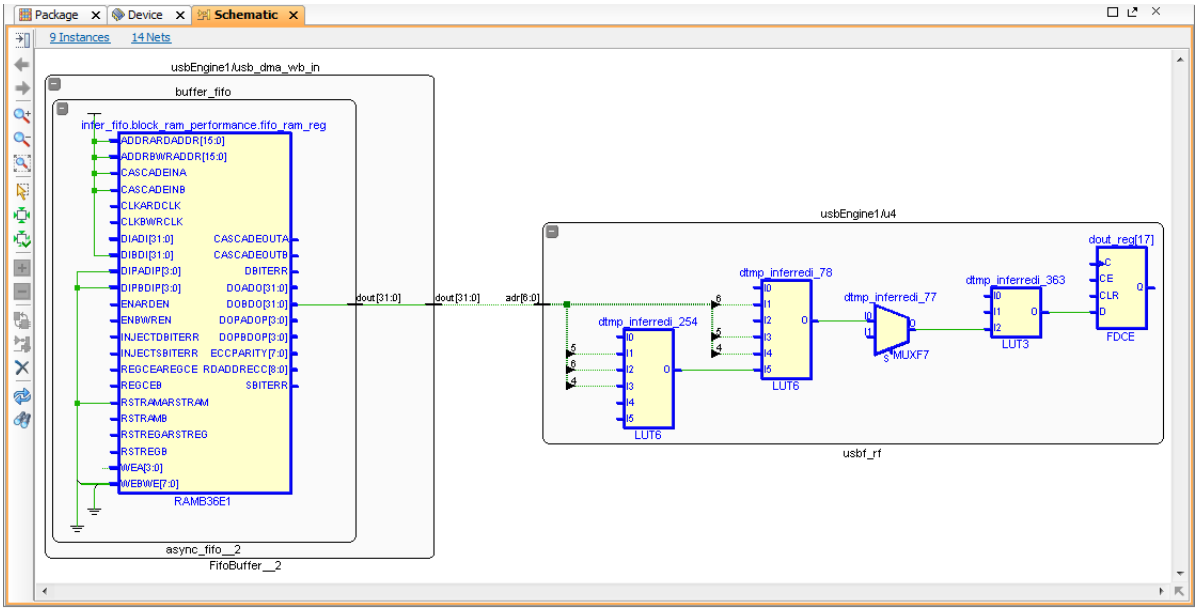


Figure 1-7: **Schematic With Timing Path**

To identify the relevant levels of hierarchy in the schematic, choose **Select Leaf Cell Parents** from the popup menu.

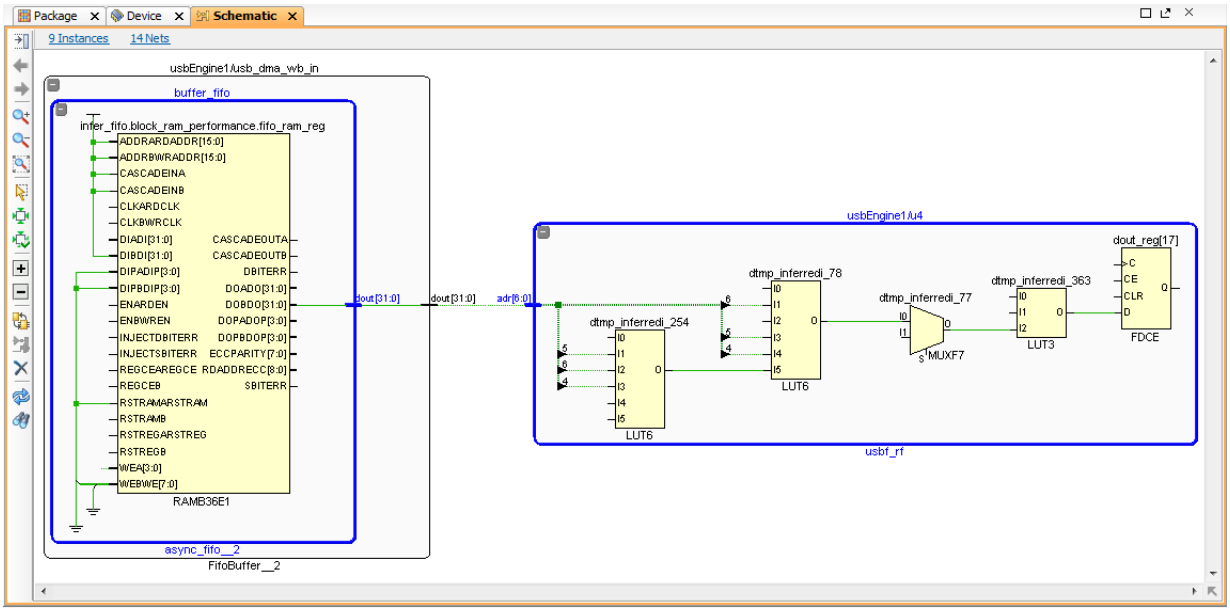


Figure 1-8: **Timing Path with Select Primitive Parents**

As you review the schematic, select the **Highlight** and **Mark** commands to track gates of interest. Color coding primitives (using either a mark or a highlight) makes it easier to track which logic was in the original path, and which logic was added.

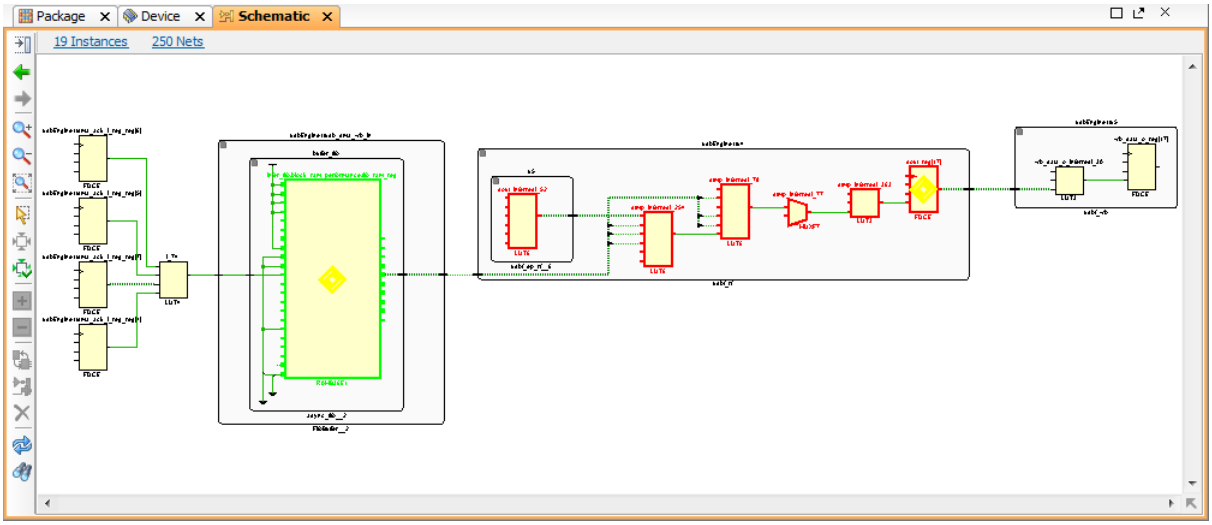


Figure 1-9: **Schematic With Timing Path Marked**

Searching for Objects Using Find

The Vivado IDE includes powerful find and search capabilities using **Edit > Find**.

Note: You can also open the Find window by pressing **Ctrl+F**.

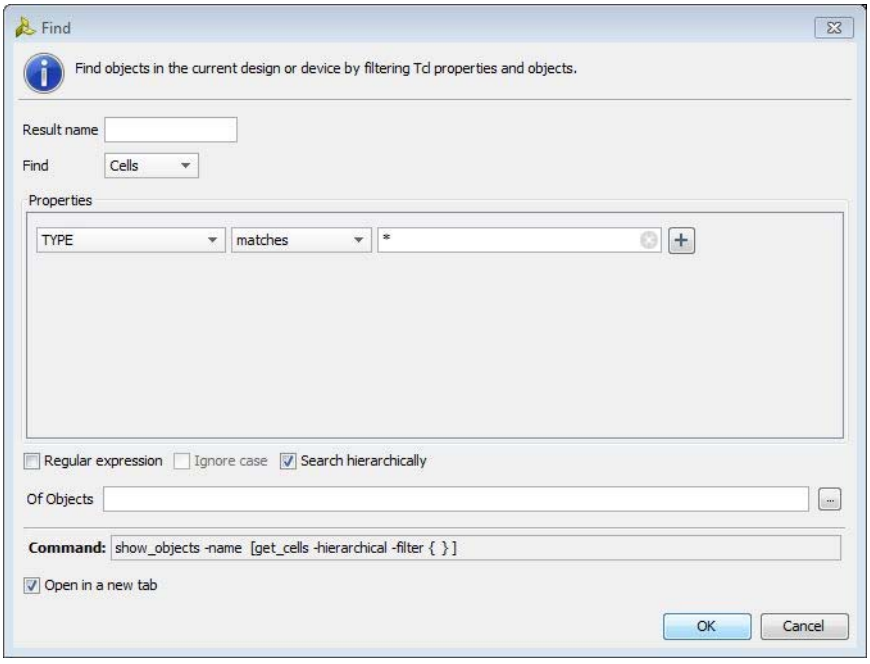


Figure 1-10: **Find Dialog Box**

Find Criteria

Select **Edit > Find** to search the netlist for:

- Instances
- Nets
- I/O Ports
- Instance Pins
- Pblocks
- RPMs

Find Sub Criteria

There are multiple sub criteria for each of the **Find** criteria. For example, instance has the following sub criteria:

- Type
- Cell types
- Black boxes
- Primitives, including
  - I/O Buffer
  - Block Arithmetic
  - Block Memory
  - LUT
- Name
- Status
- Parent Pblock
- Module
- Primitive count
- Attribute

Review the other **Find** criteria for their sub fields.

Device-Specific Find Criteria

Device-specific **Find** criteria are:

- Arcs
- Nodes
- BEL Pins
- BELs
- Site Pins
- Sites
- Tiles
- I/O Banks
- Clock Regions

Find Examples

Select **Edit > Find** to find, for example:

- All unplaced I/Os
- Only the tool-placed Global Clocks
- All nets with a fanout over 10,000
- All DSPs using the PREG embedded register

Complex Finds

To run a complex find:

1. Set the first search criterion.
2. Click **+**(plus).
3. Add additional criteria.
4. Join the additional criteria with logical operators (and, or).

Tcl Finds

Run the Tcl `Find` commands when running from a script or in the Tcl Console.

TIP: *The Tcl Console at the bottom of the Vivado IDE shows the Vivado Design Suite Tcl commands run for each action executed in the GUI. From the Tcl Console, you can also enter Vivado Design Suite Tcl commands.*

For more information on Tcl scripting, see the *Vivado Design Suite User Guide: Using Tcl Scripting (UG894)* [Ref 2].

For more information on Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3], or type `<command> -help`.

Analyzing Device Utilization Statistics

A common cause of implementation issues comes from not considering the logic and device layout implied by the pinout. Slice logic is uniform in most devices. However, specialized resources such as the following impact logic placement:

- I/O
- High Performance Banks
- High Range Banks
- MGT
- DSP48
- Block RAM
- MMCM
- BUFG
- BUFR

Blocks that are large consumers of a certain specialized resource may have to spread around the device. Take this into account when designing the interface with the rest of the design. Use a combination of the following to find block resources:

- report_utilization
- netlist properties
- Pblock properties

Using Report DRC

Design Rule Checks (DRCs) check the design and report on common issues. Run DRCs using the `report_drc` command. During implementation, the tools also run DRCS. The DRCs become more complete and comprehensive with placement and routing.

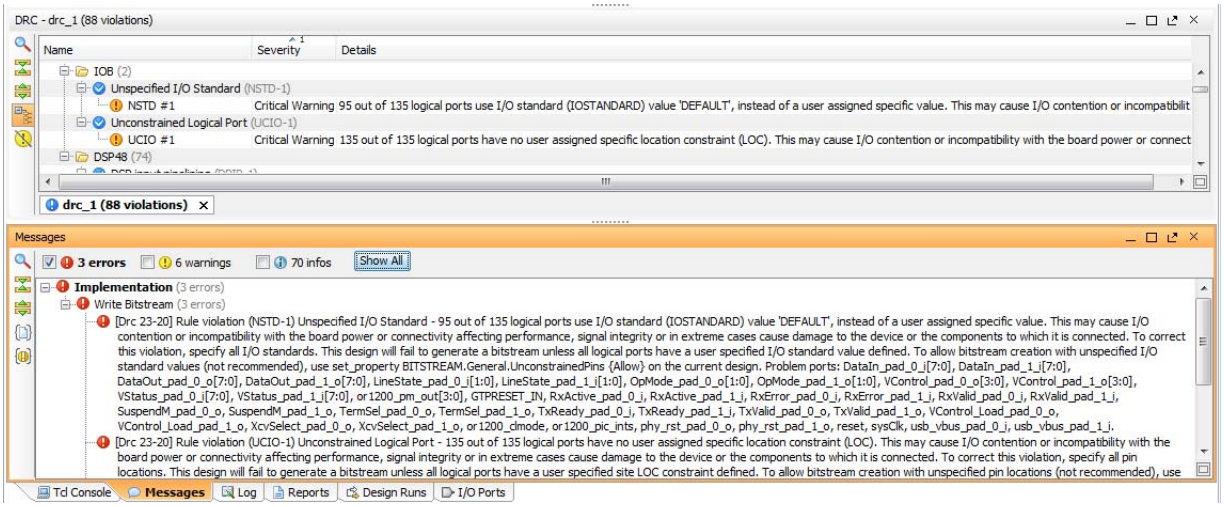


Figure 1-11: **Showing Critical Warnings and Error**

RECOMMENDED: *Review the DRC messages, Critical Warnings, and Warnings early in the flow to prevent issues later.*

At Synthesized Design, the optional Report DRC step reports a Critical Warning for the unconstrained I/Os. The routed design DRC report reports the Critical Warnings. You must review the report. At `write_bitstream`, the DRC has been elevated to an Error. Review the DRC reports early to identify areas of the design that need modification.

Validating Design Methodology Logic DRCs

Due to the importance of methodology, the Vivado tools provide a set of Design Rule Checks (DRCs) that specifically check for compliance with methodology. There are different types of DRCs depending on the stage of the design process. RTL lint-style checks are run on the elaborated RTL design; netlist-based logic and constraint checks are run on the synthesized design; and implementation and timing checks are run on the implemented design.

To run these checks at the Tcl prompt, open the design to be validated and enter following Tcl command:

```
report_drc  -ruledeck methodology_checks
```

To run these checks from the IDE, open the design to be validated and run the Report DRC command. Once the dialog appears, select the methodology checks Rule deck, as shown in

Figure 1-12.

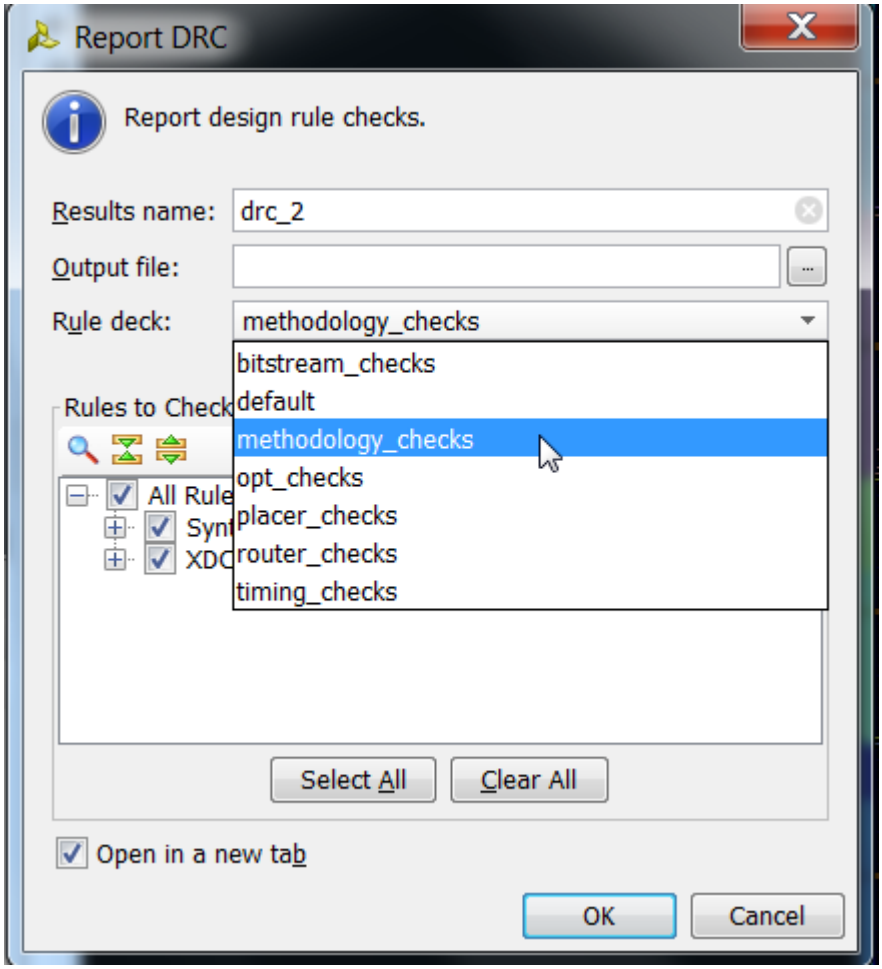


Figure 1-12: **Report DRC Dialog Box**

Violations (if there are any) are listed in the DRC window, as shown in Figure 1-13.

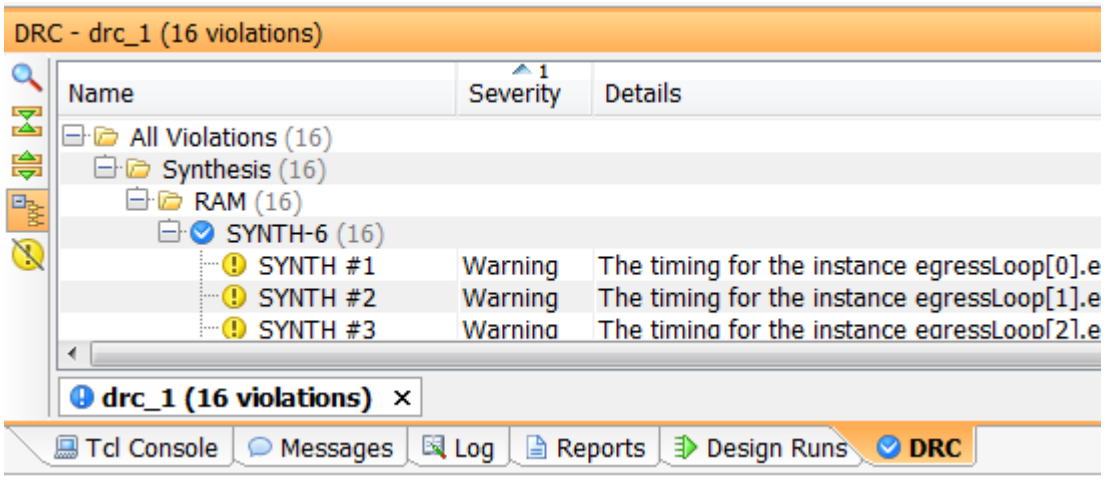


Figure 1-13: **DRC Violations**

For more information on running design methodology DRCs, refer to *Vivado Design Suite User Guide: System-Level Design Entry* (UG895) [Ref 4] and *Design Methodology Handbook for Xilinx FPGAs* (UG949) [Ref 5].

Timing Analysis Features

This section discusses Timing Analysis Features and includes:

- Report Timing Summary
- Report Timing Summary Dialog Box
- Details of the Timing Summary Report
- Report Clock Networks
- Report Clock Interaction
- Report Exception

Report Timing Summary

Timing analysis is available anywhere in the flow after synthesis. You can review the Timing Summary report files automatically created by the Synthesis and Implementation runs.

If your synthesized or implemented design is loaded in memory, you can also generate an interactive Timing Summary report from:

- **Flow Navigator > Synthesis**
- **Flow Navigator > Implementation**
- **Main Menu > Tools > Timing > Report Timing Summary**

Equivalent Tcl command: `report_timing_summary`

For more information on the `report_timing_summary` options, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3].

In a synthesized design, the Vivado IDE timing engine estimates the net delays based on connectivity and fanout. The accuracy of the delays is greater for nets between cells that are already placed by the user.

In an Implemented Design, the net delays are based on the actual routing information. You must use the Timing Summary report for timing signoff if the design is completely routed. To verify that the design is completely routed, view the Route Status report.

Report Timing Summary Dialog Box

In the Vivado IDE, the Report Timing Summary dialog box includes the following tabs:

- Options Tab
- Advanced Tab
- Timer Settings Tab

The Results name field at the top of the Report Timing Summary dialog box specifies the name of the graphical report that opens in the Results window. The graphical version of the report includes hyperlinks that allow you to cross-reference nets and cells from the report to Device and Schematic windows, and design source files.

If this field is left empty, the report is returned to the Tcl Console, and a graphical version of the report is not opened in the Results window.

Equivalent Tcl option: `-name`

Options Tab

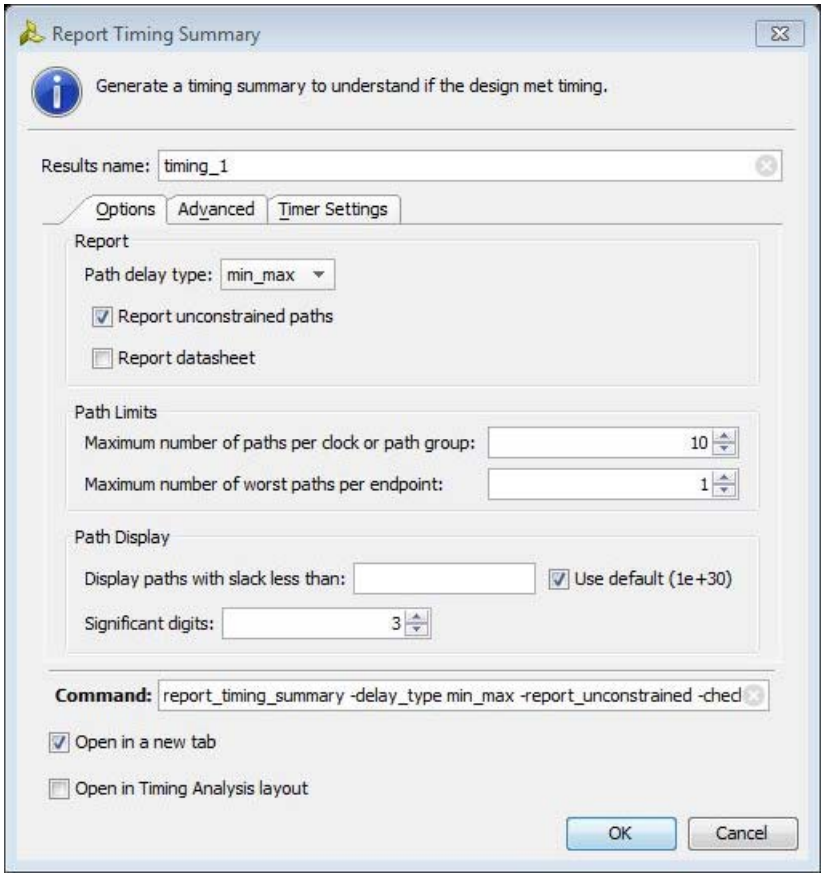


Figure 1-14: **Report Timing Summary Dialog Box: Options Tab**

The Options tab of the Report Timing Summary dialog box includes:

- Options Tab - Report Section
- Options Tab- Path Limits Section
- Options Tab - Path Display
- Options Tab - Common Section

Options Tab - Report Section

The Report section of the Options tab of the Report Timing Summary dialog box includes:

- **Path delay type**

  Sets the type of analysis to be run. For synthesized designs, only max delay analysis (setup/recovery) is performed by default. For implemented design, both min and max delay analysis (setup/hold, recover/removal) are performed by default. To run min delay analysis only (hold and removal), select delay type `min`.

  Equivalent Tcl option: `-delay_type`

- **Report unconstrained paths**

  Generates information on paths that do not have timing requirements. This option is checked by default in the Vivado IDE, but is not turned on by default in the equivalent Tcl command `report_timing_summary`.

  Equivalent Tcl option: `-report_unconstrained`

- **Report datasheet**

  Generates the design datasheet as defined in Report Datasheet, page 55, in this chapter.

  Equivalent Tcl option: `-datasheet`

Options Tab- Path Limits Section

The Path Limits section of the Options tab of the Report Timing Summary dialog box includes:

- **Maximum number of paths per clock or path group**

  Controls the maximum number of paths reported per clock pair or path group.

  Equivalent Tcl option: `-max_paths`

- **Maximum number of worst paths per endpoint**

  Controls the maximum number of paths potentially reported per path endpoint. This limit is bounded by the maximum number of paths per clock pair or path group.

  Equivalent Tcl option: `-nworst`

Options Tab - Path Display

The Path Display section of the Options tab of the Report Timing Summary dialog box includes:

- **Display paths with slack less than**

  Filters the reported paths based on their slack value. This option does not affect the content of the summary tables.

  Equivalent Tcl option: `-slack_lesser_than`

- **Number of significant digits**

  Controls the accuracy of the numbers displayed in the report.

  Equivalent Tcl option: `-significant_digits`

Options Tab - Common Section

The following controls common to all three tabs are located at the bottom of the Report Timing Summary dialog box:

- **Command**

  Displays the Tcl command line equivalent of the various options specified in the Report Timing Summary dialog box.

- **Open in a New Tab**

  Opens the results in a new tab, or to replace the last tab opened by the Results window.

- **Open in Timing Analysis Layout**

  Resets the current view layout to the Timing Analysis view layout.

  For more information on view layouts, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1].

Advanced Tab

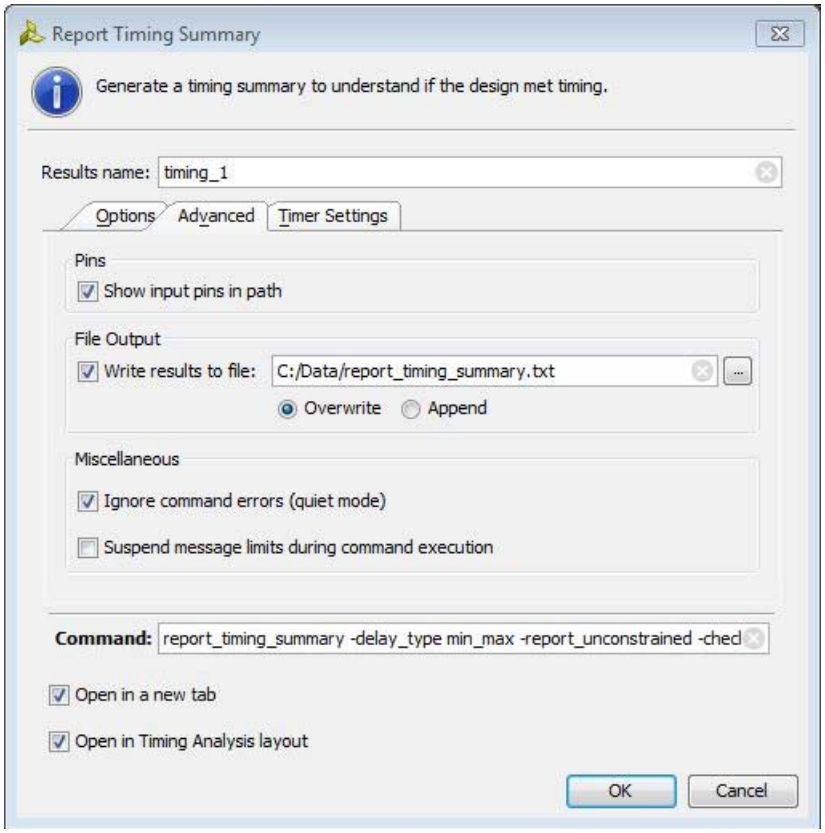


Figure 1-15: **Report Timing Summary Dialog Box: Advanced Tab**

The Advanced tab of the Report Timing Summary dialog box contains the following sections:

Pins

- **Show input pins in path**

RECOMMENDED: *Keep this option selected.*

Equivalent Tcl option: `-input_pins`

File Output

- **Write results to file**

  Writes the result to the specified file name. By default the report is written to the Timing window in the Vivado IDE.

  Equivalent Tcl option: `-file`

- **Overwrite / Append**

  When the report is written to a file, determines whether (1) the specified file is overwritten, or (2) new information is appended to an existing report.

  Equivalent Tcl option: `-append`

Miscellaneous

- **Ignore command errors**

  Executes the command quietly, ignoring any command line errors and returning no messages. The command also returns TCL_OK regardless of any errors encountered during execution.

  Equivalent Tcl option: `-quiet`

- **Suspend message limits**

  Temporarily overrides any message limits and return all messages.

  Equivalent Tcl option: `-verbose`

Timer Settings Tab

To set the timer settings, use either: (1) one of the Vivado IDE timing analysis dialog boxes; or, (2) one of the Tcl commands listed in this section.

These settings affect other timing-related commands run in the same Vivado IDE session, except the synthesis and implementation commands.

The timer settings are not saved as a tool preference. The default values are restored for each new session.

RECOMMENDED: *Do not change the default values. Keeping the default values provides maximum timing analysis coverage with the most accurate delay values.*

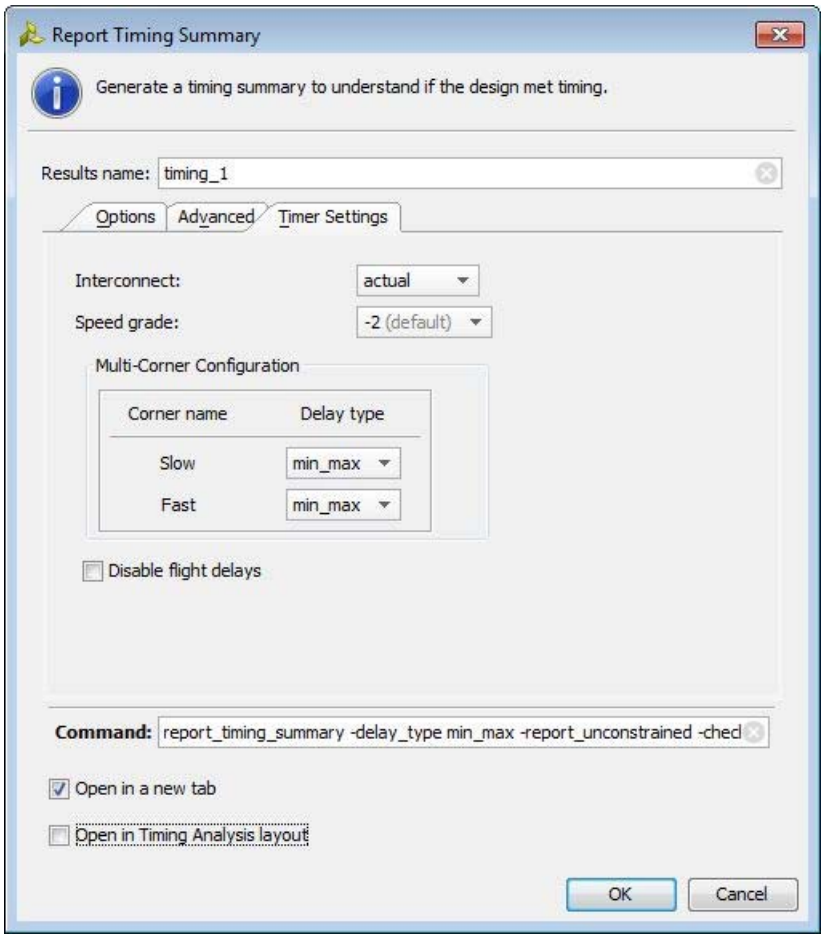


Figure 1-16: **Report Timing Summary Dialog Box: Timer Settings Tab**

- **Interconnect**

  Controls whether net delays are calculated based on the estimated route distance between leaf cell pins, by the actual routed net, or excludes net delay from timing analysis.

  This option is automatically set to **Estimated** for post-synthesis designs, and to **Actual** for post-implementation designs.

  - **Estimated**

    For unplaced cells, the net delay value corresponds to the delay of the best possible placement, based on the nature of the driver and loads as well as the fanout. A net between unplaced leaf cell pins is labeled **unplaced** in the timing path report.

    For placed cells, the net delay depends on the distance between the driver and the load as well as the fanout. This net is labeled **estimated** in the timing path report.

  - **Actual**

    For routed nets, the net delay corresponds to the actual hardware delay of the routed interconnect. This net is labeled **routed** in the timing path report.

  - **None**

    Interconnect delays are not considered in the timing report and net delays are forced to zero.

  Equivalent Tcl command: `set_delay_model`

- **Speed Grade**

  Sets the device speed grade. By default, this option is set based on the part selected when creating a project or opening a design checkpoint.

  You can change this option to report timing on the same design database against another speed grade without rerunning the complete implementation flow.

  Equivalent Tcl command: `set_speed_grade`

- **Multi-Corner Configuration**

  Specifies the type of path delays to be analyzed for the specified timing corner. Valid values are **none**, **max**, **min**, and **min_max**. Select **none** to disable timing analysis for the specified corner.

RECOMMENDED: *Keep both setup (max) and hold (min) analysis selected for both corners.*

  Equivalent Tcl command: `config_timing_corners`

- **Disable flight delays**

  Do not add package delays to I/O delay calculations.

  Equivalent Tcl command: `config_timing_analysis`

Details of the Timing Summary Report

The Timing Summary Report contains the following sections:

- General Information Section
- Timer Settings Section
- Design Timing Summary Section
- Clock Summary Section
- Check Timing Section
- Intra-Clock Paths Section
- Inter-Clock Paths Section
- Path Groups Section
- User-Ignored Paths Section
- Unconstrained Paths Section

The comprehensive information contained in the Timing Summary Report is similar to the information provided by several reports available from the Vivado IDE (Report Clock Interaction, Report Pulse Width, Report Timing, Check Timing) and to some of the reports available in Tcl only (`report_clocks`).

However, the Report Timing Summary also includes information that is unique to this report, such as Unconstrained Paths.

General Information Section

The General Information section of the Timing Summary Report provides information about the:

- Design name
- Selected device, package, and speed grade (with the speed file version)
- Vivado Design Suite release
- Current date
- Equivalent Tcl commands executed to generate the report

Timer Settings Section

The Timer Settings section of the Timing Summary Report contains details on the Vivado IDE timing analysis engine settings used to generate the timing information in the report.

Figure 1-16, Report Timing Summary Dialog Box: Timer Settings Tab, page 24, shows an example of the Timer Settings section, which includes:

- **Enable Multi-Corner Analysis**

  This analysis is enabled for each corner (Multi-Corner Configuration).

- **Enable Pessimism Removal (and Pessimism Removal Resolution)**

  Ensures that the source and destination clocks of each path are reported with no skew at their common node.

  Note: This setting must always be enabled.

- **Enable Input Delay Default Clock**

  Creates a default null input delay constraint on input ports with no user constraint. It is disabled by default.

- **Enable Preset / Clear Arcs**

  Enables timing path propagation through asynchronous pins. It does not affect recovery/removal checks and is disabled by default.

- **Disable Flight Delays**

  Disables package delays for I/O delay calculations.

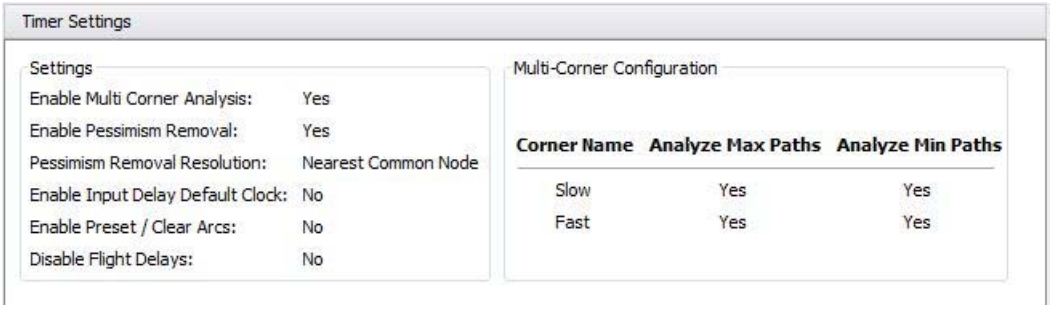


Figure 1-17: **Timing Summary Report: Timer Settings**

Design Timing Summary Section

The Design Timing Summary section of the Timing Summary Report provides a summary of the timing for the design, and combines the results of all other sections in one view.

RECOMMENDED: *Review the Design Timing Summary section to verify that all timing constraints are met after route, or to understand the status of the design at any point in the flow.*

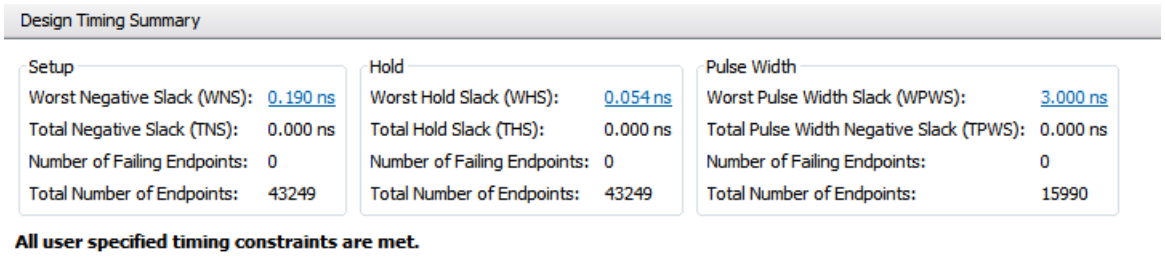


Figure 1-18: **Design Timing Summary**

The Design Timing Summary section includes the following areas:

- Setup Area (Max Delay Analysis)
- Hold Area (Min Delay Analysis)
- Pulse Width Area (Pin Switching Limits)

Setup Area (Max Delay Analysis)

The Setup area of the Design Timing Summary section displays all checks related to max delay analysis: setup, recovery, and data check.

- **Worst Negative Slack (WNS)**

  This value corresponds to the worst slack of all the timing paths for max delay analysis. It can be positive or negative.

- **Total Negative Slack (TNS)**

  The sum of all WNS violations, when considering only the worst violation of each timing path endpoint. Its value is:

  - 0ns when all timing constraints are met for max delay analysis.
  - Negative when there are some violations.

- **Number of Failing Endpoints**

  The total number of endpoints with a violation (WNS<0ns).

- **Total Number of Endpoints**

  The total number of endpoints analyzed.

Hold Area (Min Delay Analysis)

The Hold area of the Design Timing Summary section displays all checks related to min delay analysis: hold, removal, and data check.

- **Worst Hold Slack (WHS)**

  Corresponds to the worst slack of all the timing paths for min delay analysis. It can be positive or negative.

- **Total Hold Slack (THS)**

  The sum of all WHS violations, when considering only the worst violation of each timing path endpoint. Its value is:

  - 0ns when all timing constraints are met for min delay analysis.
  - Negative when there are some violations.

- **Number of Failing Endpoints**

  The total number of endpoints with a violation (WHS<0ns).

- **Total Number of Endpoints**

  The total number of endpoints analyzed.

Pulse Width Area (Pin Switching Limits)

The Pulse Width area of the Design Timing Summary section displays all checks related to pin switching limits:

- Min low pulse width
- Min high pulse width
- Min period
- Max period
- Max skew (between two clock pins of a same leaf cell).

The reported values are:

- **Worst Pulse Width Slack (WPWS)**

  Corresponds to the worst slack of all the timing checks listed above when using both min and max delays.

- **Total Pulse Width Slack (TPWS)**

  The sum of all WPWS violations, when considering only the worst violation of each pin in the design. Its value is:

  - 0ns when all related constraints are met.
  - Negative when there are some violations.

- **Number of Failing Endpoints**

  The total number of pins with a violation (WPWS< 0ns).

- **Total Number of Endpoints**

  The total number of endpoints analyzed.

Clock Summary Section

The Clock Summary section of the Timing Summary Report includes information similar to that produced by `report_clocks`:

- All the clocks in the design (whether created by `create_clock`, `create_generated_clock`, or automatically by the tool).
- The properties for each clock, such as name, period, waveform, type, and target frequency.

TIP: *The indentation of names reflects the relationship between master and generated clocks.*

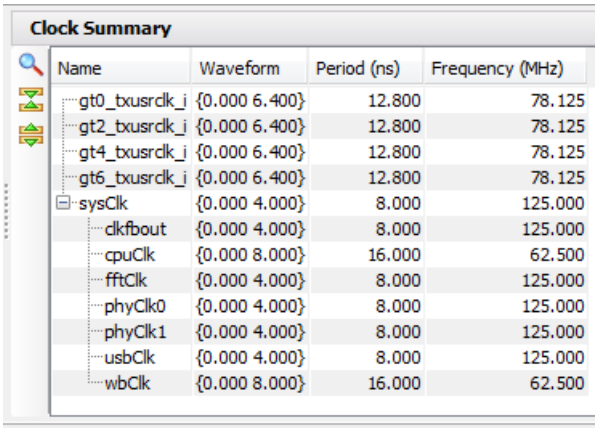
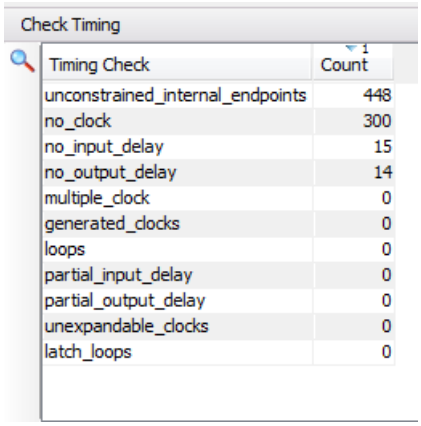


Figure 1-19: **Timing Summary Report: Clock Summary**

Check Timing Section

The Check Timing section of the Timing Summary Report contains information about missing timing constraints or paths with constraints issues that need to be reviewed. For complete timing signoff, all path endpoints must be constrained.


Check Timing

Timing Check	Count
unconstrained_internal_endpoints	448
no_clock	300
no_input_delay	15
no_output_delay	14
multiple_clock	0
generated_clocks	0
loops	0
partial_input_delay	0
partial_output_delay	0
unexpandable_clocks	0
latch_loops	0

Figure 1-20: **Timing Summary Report: Check Timing Section**

To generate Check Timing as a standalone report:

- Run the **Tools > Timing > Check Timing** menu command, or
- Run the Tcl `check_timing` command.

The list of checks reported by default, as shown in Figure 1-20, Timing Summary Report: Timer Settings, is:

- **no_input_delay**

  Number of input ports without at least one input delay constraint.

- **no_output_delay**

  Number of output ports without at least one output delay constraint.

- **unconstrained_internal_endpoints**

  Number of path endpoints excluding output ports without a timing requirement. This number is directly related to missing clock definitions, which is also reported by the no_clock check.

- **no_clock**

  Number of clock pins not reached by a defined timing clock. Constant clock pins are also reported.

- **multiple_clock**

  Number of clock pins reached by more than one timing clock. This usually happens when there is a clock multiplexer in one of the clock trees.

- **generated_clocks**

  Number of generated clocks that refer to a master clock source which is not part of the same clock tree.

- **loops**

  Number of combinational loops found in the design. The loops are automatically broken by the Vivado IDE timing engine to report timing.

- **partial_input_delay**

  Number of input ports with only a min input delay or max input delay constraint. These ports are not reported by both setup and hold analysis.

- **partial_output_delay**

  Number of output ports with only a min output delay or max output delay constraint. These ports are not reported by both setup and hold analysis.

- **unexpandable_clocks**

  Clock pairs for which the Vivado IDE timing engine could not find a common period multiplier over 1000 clock cycles. The paths between these clock pairs cannot be safely timed and the clock pairs must be treated as asynchronous.

- **latch_loops**

  Checks for and warns of sequential feedback loops in the design.

For more information on constraints definition, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

Intra-Clock Paths Section

This section describes the Timing values for the same source and destination clocks. The Intra-Clock Paths section of the Timing Summary Report summarizes the worst slack and

total violations of the timing paths. See Figure 1-21.

Intra-Clock Paths

Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
gt0_txusrclk_i	rise - rise	9.003	0.000	0	222	rise - rise	0.118	0.000	0	222	6.000	0.000	0	147
gt2_txusrclk_i	rise - rise	9.541	0.000	0	222	rise - rise	0.072	0.000	0	222	6.000	0.000	0	147
gt4_txusrclk_i	rise - rise	9.518	0.000	0	222	rise - rise	0.120	0.000	0	222	6.000	0.000	0	147
gt6_txusrclk_i	rise - rise	9.076	0.000	0	222	rise - rise	0.123	0.000	0	222	6.000	0.000	0	147
sysClk											3.000	0.000	0	9
clkfbout											8.592	0.000	0	2
cpuClk	rise - rise	5.952	0.000	0	5737	rise - rise	0.056	0.000	0	5737	9.600	0.000	0	3304
fftClk	rise - rise	4.719	0.000	0	8385	rise - rise	0.051	0.000	0	8385	4.358	0.000	0	1470
phyClk0	rise - rise	2.667	0.000	0	6088	rise - rise	0.083	0.000	0	6088	4.600	0.000	0	3795
phyClk1	rise - rise	2.456	0.000	0	6088	rise - rise	0.052	0.000	0	6088	4.600	0.000	0	3795
usbClk	rise - rise	1.416	0.000	0	2546	rise - rise	0.086	0.000	0	2546	4.600	0.000	0	1474

Figure 1-21: **Timing Summary Report: Intra-Clock Paths Section**

To view detailed information, click the names under Intra-Clock Paths in the left index pane. For example, you can view the slack and violations summary for each clock and details about the N-worst paths for SETUP/HOLD/Pulse Width checks.

The worst slack value and the number of reported paths are displayed next to the label for each analysis type. See Figure 1-22.

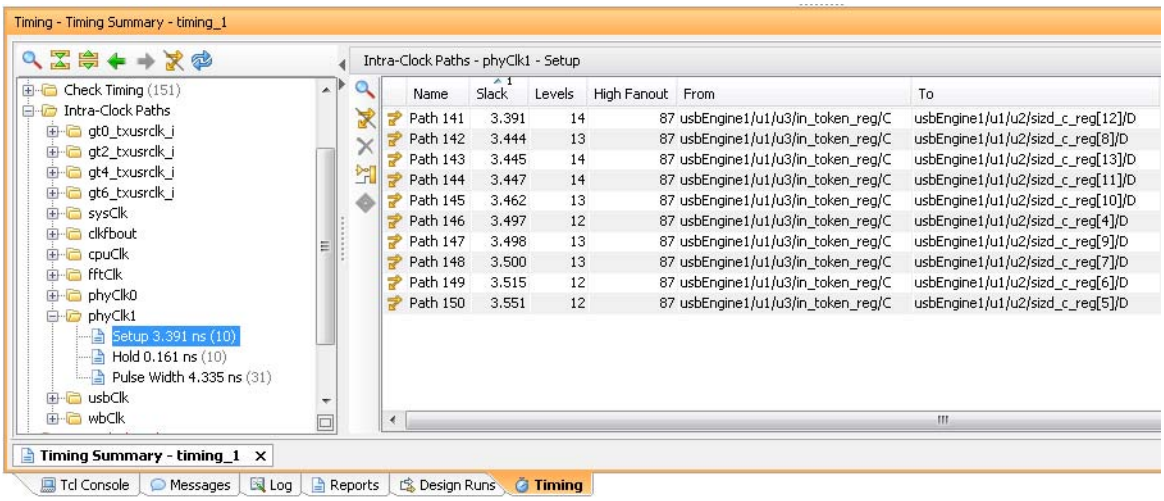


Figure 1-22: **Timing Summary Report: Intra-Clock Paths Details**

Inter-Clock Paths Section

Similar to the Intra-Clock Paths section, this section describes the Timing values for the different source and destination clocks. The Inter-Clock Paths section of the Timing Summary Report summarizes the worst slack and total violations of the timing paths with different source and destination clocks. See Figure 1-23.

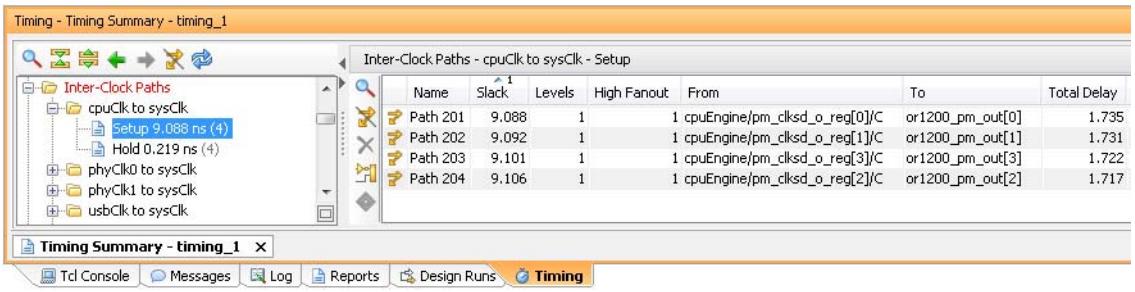


Figure 1-23: **Timing Summary Report Inter-Clock Paths Details**

To view detailed information, click the names under Inter-Clock Paths in the left index pane. For example, you can view the slack and violations summary for each clock and details about the N-worst paths for SETUP/HOLD/Pulse Width checks.

Path Groups Section

The Path Groups section of the Timing Summary Report displays default path groups and user-defined path groups.

Figure 1-24 shows an example of the Path Groups summary table. To access this table, select **Path Groups** in the left pane.

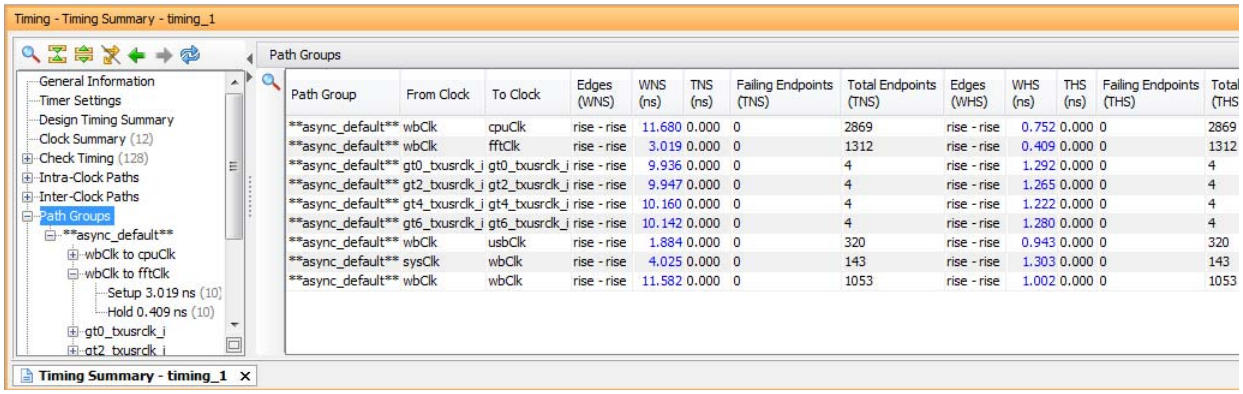


Figure 1-24: **Timing Summary Report: Path Groups Section**

TIP: *`**async_default**` is a path group automatically created by the Vivado IDE timing engine. It includes all paths ending with an asynchronous timing check, such as recovery and removal. These two checks are respectively reported under SETUP and HOLD categories, which corresponds to max delay analysis and min delay analysis. Any groups you create using `group_paths` appear in this section as well. Any combination of source and destination clocks can be present in a path group.*

User-Ignored Paths Section

The User-Ignored Paths Section of the Timing Summary Report displays the paths that are ignored during timing analysis due to the `set_clock_groups` and `set_false_path` constraints. The reported slack is infinite.

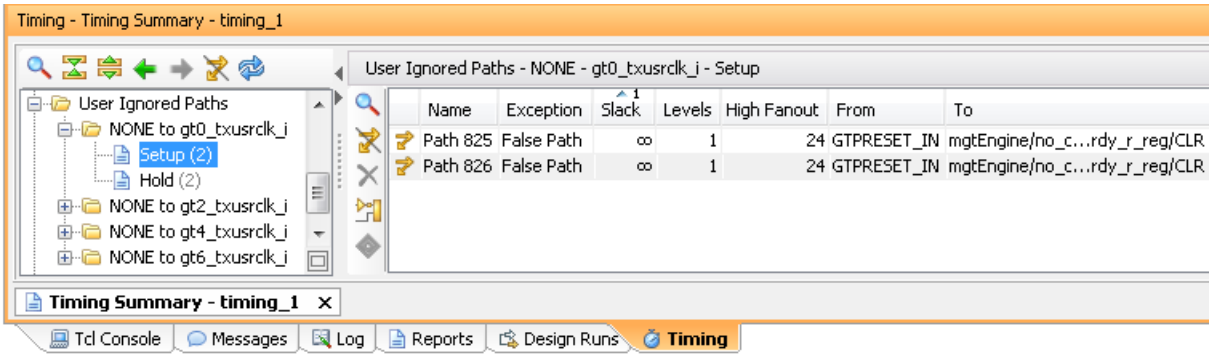


Figure 1-25: **Timing Summary Report: User-Ignored Paths Section**

Unconstrained Paths Section

The Unconstrained Paths section of the Timing Summary Report displays the logical paths that are not timed due to missing timing constraints. These paths are grouped by source and destination clock pairs. The clock name information displays as empty (or NONE) when no clock can be associated with the path startpoint or endpoint.

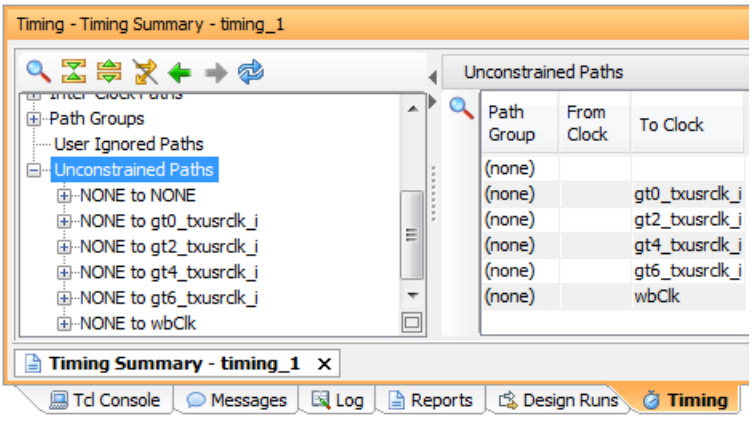


Figure 1-26: **Timing Summary Report: Unconstrained Paths Section**

Reviewing Timing Path Details

You can expand most of the sections to show paths organized by clock pairs. For each SETUP, HOLD and Pulse Width sub-section, you can view the N-worst reported paths. Select any of these paths to view more details in the Path Properties window (Report tab).

To view the same details in a new window, double click the path.

For more information on timing path details, see Chapter 3, Performing Timing Analysis.

To access more analysis views for each path:

1. Right click the path in the right pane.
2. Select one of the following options from the popup menu:
   - Open a Schematic of the path.
   - Rerun timing analysis on this same path.
   - Highlight the path in the Device and Schematic windows.

Filtering Paths With Violations

The report displays the slack value of failing paths in red. To focus on these violations, click the **Show only failing paths** button.

Figure 1-27 shows the Timing Summary window with only failing paths displayed.

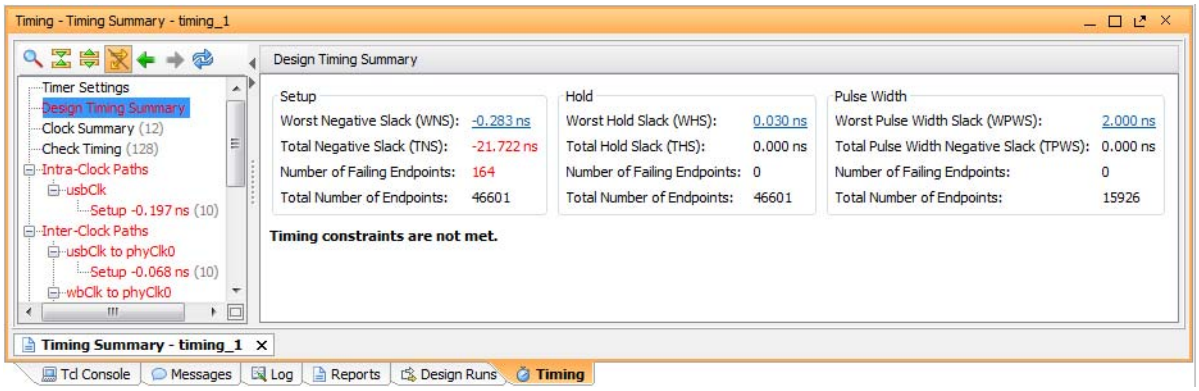


Figure 1-27: **Timing Summary Report Violating Paths Filter**

Report Clock Networks

The Report Clock Network command can be run from:

- The Flow Navigator in the Vivado IDE, or
- The Tcl command:

```
report_clock_networks -name {network_1}
```

Report Clock Networks provides a tree view of the clock trees in the design. See Figure 1-28, Clock Networks, page 37. Each clock tree shows the clock network from source to endpoint with the endpoints sorted by type.

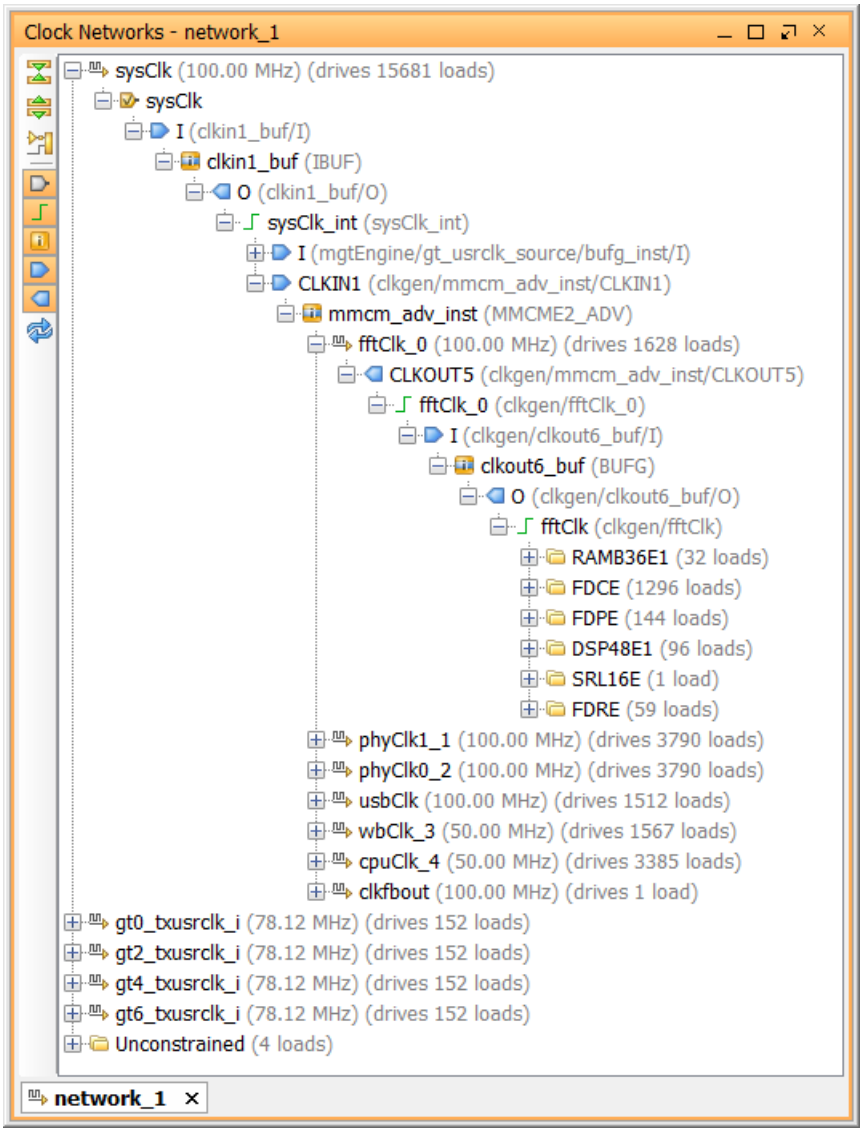


Figure 1-28: **Clock Networks**

The clock trees:

- Show clocks defined by the user or generated automatically by the tool.
- Report clocks from I/O port to load.

  Note: The full clock tree is only detailed in the GUI form of the report. The text version of this report shows only the name of the clock roots.
- Can be used to find BUFGs driving other BUFGs.
- Shows clocks driving non-clock loads.

There is a folder containing each primary clock and any generated clocks defined in the design. A separate folder displays each unconstrained clock root.

Use the Filter Ports, Nets, Instances, and related buttons to reduce the amount of data displayed in the clock tree.

To view a schematic of the clock path:

1. Select an object in the tree.
2. Run the **Trace to Source** popup command.

Report Clock Interaction

To view the Clock Interaction Report, run:

- **Main Menu > Tools > Timing > Report Clock Interaction**, or
- **Flow Navigator > Synthesis > Report Clock Interaction**, or
- **Flow Navigator > Implementation > Report Clock Interaction**

Equivalent Tcl command: `report_clock_interaction -name clocks_1`

Report Clock Interaction Dialog Box

In the Vivado IDE, the Report Clock Interaction dialog box includes the following:

- Results Name Field
- Command Field
- Open in a New Tab Checkbox
- Options Tab
- Timer Settings Tab

Results Name Field

The Results name field at the top of the Report Clock Interaction dialog box specifies the name of the graphical report that opens.

Equivalent Tcl option: `-name`

Command Field

Use the Command field to display the Tcl command line equivalent of the various options specified in the Report Datasheet dialog box.

Open in a New Tab Checkbox

Use the Open in a New Tab checkbox to either: (1) open the results in a new tab; or (2) replace the last tab opened by the Results window.

Options Tab

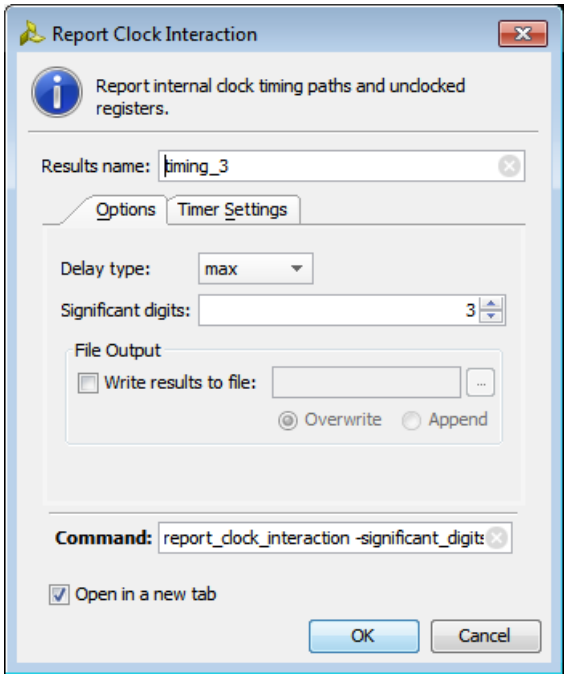


Figure 1-29: **Report Clock Interaction: Options Tab**

The Options tab of the Report Clock Interaction dialog box contains the following:

- Delay Type Field
- Significant Digits Field
- File Output Section
- Command Field
- Open in a New Tab Checkbox

Delay Type Field

Use the Delay Type field to set the type of analysis to be run.

- For *synthesized* designs, only max delay analysis (setup/recovery) is performed by default.
- For *implemented* designs, both min delay and max delay analysis (setup/hold, recover/removal) are performed by default.

To run min delay analysis only (hold and removal), select delay type `min`.

Equivalent Tcl option: `-delay_type`

Significant Digits Field

Use the Significant Digits field to specify the number of significant digits in the reported values. The default is three.

Equivalent Tcl option: `-significant_digits`

File Output Section

The File Output section includes:

- **Write Results to File Field**

  Use the Write Results to File field to write the result to a specified file. In the Vivado IDE, the report is displayed in the Clock Interaction window.

  Equivalent Tcl option: `-file`

- **Overwrite / Append Option Buttons**

  Select the Overwrite / Append option buttons to determine whether, when the report is written to a file: (1) the specified file is overwritten, or (2) new information is appended to an existing report.

  Equivalent Tcl option: `-append`

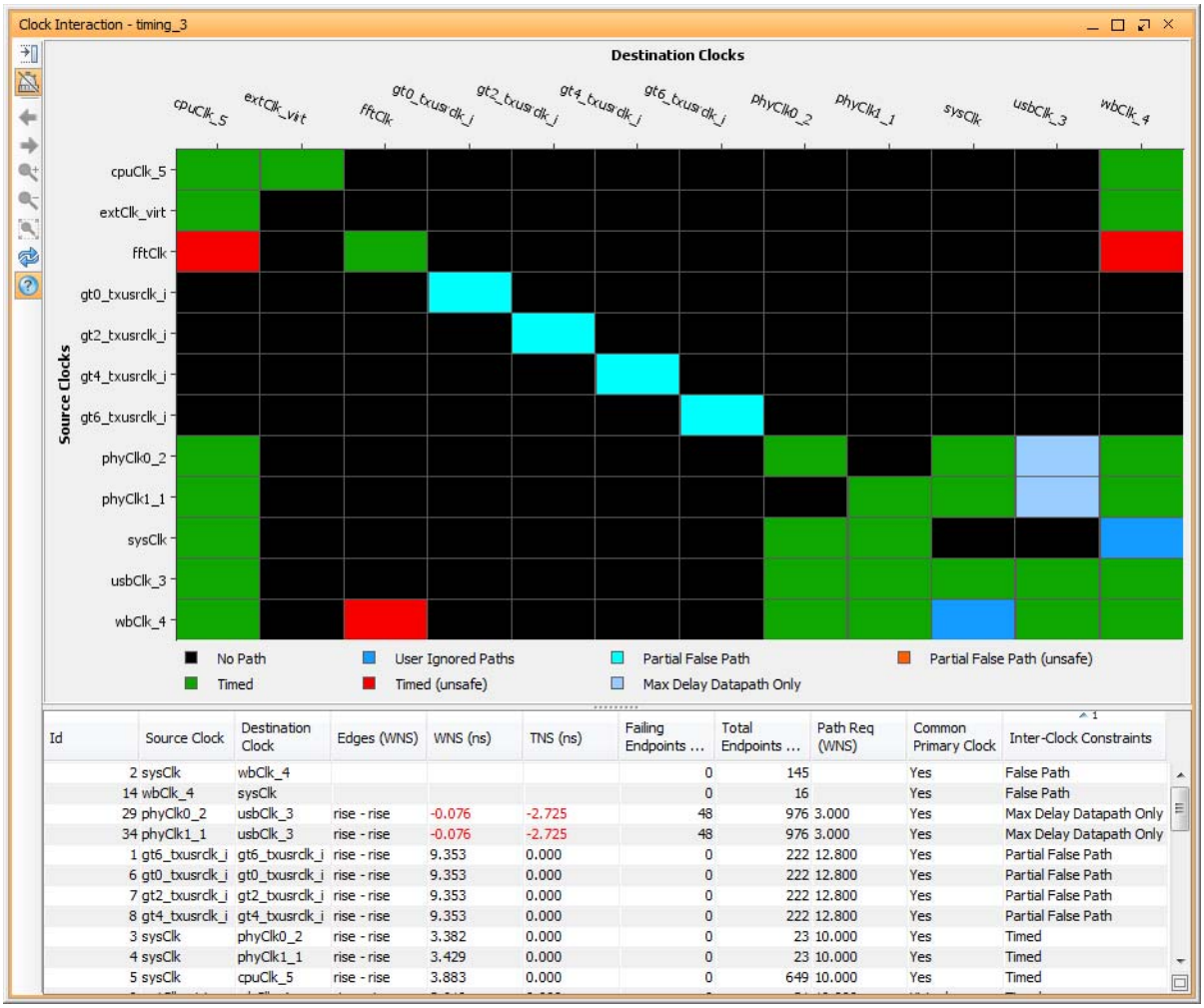
Timer Settings Tab

For details on this tab, see Timer Settings Tab, page 24.

Details of the Clock Interaction Report

The Clock Interaction report analyzes timing paths that cross from one clock domain (the source clock) into another clock domain (the destination clock). The Clock Interaction report helps to identify cases in which there may be data loss or metastability issues

After you run the Report Clock Interaction command, the results open in the Clock Interaction window. The Clock Interaction Report displays as a matrix of clock domains with the source clocks in the vertical axis and the destination clocks in the horizontal axis. See Figure 1-30, Report Clock Interaction.



Id	Source Clock	Destination Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints ...	Total Endpoints ...	Path Req (WNS)	Common Primary Clock	Inter-Clock Constraints
2	sysClk	wbClk_4				0	145		Yes	False Path
14	wbClk_4	sysClk				0	16		Yes	False Path
29	phyClk0_2	usbClk_3	rise - rise	-0.076	-2.725	48	976	3.000	Yes	Max Delay Datapath Only
34	phyClk1_1	usbClk_3	rise - rise	-0.076	-2.725	48	976	3.000	Yes	Max Delay Datapath Only
1	gt6_txusrclk_i	gt6_txusrclk_i	rise - rise	9.353	0.000	0	222	12.800	Yes	Partial False Path
6	gt0_txusrclk_i	gt0_txusrclk_i	rise - rise	9.353	0.000	0	222	12.800	Yes	Partial False Path
7	gt2_txusrclk_i	gt2_txusrclk_i	rise - rise	9.353	0.000	0	222	12.800	Yes	Partial False Path
8	gt4_txusrclk_i	gt4_txusrclk_i	rise - rise	9.353	0.000	0	222	12.800	Yes	Partial False Path
3	sysClk	phyClk0_2	rise - rise	3.382	0.000	0	23	10.000	Yes	Timed
4	sysClk	phyClk1_1	rise - rise	3.429	0.000	0	23	10.000	Yes	Timed
5	sysClk	cpuClk_5	rise - rise	3.883	0.000	0	649	10.000	Yes	Timed

Figure 1-30: **Report Clock Interaction**

Matrix Color Coding

The tiles of the matrix are color coded as shown below. The colors of the matrix are determined by the background color of the Graphical Editors as defined under **Tools > Options**. For more information, see the *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893) [Ref 1]. To hide the legend, click the **?** button on the toolbar on the left of the matrix.

- **No Path - Black**

  There are no timing paths that cross from the source clock to the destination clock. In this case, there is no clock interaction and nothing to report.

- **Timed - Green**

  The source clock and destination clock have a synchronous relationship, and are safely timed together. This state is determined by the timing engine when the two clocks have a common primary clock and a simple period ratio.

- **User Ignored Paths - Dark Blue**

  User-defined `false_path` or clock group constraints cover *all* paths crossing from the source clock to the destination clock.

- **Partial False Path - Light Blue**

  User-defined `false_path` constraints cover *some* of the timing paths crossing from the source clock to the destination clock where the source clock and destination clock have a synchronous relationship.

- **Timed (Unsafe) - Red**

  The source clock and destination clock have an asynchronous relationship. In this case, there is no common primary clock or no expandable period. For more information on asynchronous and unexpandable clocks, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

- **Partial False Path (Unsafe) - Orange**

  This category is identical to **Timed (Unsafe)**, except that at least one path from the source clock to the destination clock is ignored due to a false path exception.

- **Max Delay Datapath Only - Gray**

  A **set_max_delay -datapath_only** constraint covers all paths crossing from the source clock to the destination clock.

IMPORTANT: *The color of a cell in the matrix reflects the state of the constraints between clock domains, not the state of the timing paths worst slack between the domains. A green cell does not indicate that the timing is met, only that all timing paths that cross clock domains are properly timed, and that their clocks have a known phase relationship.*

Filtering the Clocks

To filter the clocks displayed in the Clock Interaction report:

1. Select the **Clock Interaction View Layers** command in the toolbar.
2. Select the clocks to display.

This command reduces the matrix complexity by limiting the number of clocks, but does not reduce the number of clock interactions reported in the table below the matrix. See Figure 1-31, Clock Interaction View Layers. You can also show and hide the clocks that do not directly time a logical path in the design by clicking the **Hide Unused Clocks** button in the toolbar. Because these clocks do not contribute to WNS/TNS/WHS/THS computation, they are hidden by default.

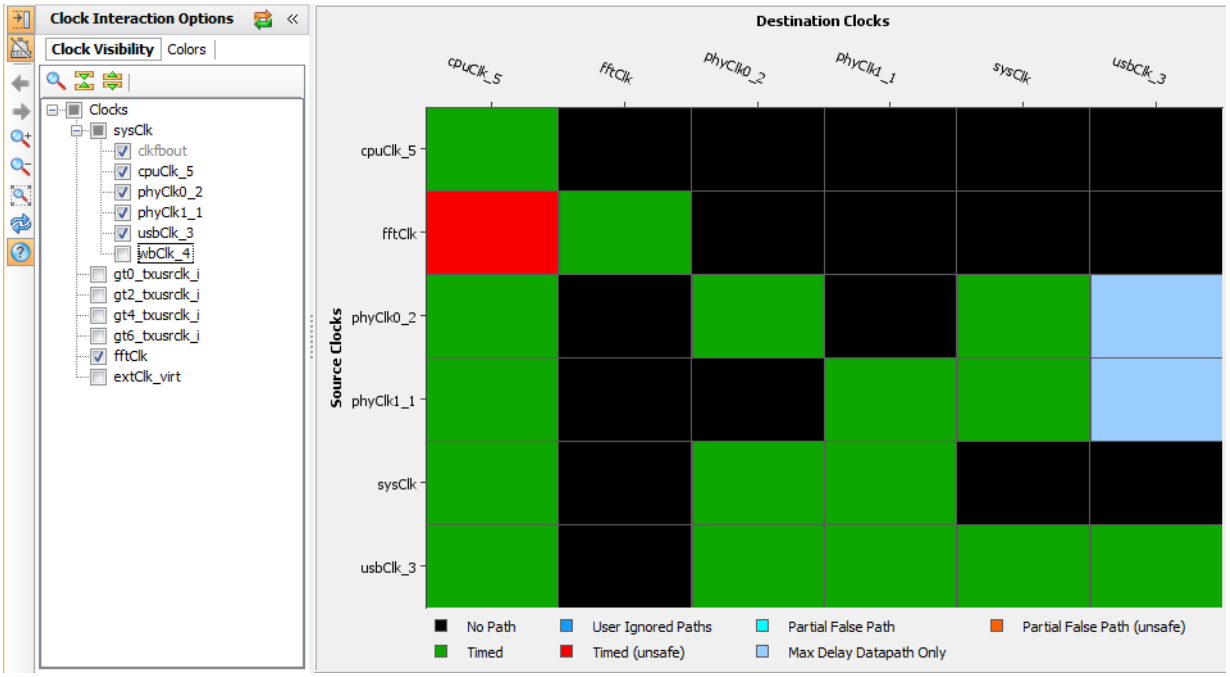


Figure 1-31: **Clock Interaction View Layers**

Clock Pairs Slack Table

The table below the matrix provides a comprehensive overview of the timing slack for setup/recovery and/or for hold/removal for source/destination clock pair. It also shows useful information about path requirement of the worst paths, common primary clock and constraints status. See Figure 1-30, Report Clock Interaction, page 42. This provides details not displayed in the matrix above.

Sorting the Data

To sort the data in the table in increasing or decreasing values, single click multiple times on a column header.

Selecting Cells and Rows

Selecting a cell in the matrix cross-selects a specific row of the table below.

Selecting a row from the table highlights a cell in the matrix above.

Table Columns

The table columns are:

- **ID**

  A numeric ID for the source/destination clock pair being displayed.

- **Source Clock**

  The clock domain from which the path originates.

- **Destination Clock**

  The clock domain within which the path terminates.

- **Edges (WNS)**

  The clock edges used to calculate the worst negative slack for max delay analysis (setup/recovery).

- **WNS (Worst Negative Slack)**

  The worst slack calculated for various paths crossing the specified clock domains. A negative slack indicates a problem in which the path violates a required setup (or recovery) time.

- **TNS (Total Negative Slack)**

  The sum of the worst slack violation for all the endpoints that belong to paths crossing the specified clock domains.

- **Failing Endpoints (TNS)**

  The number of endpoints in the crossing paths that fail to meet timing. The sum of the violations corresponds to TNS.

- **Total Endpoints (TNS)**

  The total number of endpoints in the crossing paths.

- **Path Requirements (WNS)**

  The timing path requirement corresponding to the path reported in the WNS column. There can be several path requirements between any clock pairs if both rising and falling edges are active for at least one of the two clocks, or some timing exceptions have been applied on paths between the two clocks. The value reported in this column is not always the most challenging requirement.

  For more information, see "Basics of Timing Checks" in the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

- **Common Primary Clock**

  Whether the source clocks and destination clocks of the timing path are defined by a common primary clock. If either the source clock or the destination clock of the timing paths is with respect to a virtual clock, **Virtual** is displayed in the Common Primary Clock field.

- **Inter-Clock Constraints**

  Shows the constraints summary for all paths between the source clock and destination clock. The possible values are listed in the Matrix Color Coding section above. Following are example definitions of these constraints:

```
set_clock_groups -async -group wbClk -group usbClk
set_false_path -from [get_clocks wbClk] -to [get_clocks cpuClk]
```

When the min delay analysis is also selected (hold/removal), the following columns also appear in the table:

- **Edges (WHS)**

  The clock edges used to calculate the worst hold slack.

- **WHS (Worst Hold Slack)**

  The worst slack calculated for various paths crossing the specified clock domains. A negative slack indicates a problem in which the path violates a required hold (or removal) time.

- **THS (Total negative Hold Slack)**

  The sum of the worst slack violation for all the endpoints that belong to paths crossing the specified clock domains for min delay analysis (hold/removal).

- **Failing Endpoints (THS)**

  The number of endpoints in the crossing paths that fail to meet timing. The sum of the violations corresponds to THS.

- **Total Endpoints (THS)**

  The total number of endpoints in the crossing paths for min delay analysis (hold/removal).

- **Path Requirements (WHS)**

  The timing path requirement corresponding to the path reported in the WHS column. Like with WNS, there can be several possible path requirements for min delay analysis between two clocks, and the value reported in this column does not always correspond to the most challenging ones.

  For more information, see "Basics of Timing Checks" in the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

One or multiple clock pairs can be selected from the table. **Report Timing** between a selected source/destination clock pair can be run from the popup menu.

Exporting the Table

Run the **Export to Spreadsheet** command to export the table to an XLS file for use in a spreadsheet.

Report Pulse Width

The Pulse Width Report checks that the design meets min period, max period, high pulse time, and low pulse time requirements for each instance clock pin. It also checks that the maximum skew requirement is met between two clock pins of a same instance in the implemented design (for example, PCIe clocks).

Equivalent Tcl command: `report_pulse_width`

Note: Xilinx Integrated Software Environment (ISE®) Design Suite implementation calls this check Component Switching Limits.

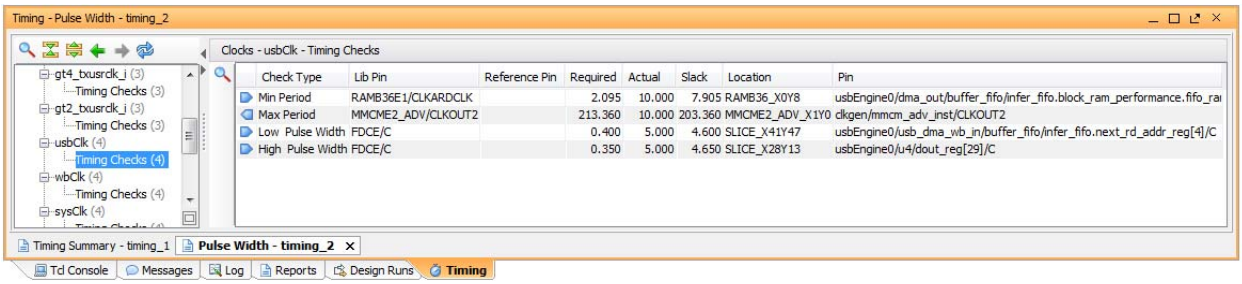


Figure 1-32: **Report Pulse Width**

Report Timing

Read Report Timing to view specific timing paths at any point of the flow after synthesis when: (1) you need to further investigate timing problems reported by Report Timing Summary; or (2) you want to report the validity and the coverage of particular timing constraints. Report Timing does not cover Pulse Width reports.

Running Report Timing

If a design is already loaded in memory, you can run Report Timing from the:

- Menu
- Clock Iteration Report
- Timing Report and Timing Report Summary Paths List

Running Report Timing From the Menu

To run Report Timing from the Menu:

1. Select **Tools > Timing > Report Timing**.

Running Report Timing From the Clock Iteration Report

To run Report Timing from the Clock Iteration Report:

1. Select a **from/to** clock pair.
2. Right click.
3. Select **Report Timing** to run a report from or to the selected clocks.

Running Report Timing From a Paths List

To run Report Timing from a Paths List:

1. Select a path.
2. Right click.
3. Select **Report Timing** to run a report between the selected path startpoint endpoint.

Equivalent Tcl command: `report_timing`

When setting specific Report Timing options, you can view the equivalent `report_timing` command syntax in:

- The Command field at the bottom of the dialog box, and
- The Tcl Console after execution

The `report_timing` options are listed along with the dialog box description in the following section.

Overall, the Report Timing options are identical to the Report Timing Summary options, plus a few additional filtering options.

Targets Tab

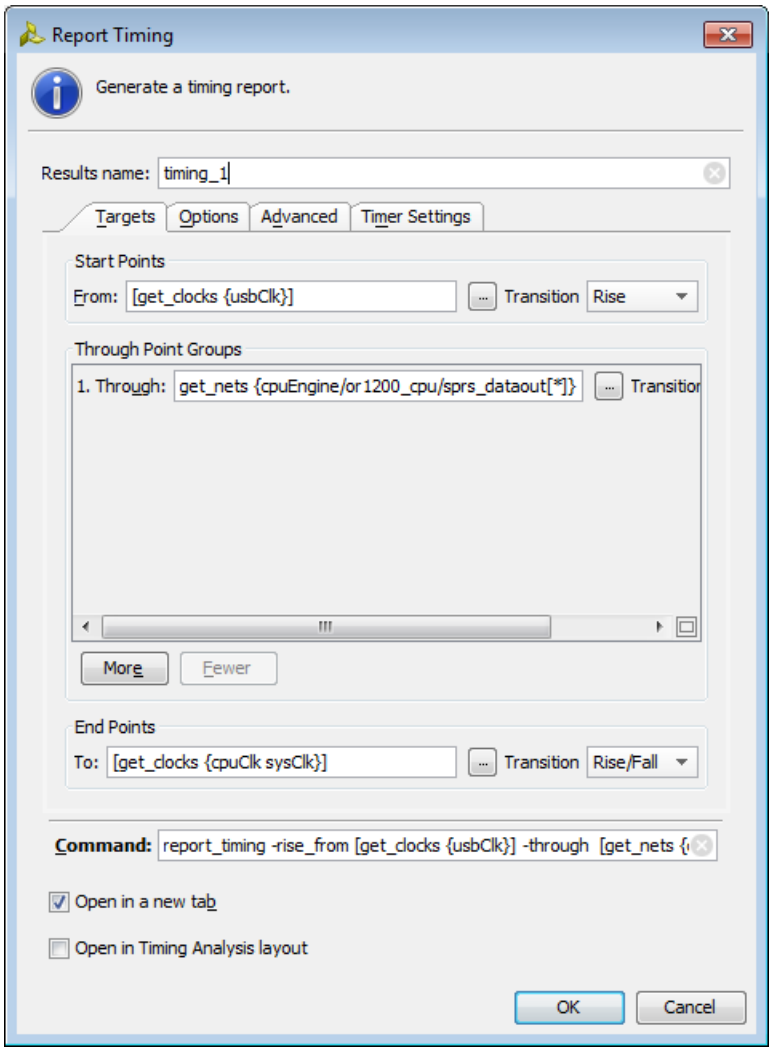


Figure 1-33: **Report Timing Dialog Box: Targets Tab**

Report Timing provides several filtering options that you must use in order to report a particular path or group of paths. The filters are based on the structure of a timing path.

- **Startpoints (From)**

  List of startpoints, such as sequential cell clock pins, sequential cells, input ports, bidirectional ports or source clock.

  If you combine several startpoints in a list, the reported paths will start from any of these netlist objects.

  The Rise/Fall filter selects a particular source clock edge.

  Equivalent Tcl option: `-from`, `-rise_from`, `-fall_from`

- **Through Point Groups (Through)**

  List of pins, ports, combinational cells or nets.

  You can combine several netlist objects in one list if you want to filter on paths that traverse any of them.

  You can also specify several Through options to refine your filters and report paths that traverse all groups of through points in the same order as they are listed in the command options.

  The Rise/Fall filter applies to the data edge.

RECOMMENDED: *Use the default value (Rise/Fall).*

  Equivalent Tcl option: `-through`, `-rise_through`, `-fall_through`

- **Endpoints (To)**

  List of endpoints, such as input data pins of sequential cells, sequential cells, output ports, bidirectional ports or destination clock.

  If you combine several endpoints in a list, the reported paths will end with any of these netlist objects.

  In general, the Rise/Fall option selects a particular data edge. But if you specified a destination clock, it selects a particular clock edge.

  Equivalent Tcl option: `-to`, `-rise_to`, `-fall_to`

Figure 1-33, page 50, defines the paths from the rising clock edge of `usbClk`, through any of the `u4cpuEngine/or1200_cpu/sprs_dataout[*]` nets, to either edge of `cpuClk` or `sysClk`.

Options Tab

The Options tab contains the following options:

- Reports
- Path Limits
- Path Display

Reports

- **Path delay type**

  See Report Timing Summary, page 19.

- **Do not report unconstrained paths**

  By default, Report Timing reports paths that are not constrained if no path that matches the filters (from/through/to), is constrained. Check this box if you do not want to display unconstrained paths in your report.

  Equivalent Tcl option: `-no_report_unconstrained`

Path Limits

- **Number of paths per group**

  See Report Timing Summary, page 19.

- **Number of paths per endpoint**

  See Report Timing Summary, page 19.

- **Limit paths to group**

  Filters on one or more timing path groups. Each clock is associated to a group. The Vivado IDE timing engine also creates default groups such as **`async_default`** which groups all the paths ending with a recovery or removal timing check.

  Equivalent Tcl option: `-group`

Path Display

- **Display paths with slack greater than**

  Displays the reported paths based on their slack value.

  Equivalent Tcl option: `-slack_greater_than`

- **Display paths with slack less than**

  See Report Timing Summary, page 19.

- **Number of significant digits**

  See Report Timing Summary, page 19.

- **Sort paths by**

  Displays the reported paths by group (default) or by slack. When sorted by group, the N worst paths for each group and for each type of analysis (`-delay_type min/max/min_max`) are reported.

  The groups are sorted based on their individual worst path. The group with the worst violation appears at the top of the list.

  When sorted by slack, the N worst paths per type of analysis are reported (all groups combined) and are sorted by increasing slack.

  Equivalent Tcl option: `-sort_by`

Advanced Tab

Same options as Report Timing Summary, page 19.

Timer Settings Tab

Same options as Report Timing Summary, page 19.

Reviewing Timing Path Details

After you click **OK** to run the report command, a new window opens. You can now review its content. You can view the N-worst paths reported for each type of selected analysis (`min/max/min_max`).

Figure 1-34 shows the Report Timing window in which both min and max analysis (SETUP and HOLD) were selected, and N=4.

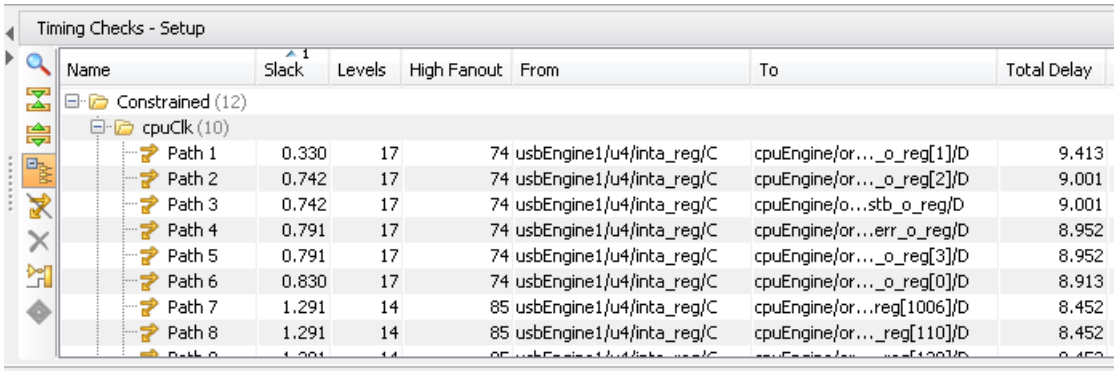


Figure 1-34: **Report Timing Paths List**

Select any of these paths to view more details in the Path Properties window (Report tab).

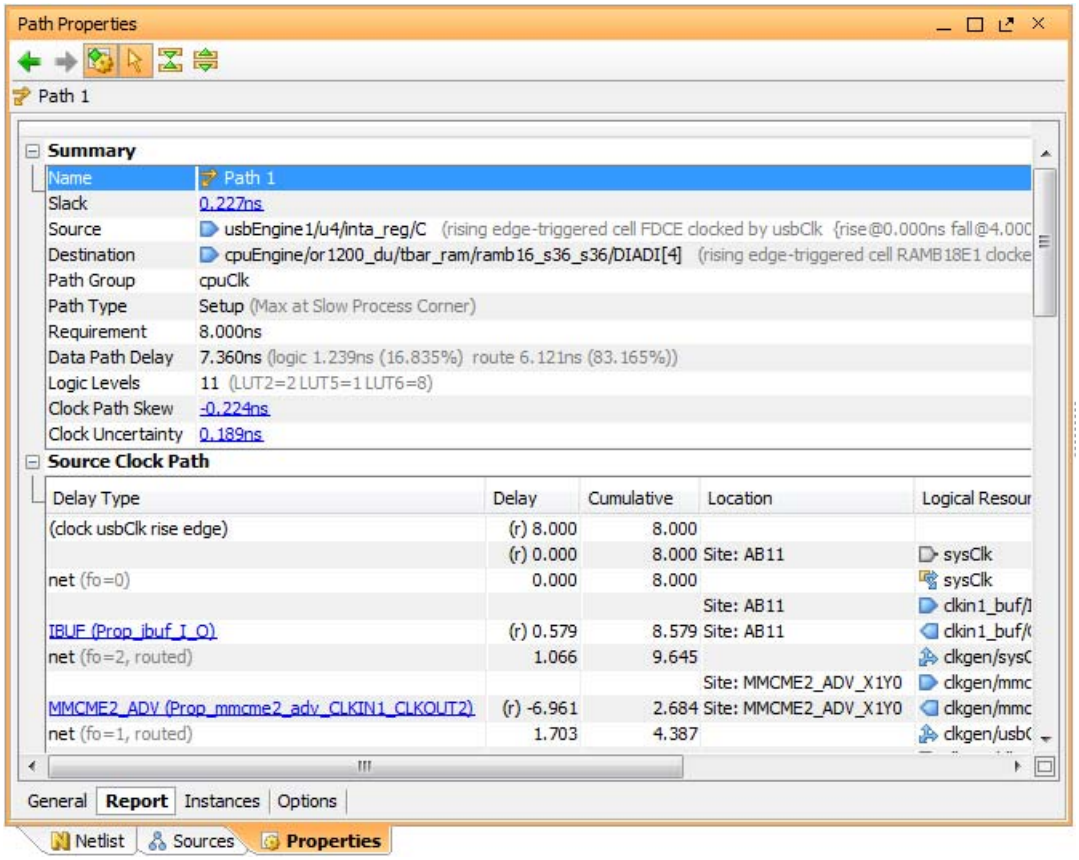


Figure 1-35: **Timing Path Properties Window**

To view the same details in a new window, double click the path.

For more information on timing path details, see Chapter 3, Performing Timing Analysis.

To access more analysis views for each path:

1. Right click the path in the right pane.
2. Select one of the following actions from the popup menu:
   - View the timing path Schematic.
   - Rerun timing analysis on the same startpoint and endpoint of the selected path.
   - Highlight the path in the Device and Schematic windows.

Filtering Paths With Violation

The report displays the slack value of failing paths in red. To focus on these violations, click **Show only failing checks mode**.

Report Datasheet

The Report Datasheet command reports the operating parameters of the FPGA device for use in system-level integration.

Report Datasheet Dialog Box

In the Vivado IDE, select **Tool > Timing > Report Datasheet** to open the Report Datasheet dialog box. See Figure 1-36.

Report Datasheet Dialog Box: Options Tab

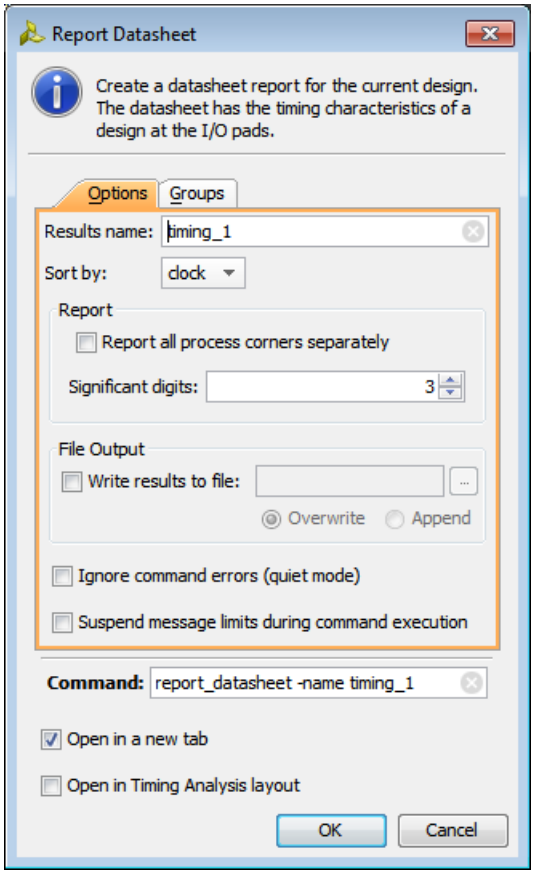


Figure 1-36: **Report Datasheet Dialog Box: Options Tab**

The Report Datasheet Dialog Box Options tab includes the following:

- **Results name**

  Specifies the name for the returned results of the Report Datasheet command. The report opens in the Timing window of the Vivado IDE with the specified name.

  Equivalent Tcl option: `-name`

- **Sort by**

  Sorts the results by port name or by clock name.

  Equivalent Tcl option: `-sort_by`

- **Report all process corners separately**

  Reports the data for all defined process corners in the current design.

  Equivalent Tcl option: `-show_all_corners`

- **Significant digits**

  Specifies the number of significant digits in the reported values. The default is three.

  Equivalent Tcl option: `-significant_digits`

- **Write results to file**

  Write the result to the specified file name. By default the report is written to the Results window in the Vivado IDE.

  Equivalent Tcl option: `-file`

- **Overwrite / Append**

  When the report is written to a file, determines whether the specified file is overwritten, or new information is appended to an existing report.

  Equivalent Tcl option: `-append`

- **Ignore command errors**

  Executes the command quietly, ignoring any command line errors and returning no messages.

  Returns TCL_OK regardless of any errors encountered during execution.

  Equivalent Tcl option: `-quiet`

- **Suspend message limits**

  Temporarily overrides any message limits.

  Returns all messages from this command.

  Equivalent Tcl option: `-verbose`

- **Command**

  Displays the Tcl command line equivalent of the various options specified in the Report Datasheet dialog box.

- **Open in a new tab**

  Opens the results in a new tab, or replaces the last tab opened by the Results window.

- **Open in Timing Analysis layout**

  Resets the current view layout to the Timing Analysis view layout.

Report Datasheet Dialog Box: Groups Tab

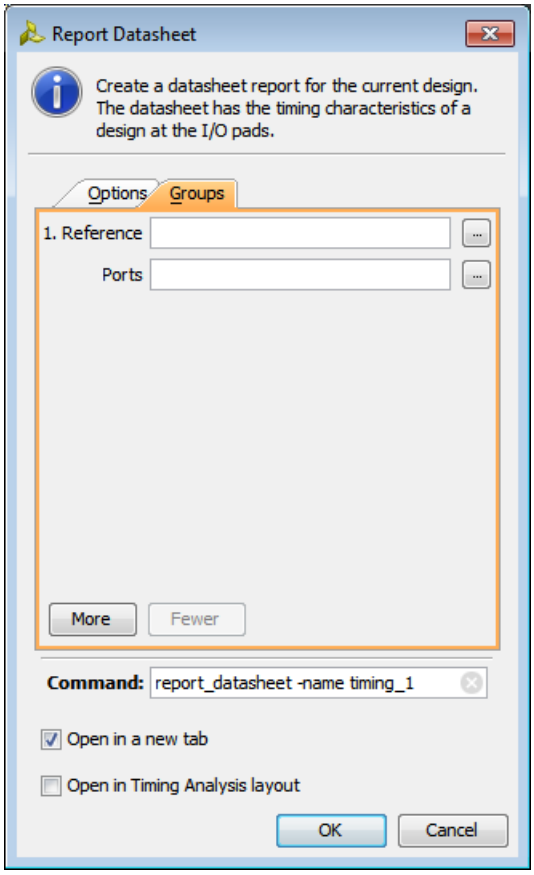


Figure 1-37: **Report Datasheet Dialog Box: Groups Tab**

The Report Datasheet dialog box Groups tab allows you to define your own custom group of ports for analysis by specifying the reference port and additional ports to report. When Groups are not specified, the timer automatically finds the group of output ports based on the launching clock, and reports skew based on that clock.

The Report Datasheet dialog box Groups tab includes:

- **Reference**

  Specifies the reference port for skew calculation. In most cases, this will be a clock port of a source synchronous output interface.

  Equivalent Tcl option: `-group`

- **Ports**

  Defines additional ports to report.

- **More**

  Specifies multiple groups, each with its own reference clock port. This field allows you to define a new group of ports, including a new reference port.

- **Fewer**

  Removes additional groups of ports as needed.

Details of the Datasheet Report

General Information

This section provides details of the design and Xilinx device, and the tool environment at the time of the report.

- **Design Name**

  The name of the design

- **Xilinx Part**

  The target Xilinx part

- **Speedfile**

  The path to the speedfile used for analysis

- **Vivado Version**

  The version of the Vivado tools used when the report was generated

- **Time generated**

  The date and timestamp of the report

- **Command line**

  The command line used to generate the report

Setup/Hold for Input Ports

The report displays worst-case setup and hold requirements for every input port with regard to the reference clock. The internal clock used to capture the input data is also reported.

Max/Min Delays for Output Ports

Shows worst-case maximum and minimum delays for every output port with regard to the reference clock. The internal clock used to launch the output data for is also reported.

Setup Between Clocks

For every clock pair, the worst-case setup requirements are reported for all clock edge combinations.

Setup/Hold for Input Buses

Input buses are automatically inferred and their worst-case setup and hold requirements are displayed. Worst case data window for the entire bus is the sum of the largest setup and hold values. If the input ports are constrained, the slack is also reported.

An optimal tap point is reported for input clocks with IDELAY defined. The optimal tap point can be used to configure IDELAY for balanced setup and hold slack.

The source offset is the delta between two windows. The first window is defined by the setup and hold time of the input port with regard to the clock. The second window is derived from the input delay and the clock period. If the input clock is offset with this value, then it will be in the center of the window.

Figure 1-38 reports a design in which a DDR input bus, `dq[0-7]`, has a worst case data window of 8.150 ns. The ideal clock offset is 0.179 ns. The optimal tap point for IDELAY is 13. The optimal tap point can be specified by using the Tcl command:

```
set_property IDELAY_VALUE 13 [get_cells idelay_clk]
```

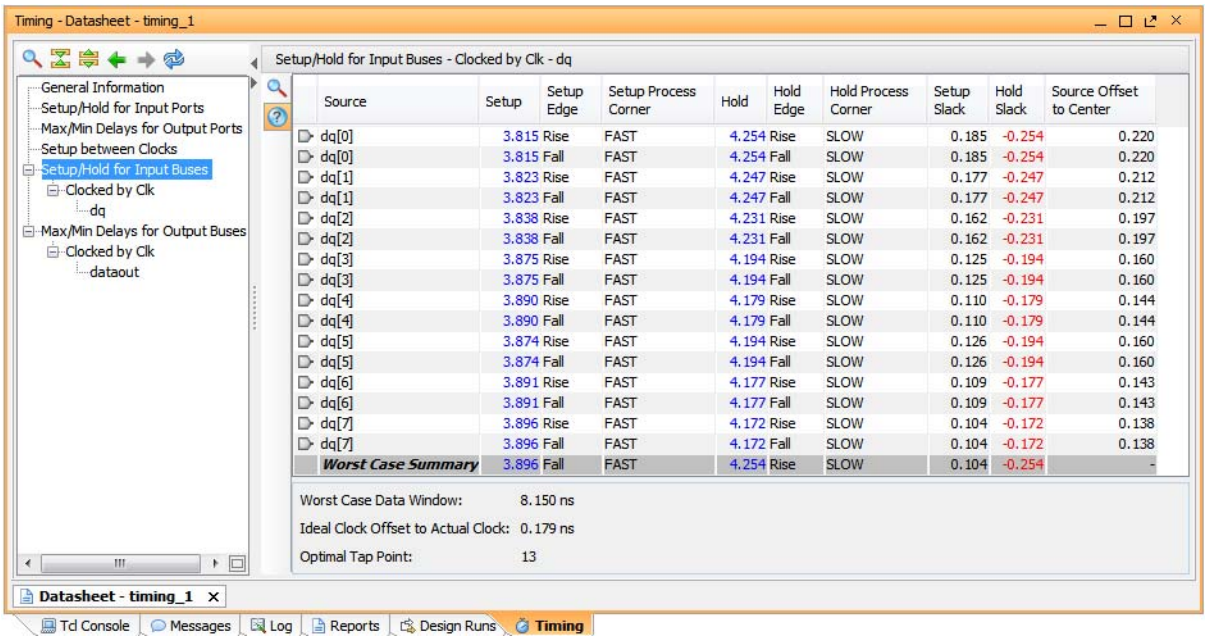


Figure 1-38: **Setup and Hold Delays for Input Buses**

Max/Min Delays for Output Buses

Output buses are automatically inferred and their worst case maximum and minimum delays are displayed. The bus skew is also reported. For bus skew calculation, one bit is considered as the reference and the offset of every other bit is calculated with respect to this reference bit. The worst offset is the skew for the entire bus.

Max/Min Delays for Groups

For DDR, the output skew is desired w.r.t. forwarded clock. A custom group report can be generated by specifying the reference port as the forwarded clock port. This table looks similar to "max/min delays for output buses" except the reference port is used as the reference bit for calculating source offset and bus skew.

As an example, for a DDR output skew calculation, if multiple bits (for example, `rldiii_a[0-19]`, `rldiii_ba[0-3]`, `rldiii_ref_n`, `rldiii_we_n`) should be grouped together with regard to the forwarded clock port (`rldiii_ck_n[0]`), the following command can be used:

```
report_datasheet -group [get_ports {rldiii_ck_n[0] rldiii_a[*] rldiii_ba[*]
rldiii_ref_n rldiii_we_n}]
```

The first port in the group list is considered the reference pin.

For all these sections, the worst case data is calculated from multi-corner analysis. If `-show_all_corners` is used, the worst case data is reported for each corner separately.

Report Exception

You can use the Report Exception command anywhere in the flow after the synthesis. The Report Exception command reports the following information:

- All the timing exceptions that have been set in the design and that are affecting timing analysis
- All the timing exceptions that have been set in the design but that are being ignored as they are overridden by other timing exception

The timing exceptions analyzed by the Report Exception command are (in the order of precedence):

- false paths
- max/min delays
- multicycle paths

The Report Exception is a powerful command to help debugging issues related to timing exceptions. Some designs have timing constraints with complex timing exceptions. Because the timing exceptions have different priorities, it can quickly become difficult to understand which timing exceptions might be partially or fully ignored by other exception(s). The Report Exception reports timing exceptions that are partially overridden, as well as those that are totally overridden. It also provides a hint to the overriding constraint(s).

Since Clock Groups constraints (`set_clock_groups`) are not exactly a timing exception, they are not covered in the summary table of all the timing exceptions affecting the design. However, when a timing exception is overriden by a clock group constraint, the clock group overriding the constraint is reported in the **Status** column of the table.

The Report Exception command is only available through the Tcl command line, using Tcl command `report_exception`.

For more information about the report_exception command line options, refer to the *Vivado Design Suite Tcl Command Reference Guide* (UG835). For more information about the timing exception priority order, refer to the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

The report_exception command has two modes of operation:

- Reporting the timing exceptions affecting the timing analysis
- Reporting the timing exceptions being ignored

Example: Reporting the Timing Exceptions Affecting the Timing Analysis

This example describes how to take the design shown in Figure 1-39 through some timing exceptions. The design is fully constrained (clock *clk* and input/output delays defined relative to *clk*).

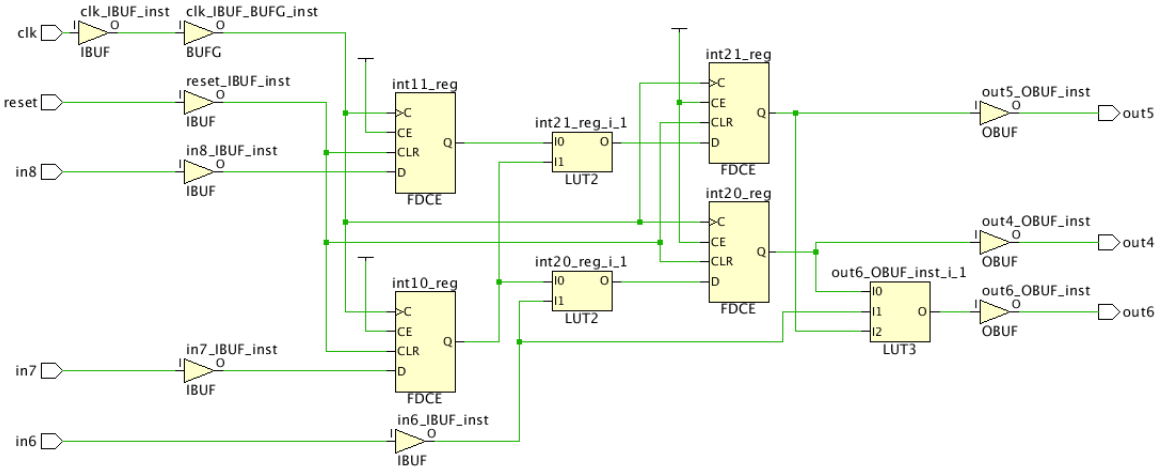


Figure 1-39: **Fully Constrained Design for Timing Exception Example**

The first mode of operation of the Report Exception command is `report_exception`.

1. Select **Window > Timing Constraints** and add the following timing exceptions to the design:

```
set_multicycle_path 3 -from [get_cell int10_reg] -to [get_cell int20_reg]
set_multicycle_path 4 -to [get_cell int20_reg]
set_false_path -from [get_ports in6] -to [get_cell int20_reg]
set_false_path -to [get_ports out5]
set_false_path -to [get_cell int21_reg]
set_false_path -from [get_ports in6] -to [get_ports out6]
set_max_delay 5 -to [get_ports out6]
set_min_delay 3 -from [get_cells int10_reg] -to [get_cell int20_reg]
```

The Constraints window displays the timing constraints applied to the design, as shown in Figure 1-40.

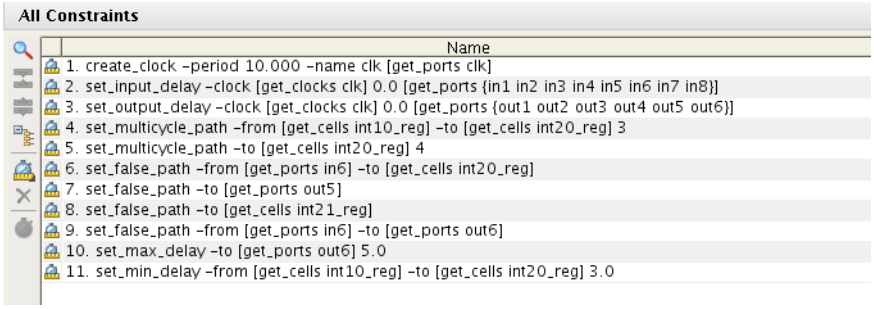


Figure 1-40: **Constraints Window Displaying Timing Constraint Changes**

The actual Exception Report (report_exception) is shown in Figure 1-41.

Exceptions Report

Position	From	Through	To	Setup	Hold	Status
4	[get_cells int10_reg]	*	[get_cells int20_reg]	cycles=3	-	
5	*	*	[get_cells int20_reg]	cycles=4	-	Partially overridden path by MCP 4 - FP 6
6	[get_ports in6]	*	[get_cells int20_reg]	false	false	
7	*	*	[get_ports out5]	false	false	
8	*	*	[get_cells int21_reg]	false	false	
9	[get_ports in6]	*	[get_ports out6]	false	false	
10	*	*	[get_ports out6]	max=5	-	Partially overridden path by FP 9
11	[get_cells int10_reg]	*	[get_cells int20_reg]	-	min=3	

Figure 1-41: **Report Exception**

The Exceptions Report contains the following information:

- The *Position* column indicates the constraint position number. This is the same position number reported by the Timing Constraint Window, shown in Figure 1-40.
- The *From/Through/To* columns indicate the patterns or objects specified with *-*from/-*through/-*to* command line options (including all the *rise/fall* versions of those options). An asterisk is displayed when the associated option was not specified.
- The *Setup/Hold* columns indicate whether the constraint applies to setup check, hold check, or both. The naming convention for the *Setup/Hold* columns is:

Short Name	Timing Exception
cycles=	set_multicycle_path
false	set_false_path
max=	set_max_delay
max_dpo=	set_max_delay -datapath_only
min=	set_min_delay

- The *Status* column reports a message when a constraint is partially overridden by another timing exception. The naming convention for the *Status* column is:

Short Name	Timing Exception
MCP	multicycle path
FP	false path
MXD	max delay
MND	min delay
CG	clock group

Note: The clock group is only reported in the **Status** column of the `report_timing -ignored` command when a clock group constraint overrides another timing exception.

In this example, there are two messages regarding partially overridden constraints:

- The timing constraint position 5 (`set_multicycle_path 4 -to [get_cell int20_reg]` based on the Timing Constraints Window) is partially overridden by the multicycle constraint position 4 (`set_multicycle_path 3 -from [get_cell int10_reg] -to [get_cell int20_reg]`) and by the false path constraint position 6 (`set_false_path -from [get_ports in6] -to [get_cell int20_reg]`).
- The timing constraint position 10 (`set_max_delay 5 -to [get_ports out6]`) is partially overridden by the false path position 9 (`set_false_path -from [get_ports in6] -to [get_ports out6]`).

Reporting the Timing Exceptions Being Ignored

The second mode of operation of the Report Exception command is `report_exception -ignored`

To illustrate, add the following timing exceptions on the top of the previous ones:

```
set_max_delay 5 -to [get_ports out5]
set_multicycle_path 1 -hold -to [get_cell int21_reg]
set_multicycle_path 2 -setup -to [get_ports out6]
set_false_path -from [get_cell int11_reg] -to [get_cell int20_reg]
```

All those exceptions are either already covered by a timing exception from the previous section (*Reporting the timing exceptions affecting the timing analysis*) or target a non-existing path (there is no physical connection between the registers *int11_reg* and *int20_reg*).

After adding these four constraints, the Timing Constraints Window looks like Figure 1-42.

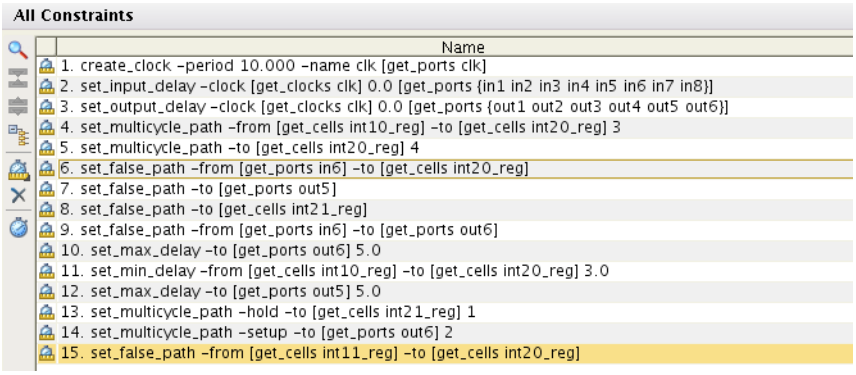


Figure 1-42: **Timing Constraints Window**

The Exceptions Report (report_exception -ignored) is as shown here:

Exceptions Report

Position	From	Through	To	Setup	Hold	Status
12	*	*	[get_ports out5]	max=5	-	Totally overridden path by FP 7
13	*	*	[get_cells int21_reg]	-	cycles=1	Totally overridden path by FP 8
14	*	*	[get_ports out6]	cycles=2	-	Totally overridden path by FP 9 - WXD 10
15	[get_cells int11_reg]	*	[get_cells int20_reg]	false	false	Non-existent path

Figure 1-43: **Exceptions Report**

Note: The *Status* column provides some explanations why the timing exceptions are being ignored.

Implementation Results Analysis Features

This section discusses techniques for reviewing a design after implementation to understand behavior inside the device, including:

- Reviewing placement for hierarchical blocks
- I/Os
- Looking at connectivity
- Cross probing between views
- Reviewing detailed routing

Using the Design Runs Window

The Design Runs window displays the state of the current runs.

For more information, see *Using the Design Runs Window* in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 7].

If the run is running, finished cleanly, or finished with errors, the Design Runs window appears when a run is done.

TIP: *If the run is not up to date, you can select* `Force Up-to-Date` *from the pop-up menu.*

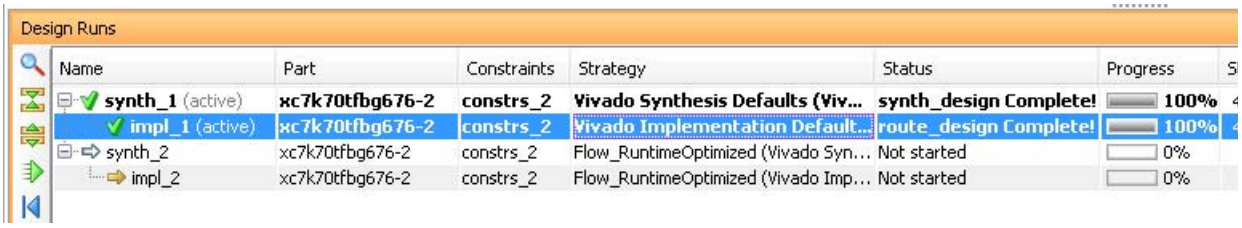


Figure 1-44: **Design Runs Window**

The Design Runs Window columns show:

- The name of the run
- The target part
- The constraints set associated with a run
- The run strategy
- The status of the last completed step of a run
- The progress of a run
- The start time of a run
- The elapsed time of a run during execution or the final runtime of a completed run
- The timing score of a run: WNS, TNS, WHS, THS and TPWS (see Report Timing Summary for more information on these numbers). This is where you can quickly verify that a run meets timing. If it does not meet timing, you must start the analysis with the Timing Summary Report.
- The number of nets that were not successfully routed
- A brief description of the run strategy

If you are using the Vivado IDE project flow, review the Messages tab for your active synthesis and implementation runs. Messages are grouped by run steps in the flow. All the information saved in the run log files, and the main Vivado session log file, appear in this consolidated and filtered view.

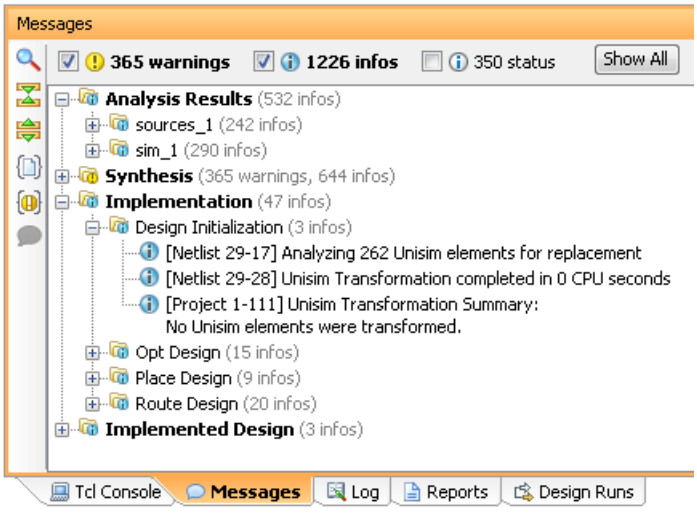


Figure 1-45: **Messages Grouped by Step**

Some messages crossprobe back to a source file that can always be opened by clicking on the file name, or in some cases to a design object related to the message. Depending on which step of the flow you are analyzing, you must open either the synthesized design or the implemented design in order to be able use the object crossprobing from the message.

Placement Analysis

This section discusses Placement Analysis and includes:

- Highlighting Placement
- Showing Connectivity
- Viewing Metrics

Highlighting Placement

Another way to review design placement is to analyze cell placement. The **Highlight Leaf Cells** command helps in this analysis.

- In the Netlist Window, select the levels of hierarchy to analyze.
- From the popup menu, select **Highlight Leaf Cells > Select a color**.
- If you select multiple levels of hierarchy, select **Cycle Colors**.

The leaf cells that make up the hierarchical cells are color coded in the Device window.

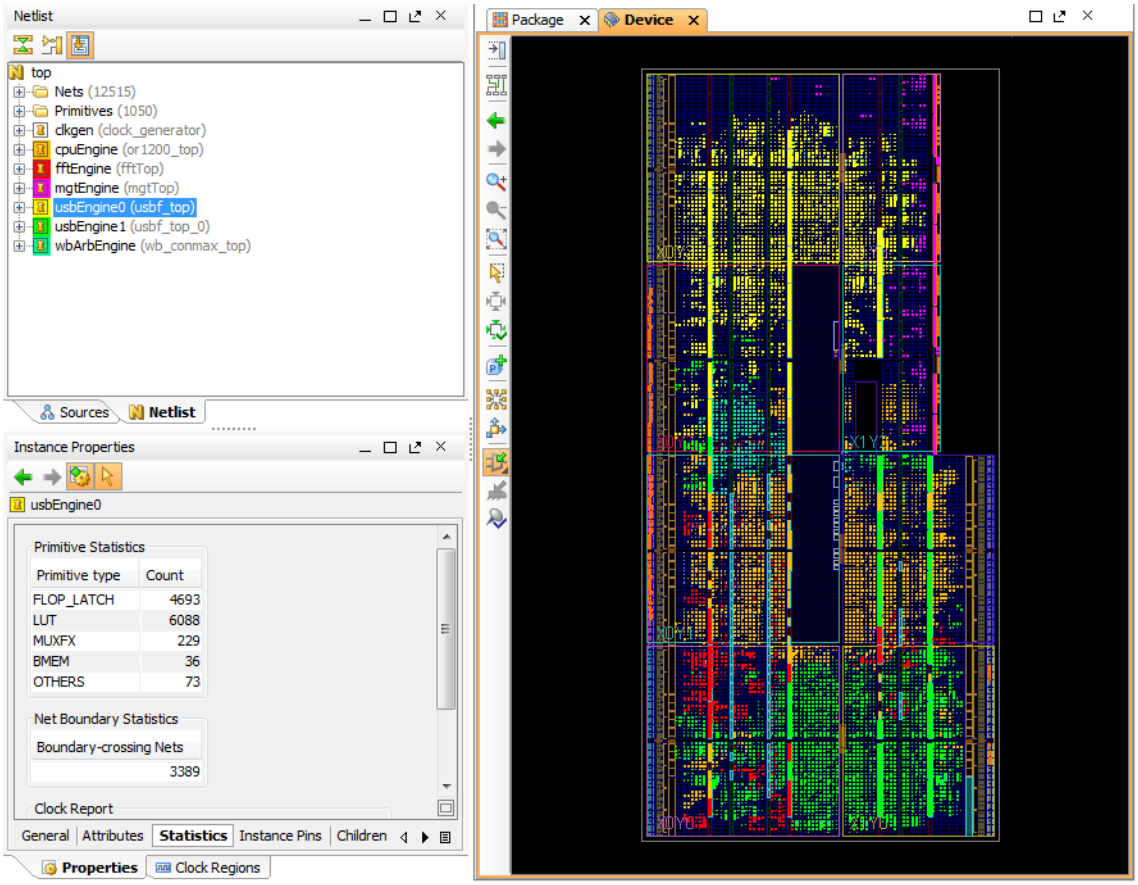


Figure 1-46: **Highlight Hierarchy**

The color coding readily shows that `UsbEngine0` (in yellow):

- Uses a number of Block RAM and DSP48 cells.
- Is in the top clock region of the chip except where the DSPs bleed out.
- Is not highly intermingled with other logic (cells) in the design.

It is easy to see that the `fftEngine` (in red) and the `cpuEngine` (in brown) are intermingled. The two blocks primarily use different resources (DSP48 as opposed to slices). Intermingling makes best use of the device.

Showing Connectivity

It can be useful to analyze a design based on connectivity. Run the **Show Connectivity** command to review the placement of all logic driven by an input, a Block RAM, or a bank of DSPs.

Show Connectivity takes a set of cells or nets as a seed, and selects objects of the other type.

TIP: *Use this technique to build up and see cones of logic inside the design.*

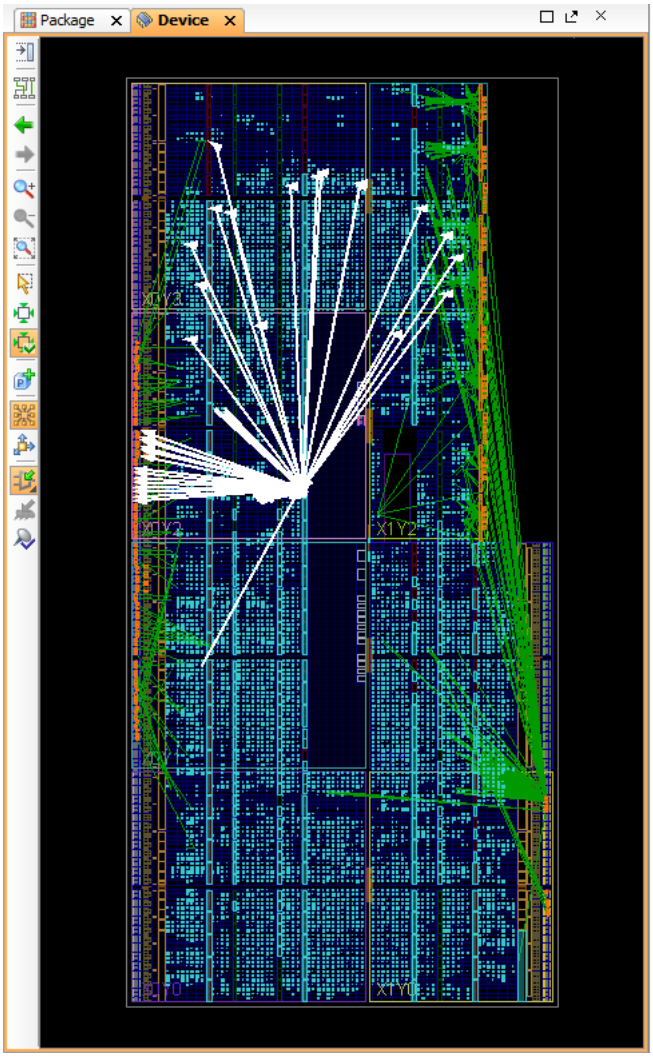


Figure 1-47: **Show Connectivity**

Figure 1-47 shows a Block RAM driving logic inside the device including OBUFs. A synthesis pragma stops synthesis from placing the output flop in the Block RAM during memory inferencing.

Fixed and Unfixed Logic

The Vivado tools track two different types of placement:

- Elements placed by the user (shown in orange) are **Fixed**.
  - **Fixed** logic is stored in the XDC.
  - **Fixed** logic normally has a LOC constraint and might have a BEL constraint.
- Elements placed by the tool (shown in blue) are **Unfixed**.

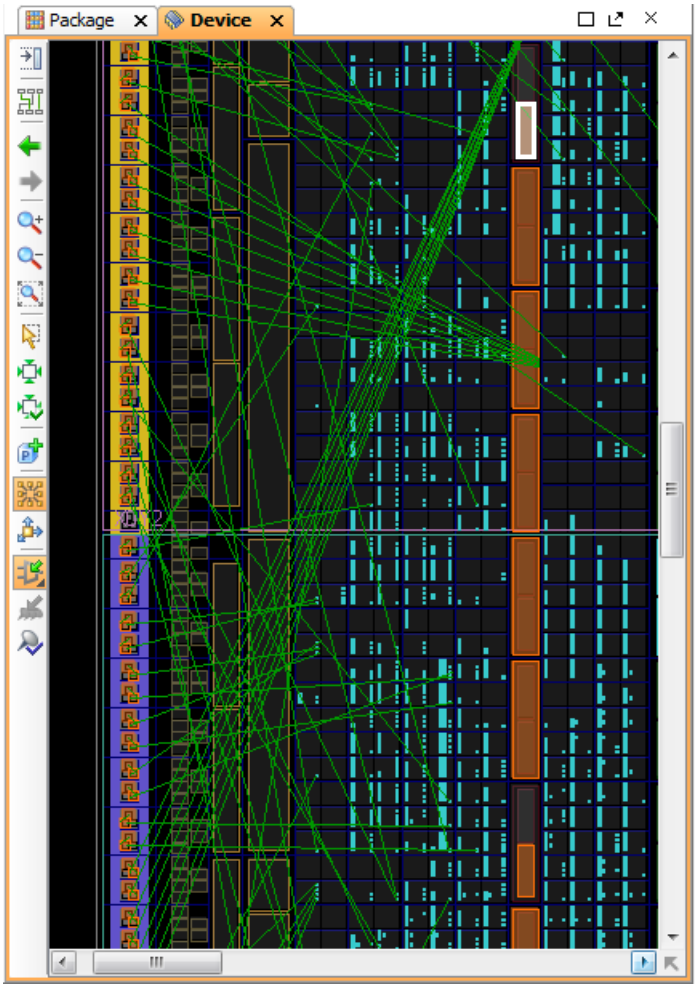


Figure 1-48: **Fixed Unfixed**

In Figure 1-48, the I/O and Block RAM placement is **Fixed**. The slice logic is **Unfixed**.

Cross Probing

For designs synthesized with Vivado Synthesis, it is possible to cross probe back to the source files once the netlist design is in memory.

To cross probe:

1. Select the gate.
2. Select `Go to instantiation` from the popup menu.

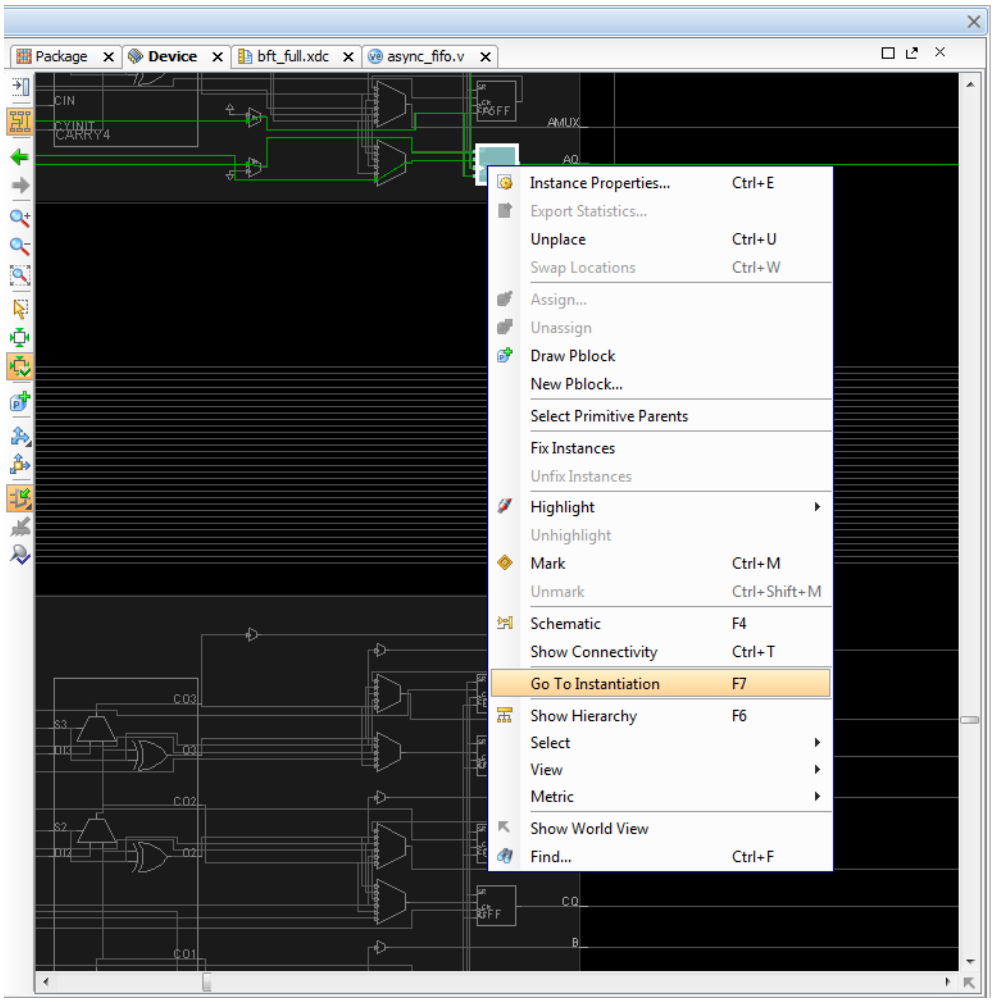


Figure 1-49: **Xprobe Back To Source**

Use crossprobing to determine which source is involved in netlist gates. Due to the nature of synthesis transforms, it is not possible to cross probe back to source for every gate in the design.

Viewing Metrics

After implementation finishes, you may want to analyze the design to see how it interacts with the device. The Vivado IDE has a number of metrics to help you determine logic and routing usage inside the device. The Metrics color code the device window based on a specified rule. To view a metric, right-click in the device view, select **Metric**, and then select the metric you would like to view.

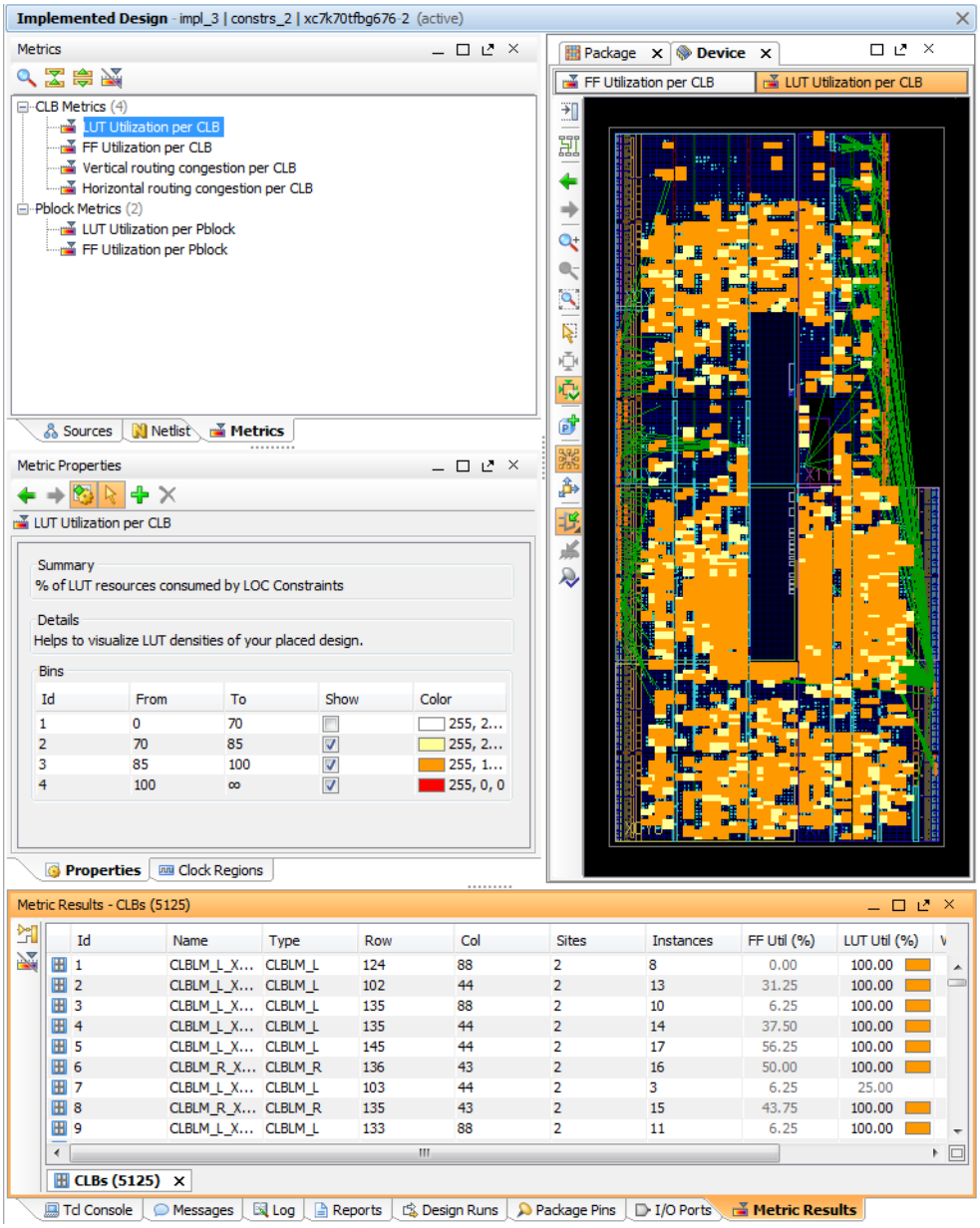


Figure 1-50: **Metrics**

Metrics Requiring a Placed Design

Four metrics require a placed design in order to be accurate. They do not require a fully routed design.

- **LUT Utilization per CLB**

  Color codes slices based on placed LUT utilization.

- **FF Utilization per CLB**

  Color codes slices based on placed FF utilization.

- **Vertical Routing Congestion per CLB**

  Color codes the fabric based on a best case estimate of vertical routing usage.

- **Horizontal Routing Congestion per CLB**

  Color codes the fabric based on a best case estimate of horizontal routing usage.

Metrics in a Netlist Design With No Placement

Two metrics are applicable if there are Pblocks. They do not depend on placement.

- **LUT Utilization per Pblock**

  Color codes the Pblock based on an estimate of how the LUTs will be placed into the slices contained in the Pblock.

- **FF Utilization per Pblock**

  Color codes the Pblock based on an estimate of how the FFs will be packed into the slices contained in the Pblock.

More than one rule can be used at a time as shown in Figure 1-50, page 72. Both LUT Utilization per CLB and FF Utilization per CLB are on.

TIP: *If there are sections of the design with high utilization or high estimates of routing congestion, consider tweaking the RTL or placement constraints to reduce logic and routing utilization in that area.*

Routing Analysis

Turn on Routing Resources in the Device View to view the exact routing resources.

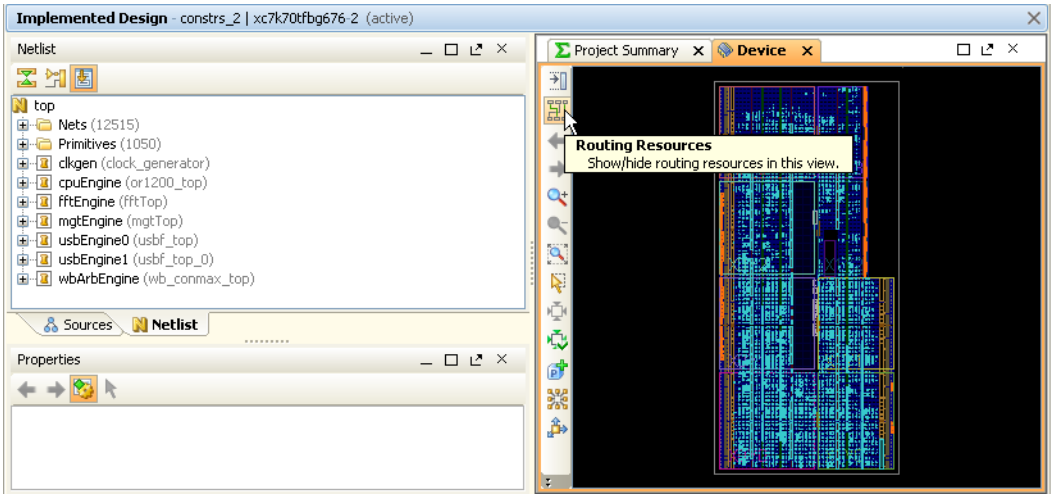


Figure 1-51: **Enable Routing**

Displaying Routing and Placement

Routing and placement display in two different ways depending on the zoom level:

- When zoomed out
- At closer zoom levels

TIP: *The two visualizations of the Device view minimize runtime and memory usage while showing the details of designs of all sizes.*

Displaying Routing and Placement When Zoomed Out

When zoomed out, an abstract view is shown. The abstract view:

- Condenses the routes through the device.
- Shows lines of different thicknesses depending on the number of routes through a particular region.

Placement similarly displays a block for each tile with logic placed in it. The more logic in a tile, the larger the block representing that tile will be.

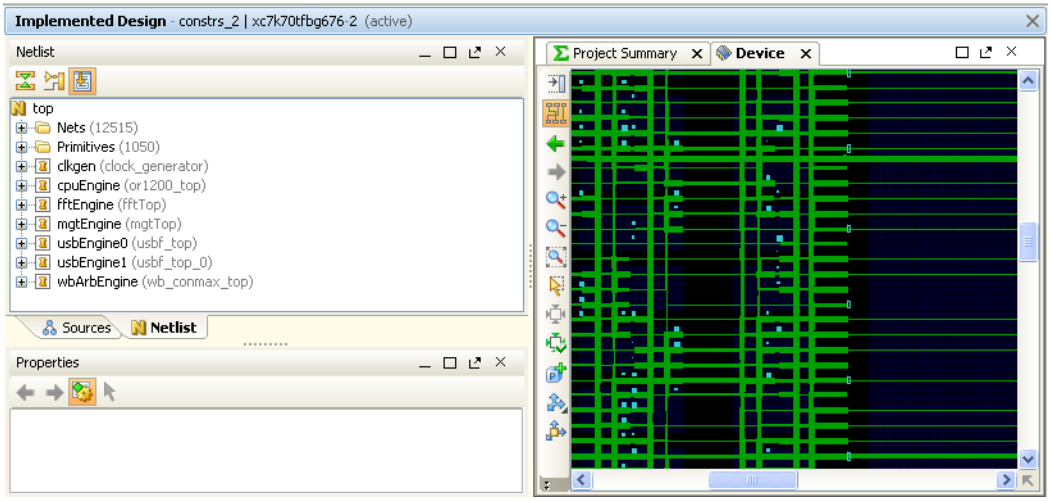


Figure 1-52: **Abstract View**

Displaying Routing and Placement at Closer Zoom Levels

At closer zoom levels, the actual logic cells and routes show.

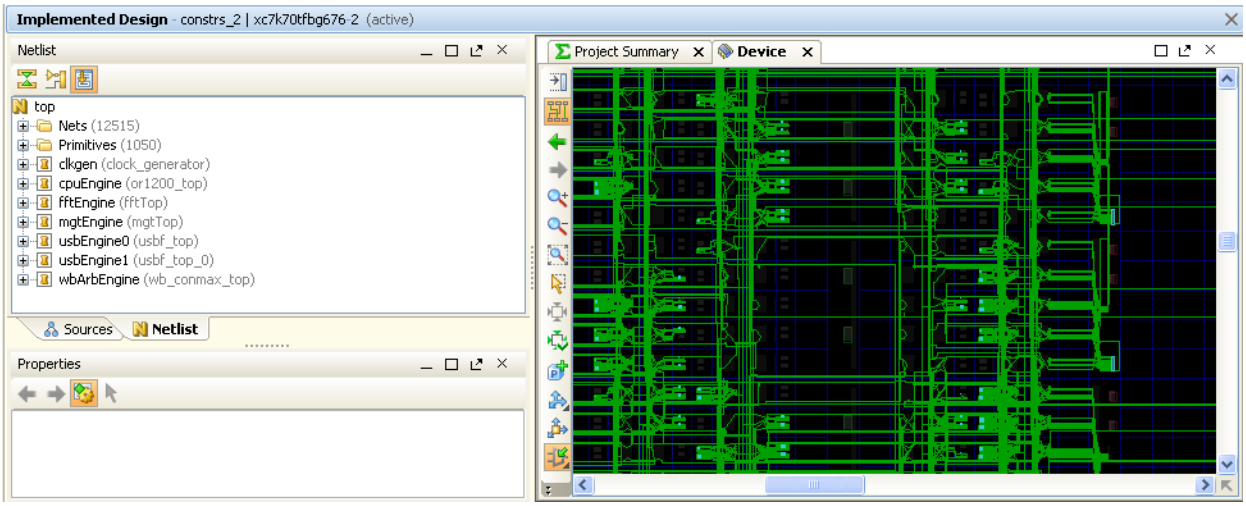


Figure 1-53: **Detailed View**

Viewing Options

The Device View is customizable to show the device, and design, in a variety of ways. Most of these are controlled through the Device View Options slideout.

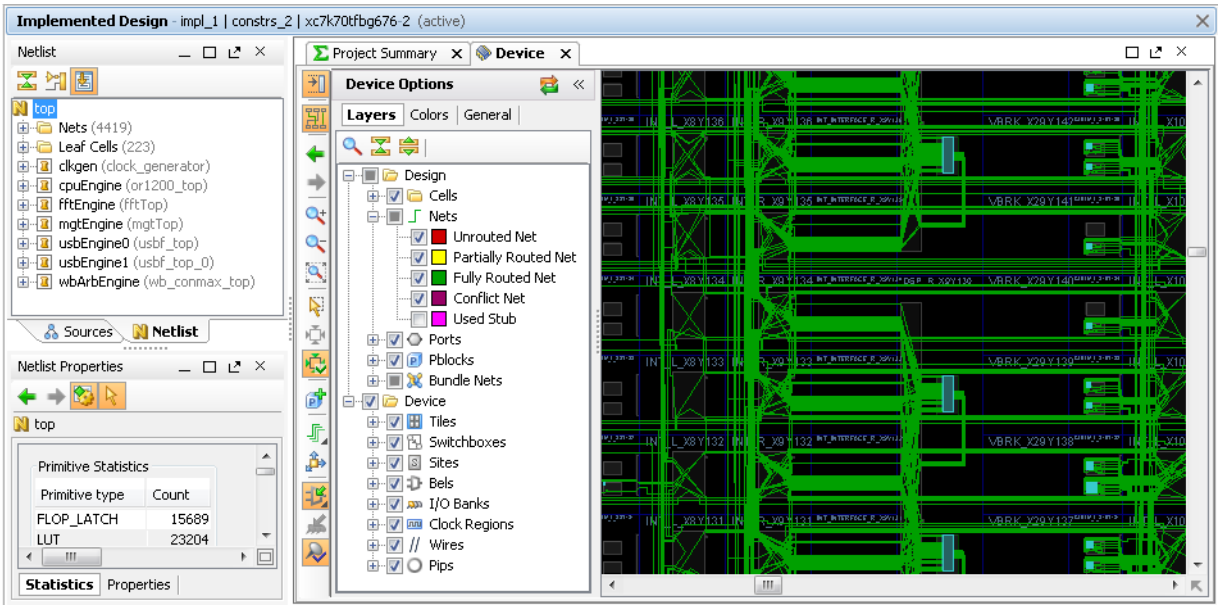


Figure 1-54: **Device View Layers**

You can enable or disable the graphics for different design and device resources, as well as modify the display colors.

Navigating in the Device View

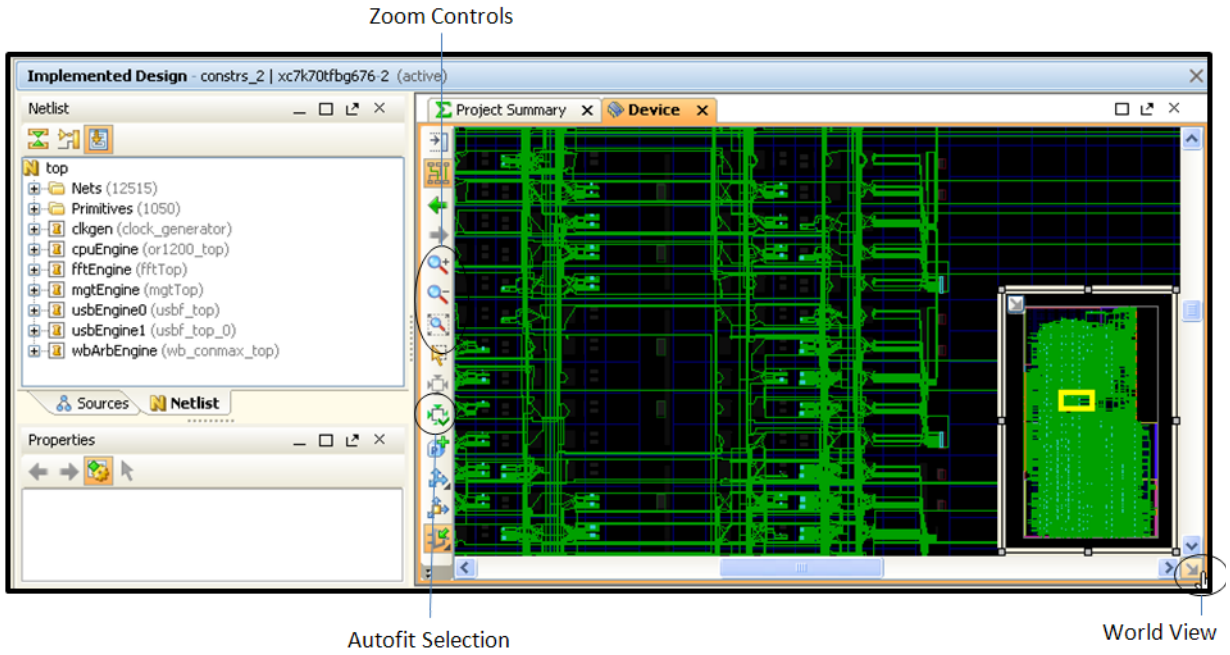


Figure 1-55: **Navigating the Device View**

Use the following tools to navigate in the Device View.

- **Zoom Controls**

  Standard Zoom In, Zoom Out, and Zoom Full tools.

- **Autofit Selection**

  Automatically zoom and pan to an object selected in any view outside of the device. Autofit Selection is particularly useful for cross probing.

- **World View**

  The World View shows where the currently visible portion of the device is on the overall device. You can move and resize the World View, as well as drag and resize the yellow box to zoom and pan.

- **Control Hotkey**

  Press **Ctrl** while clicking and dragging to pan the view.

Viewing Reports and Messages

Introduction to Reports and Messages

The Xilinx® Vivado® Integrated Design Environment (IDE) generates reports and messages to inform you of the state of the design or design processes during various tool interactions. Reports are generated by you (or by the tool) at key steps in the design flow. The reports summarize specific information about the design.

The tool generates messages automatically at each step of the design process, and for many user actions.

Messages and reports are stored in the Messages and Reports windows in the Results window area.

When you run any of the following commands, the tool starts a new process:

- Run Synthesis
- Run Implementation
- launch_runs (Tcl)

For more information on Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3], or type `<command> -help`.

The process generates messages and reports that persist on disk until you reset the run. Messages that relate to a run appear when a project is open. The tool displays only the messages for the active run in the Messages window.

Reports result from a variety of actions in the Vivado IDE:

- When you load a design, many different reporting commands are available through the Tools menu.
- Running Synthesis or Implementation creates reports as part of the run.

Viewing and Managing Messages in the IDE

Messages provide brief status notes about specific elements of the design, or about errors that occurred in tool processes.

TIP: *Review the messages to determine whether the Vivado tools are having difficulty, or are encountering errors in any sections of the design.*

Using the Reports Window

The reports for the active Synthesis and Implementation runs appear in the Reports window. Double click a report to view it in the text viewer. Select the Reports tab of the Run Properties window to view reports of the run selected in the Design Runs window.

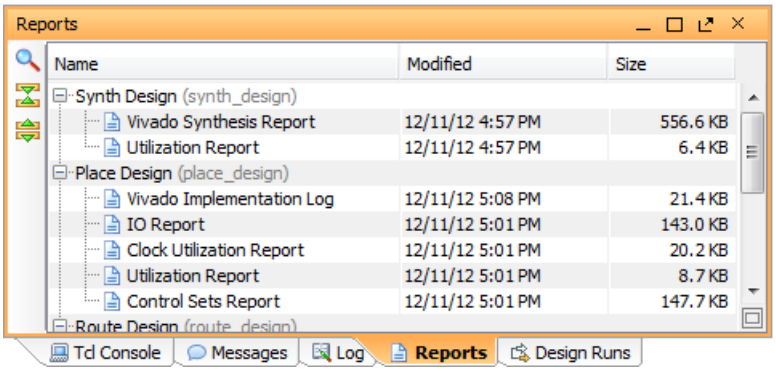


Figure 2-1: **Reports Window**

Using the Messages Window

There are two types of messages:

- Messages stored on disk
- Messages stored in memory

The Vivado Integrated Design Environment (IDE) groups messages in the Messages window by the action that created the message.

Use the command buttons on the toolbar menu to group the messages by message ID or file.

Figure 2-2: **Messages Window**

Some messages include hyperlinks to a file or a design element to help in debugging. Click the link to view the source.

TIP: *Use the popup menu to copy messages to paste into another window or document.*

Each message is labeled with a message ID and a message severity.

- **Message ID**

  The message ID identifies different messages, allowing them to be grouped and sorted.

- **Message Severity**

  The message severity describes the nature of the information presented.

Some messages require your attention and resolution before the design can be elaborated, synthesized, or implemented. Some messages are informational only. Informational messages provide details about the design or process, but require no user action.

Table 2-1: **Message Severities**

Icon	Severity	Message
	Status	Communicates general status of the design processing.
	Info	General status of the process and feedback regarding design processing.
	Warning	Design results may be sub-optimal because constraints or specifications may not be applied as intended.
	Critical Warning	Certain user input or constraints will not be applied, or are outside the best practices, which usually leads to an error later on in the flow. Examine their sources and constraints. Changes are highly recommended.
	Error	An issue that renders design results unusable and cannot be resolved without user intervention. The design flow stops.

RECOMMENDED: *Carefully review all errors and critical warnings issued by the tools when loading a design in memory, or from your active synthesis and implementation run. The messages provide information about problems that require your attention. Many messages include a longer description. along with resolution advice that can be displayed by clicking on the message ID.*

For an example, see Figure 2-3. In this example, a primary clock constraint refers to a port that cannot be found in the design (first warning), so the clock is not created (first critical warning) and any other constraints that refer to this clock fail as well.

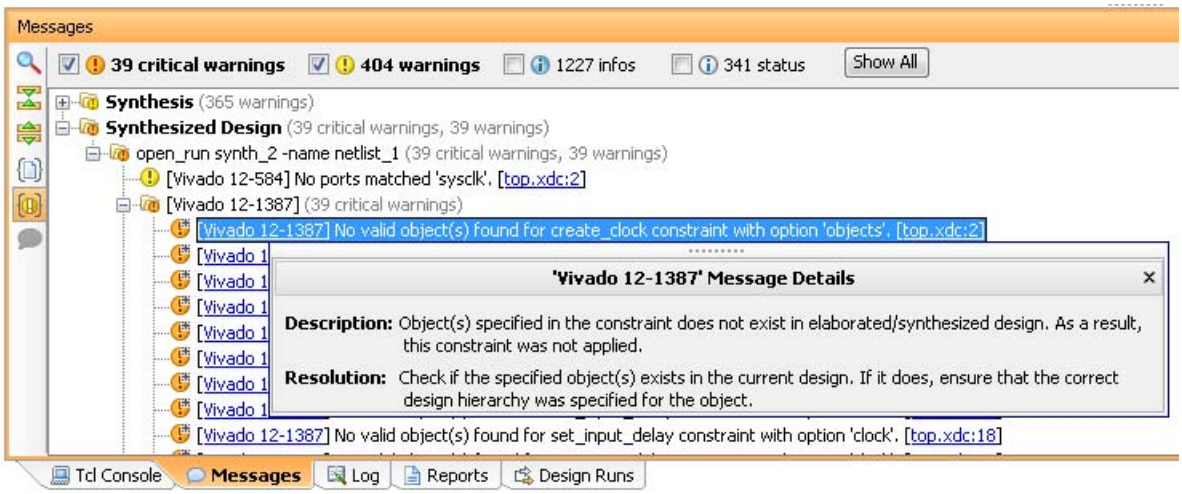


Figure 2-3: **Reviewing Errors and Critical Warning**

Filtering Messages

You can filter messages by severity.

To enable or disable the display of a specific message type:

1. Go to the Messages window.
2. Select (to enable) or deselect (to disable) the checkbox next to a message severity in the window header.

You can change the severity of a specific message ID. For example, you can decrease the severity of a message you do not believe is critical, or increase the severity of a message you think demands more attention.

To increase or decrease the severity of a message, use the **`set_msg_severity`** Tcl command. For example:

```
set_msg_severity "Common 17-81" "CRITICAL WARNING"
```

For more information on Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3].

Vivado Generated Reports and Messages

This section discusses Vivado Generated Reports and Messages and includes:

- Synthesis Report and Messages
- Implementation Log
- WebTalk Report

Synthesis Report and Messages

The Vivado Synthesis Report is the primary output from the Vivado Synthesis tool including:

- The files processed, which are:
  - VHDL
  - Verilog
  - System Verilog
  - XDC
- Parameter settings per cell
- Nets with Multiple Drivers
- Undriven hierarchical pins
- Optimization information
- Black boxes
- Final Primitive count
- Cell usage by Hierarchy
- Runtime and memory usage

IMPORTANT: *Review this report or the messages tab for Errors, Critical Warnings and Warnings. The Synthesis tool can issue Critical Warnings and Warnings that become more serious later in the flow.*

Implementation Log

The Vivado Implementation Log includes:

- Information about the location, netlist, and constraints used.
- Logic optimization task. The tool runs logic optimization routines by default to generate a smaller and faster netlist.
- The placement phases, plus a post-placement timing estimate (WNS and TNS only).
- The router phases, plus several timing estimates and a post-routing timing summary (WNS, TNS, WHS and THS only).
- Elapsed time and memory for each implementation command and phases.

Review this report or the proper section of the messages tab for Errors, Critical Warnings and Warnings. The Placer generates warnings that may be elevated to Errors later in the flow. If using Stepwise runs, the log contains only the results for the last step.

IMPORTANT: *Review the Timing Summary Report to view: (1) the Pulse Width timing summary, and (2) additional information about timing violations or missing constraints.*

WebTalk Report

The WebTalk Report is generated during Route and Bitstream. The report collects information about how you use Xilinx parts. This helps Xilinx provide you with better software. No proprietary information is collected. For more information, go to:

http://www.xilinx.com/ise/webtalk/

Creating Design Related Reports

This section discusses Creating Design Related Reports and includes:

- Report Utilization
- Report I/O
- Report Clock Utilization
- Report Control Sets
- Report DRC
- Report Route Status
- Report Noise
- Report Power

Report Utilization

The Utilization Report is generated during various steps in the flow by **`report_utilization`**. The report includes the device used for the run and utilization for:

- **Slice Logic**
  - LUT
  - MuxFx
  - Register
- **Memory**
  - BlockRam
  - FIFO
- **DSP48E1**
- **I/O Resources**
- **Clocking Resources**
  - BUFGCTRL
  - BUFR
  - BUFHCE
  - MMCME2_ADV
  - PLLE2_ADV
- **Specific Device Resources**:
  - STARTUPE2
  - XADC
- **Primitive type count sorted by usage**
- **Black Boxes**
- **Instantiated Netlists**

When run from the Tcl Console, the report can include usage of a particular hierarchical cell when using the **`-cells`** option. When run from the Vivado IDE graphical user interface, this information appears in an interactive table.

The numbers may change at various points in the flow, when logic optimization commands change the netlist.

Report I/O

The I/O Report replaces the Xilinx® ISE® Design Suite PAD file. The I/O Report lists:

- **Pin Number**

  All the pins in the device

- **Signal Name**

  The name of the user I/O assigned to the pin

- **Pin Usage**

  The type of pad or buffer used by the pin

- **Pin Name**

  Name of the pin

- **Direction**

  Whether the pin is an input, output, inout, or unused

- **I/O Standard**

  The I/O standard for the User I/O

  An asterisk (*) indicates that it is the default. This differs from the I/O Ports window of the Vivado IDE.

- **I/O Bank Number**

  The I/O Bank where the pin is located

- **Drive (mA)**

  The drive strength in milliamps

- **Slew Rate**

  The Slew Rate configuration of the buffer: Fast or Slow

- **Termination**

  The off chip termination settings

- **IOB Delay**

  Delay value set for this pin

- **Voltage**

  The values for various pins, including VCCO, VCCAUX, and related pins

- **Constraint**

  Displays Fixed if the pin has been constrained by the user

- **IOB Sequential Element**

  The flip flops packed into the I/O Bank next to the given port

- **Signal Integrity**

  The Signal Integrity of the pin

Report Clock Utilization

The Clock Utilization Report helps you analyze the utilization of clock resources inside the device. It can be useful for debugging clock placement issues. The Clock Utilization Report displays:

- The number of clocking primitives available, occupied, and constrained
- Loading and skew per BUFG

  Look for nets with large maxdelay and skew.

- Loading and skew per MMCM

  Look for nets with unexpected loading, large maxdelay, and skew.

Regional Clocks

Regional clock networks are clock networks independent of the global clock network. Unlike global clocks, the span of a regional clock signal (BUFR) is limited to one clock region. One I/O clock signal drives a single bank.

These networks are especially useful for source-synchronous interface designs. The I/O banks in Xilinx 7 series FPGA devices are the same size as a clock region.

Local Clocks

Local clocks are clock networks routed onto general routing resources.

RECOMMENDED: *Avoid local clocks where possible. They can experience very large clock skew, and are more susceptible to PVT variations. The tools may route the clock differently each time you rerun implementation.*

The Locked column for the clocking resources shows whether you placed the clock, or whether the tools are free to place the clocking resource.

If there are too many global clocks, consider moving low fanout global clocks to other clocking resources, such as BUFH or BUFR.

Report Control Sets

A control set is the unique combination of a clock signal, a clock enable signal, and a set/reset signal. Each slice supports at most one control set which any flip flop located in it can use. Flip flops with different control sets cannot be placed in the same slice.

The Control Sets Report lists the number of unique control sets in the design. Based on the placement of the designs, the tool displays the minimum number of register sites lost to the control set packing.

- **Clock Signal**

  The logical clock signal name

- **Enable Signal**

  The logical clock enable signal name

- **Set/Reset Signal**

  The logical set/reset signal name

- **Slice Load Count**

  The number of unique slices that contain cells connected to the control set

- **BEL Load Count**

  The number of cells connected to the control set

Report DRC

The DRC Report is generated by the router. Before the router runs, the tool checks for a common set of design issues. The report lists the checks used in the run.

IMPORTANT: *Review the Critical Warnings. The severity of a particular check may be increased later in the flow.*

Report DRC runs common Design Rule checks to look for common design issues and errors.

Elaborated Design

The tool checks for DRCs related to I/O, Clock Placement, potential coding issues with your HDL, and XDC constraints. The RTL netlist typically does not have all the I/O Buffers, Clock Buffers, and other primitives the post synthesis designs have. Elaborated Design DRCs do not check for as many errors as later DRCs.

Synthesized Design and Implemented Design

- Checks for DRCs related to the post synthesis netlist.
- Checks for I/O, BUFG, and other placement.
- Basic checks on the attributes wiring on MGTs, IODELAYs, and other primitives.
- The same DRCs run taking into account any available placement and routing.
- DRCs have four severities: Info, Warning, Critical Warning, and Error. Critical Warnings and Errors do not block the design flow at this point.

Steps of the implementation flow also run the DRCs, which can stop the flow at critical points. The placer and router check for issues that block placement. Certain messages have a lower severity depending on the stage. These are DRCs flagging conditions that do not stop **opt_design**, **place_design,** or **route_design** from completing, but which can lead to issues on the board.

For example, some DRCs check that the user has manually constrained the package pin location and the I/O standard for all design ports. If some of these constraints are missing, **place_design** and **route_design** issue critical warnings. However, these DRCs appear as an ERROR in **write_bitstream**. The tools will not program a part without these constraints.

The decreased severity earlier in the flow allows you to run the design through implementation iterations before the final pinout has been determined. You must run bitstream generation for a comprehensive DRC signoff.

Figure 2-4 shows the Vivado IDE graphical user interface form of Report DRC.

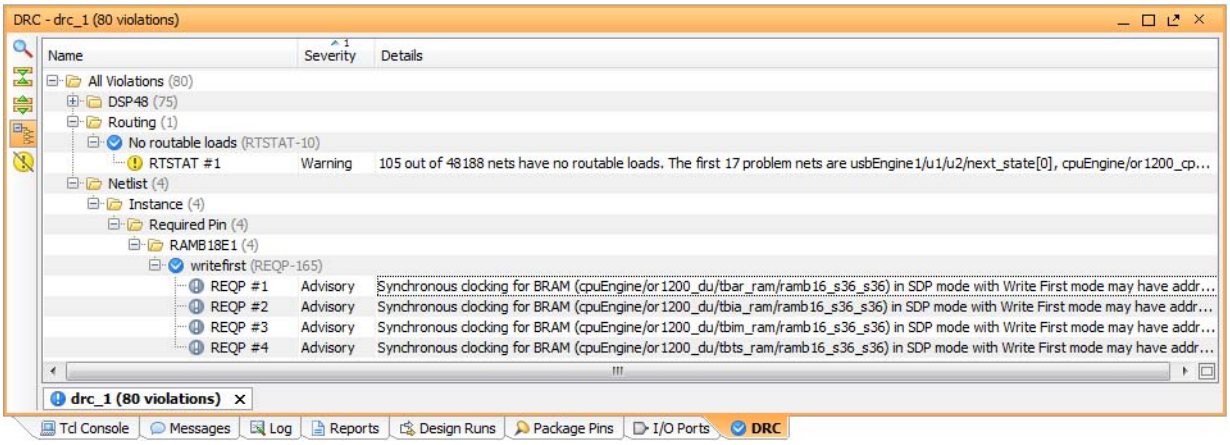


Figure 2-4: **DRC Report**

Click a DRC to open the properties for a detailed version of the message. Look in the Properties window to view the details. Most messages have a hyperlink for nets, cells, and ports referenced in the DRC.

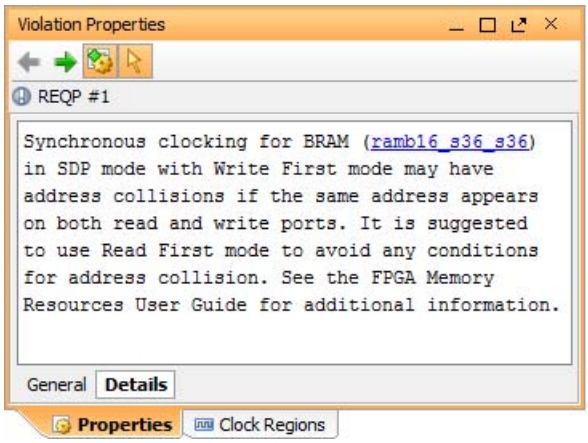


Figure 2-5: **DRC Properties**

The DRC report is static. You must rerun Report DRC for the report to reflect design changes. The tool determines that the links are stale after certain design operations (such as deleting objects and moving objects), and invalidates the links.

Selecting an object from the hyperlink selects the object, but does not refresh the Properties window. To display the properties for the object, you must unselect and reselect it.

To create a DRC report in Tcl, run:

```
report_drc
```

To write the results to a file, run:

```
report_drc -file myDRCs.txt
```

TIP: *For more information on report_drc, run report_drc -help.*

Report Route Status

The Route Status Report is generated during the implementation flow and is available by using the **report_route_status** Tcl command.

The Route Status Report displays a breakdown of the nets in the design as follows:

- The total number of logical nets in the design
  - The number of nets that do not need routing resources
    - The number of nets that do not use routing resources outside of a tile. Examples include nets inside of a CLB, BlockRam, or I/O Pad.
    - The number of Nets without loads, if any exist
  - The number of routable nets that require routing resources
    - The number of unrouted nets, if any exist
    - The number of fully routed nets
  - The number of nets with routing errors
    - The number of nets with some unrouted pins, if any exist
    - The number of nets with antennas/islands, if any exist
    - The number of nets with resource conflicts, if any exist

The following is an example of the Report Route Status for a fully routed design:

```
Design Route Status
                                                :      # nets :
   ------------------------------------------- : ------------ :
   # of logical nets..........................  :         6137 :
       # of nets not needing routing..........  :          993 :
           # of internally routed nets........  :          993 :
       # of routable nets.....................  :         5144 :
           # of fully routed nets.............  :         5144 :
       # of nets with routing errors..........  :            0 :
   ------------------------------------------- : ------------ :
```

Report Noise

The Report Noise command performs the Simultaneous Switching Noise (SSN) calculation for Xilinx 7 series FPGA devices. By default, the SSN report opens in a new tab in the Results window area of the Vivado IDE. You can export the results to a CSV or HTML file.

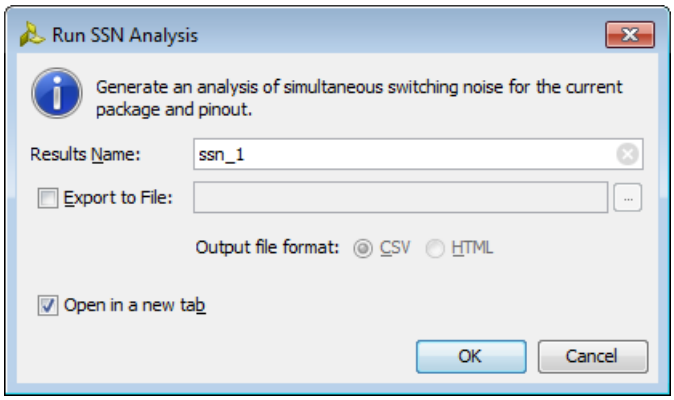


Figure 2-6: **Run SSN Analysis**

The Noise Report has four sections:

- Noise Report Summary Section
- Noise Report Messages Section
- Noise Report I/O Bank Details Section
- Noise Report Links Section

Noise Report Summary Section

The Summary section of the Noise Report includes:

- When the report ran
- Number and percentage of applicable ports analyzed
- Status, including whether it passed
- Number of Critical Warnings, Warnings, and Info messages

Noise Report Messages Section

The Messages section of the Noise Report includes a detailed list of the messages generated during the report.

Noise Report I/O Bank Details Section

The I/O Bank Details section of the Noise Report includes a list of Pins, Standards, and Remaining Margin.

Noise Report Links Section

The Links section of the Noise Report contains links to documentation located online at www.xilinx.com/support.

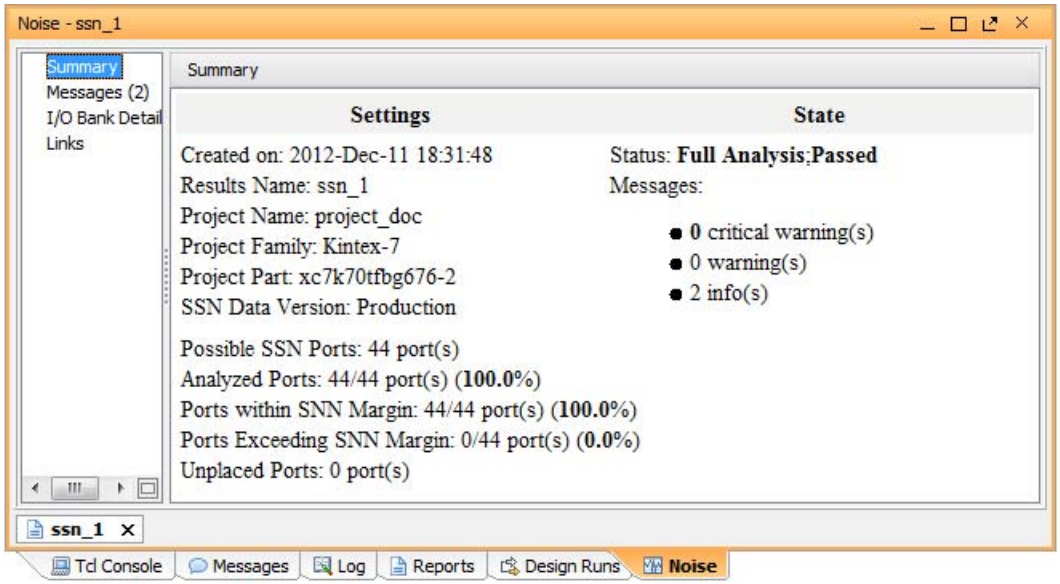


Figure 2-7: **Noise Report**

To create an HTML version of the report, select the option or run the following Tcl command:

```
report_ssn -format html -file myImplementedDesignSSN.html
```

Report Power

The Power Report is generated after routing to report details of power consumption based on the current operating conditions of the device, and the switching rates of the design. Power analysis requires a synthesized netlist, or a placed and routed design.

- Use the **`set_operating_conditions`** command to set operating conditions.
- Use the **`set_default_switching_activity`** command to define switching activity.

The Report Power command is available when a Synthesized Design or an Implemented Design is open.

The Power Report estimates power consumption and junction temperature based on design inputs, including:

- Thermal statistics, such as junction and ambient temperature values.
- Data on board selection, including number of board layers and board temperature.
- Data on the selection of airflow and the head sink profile used by the design.
- Reporting the FPGA device current requirements from the different power supply sources.
- Allowing detailed power distribution analysis to guide power saving strategies and to reduce dynamic, thermal or off-chip power.
- Simulation activity files can be used to make power estimation more accurate.

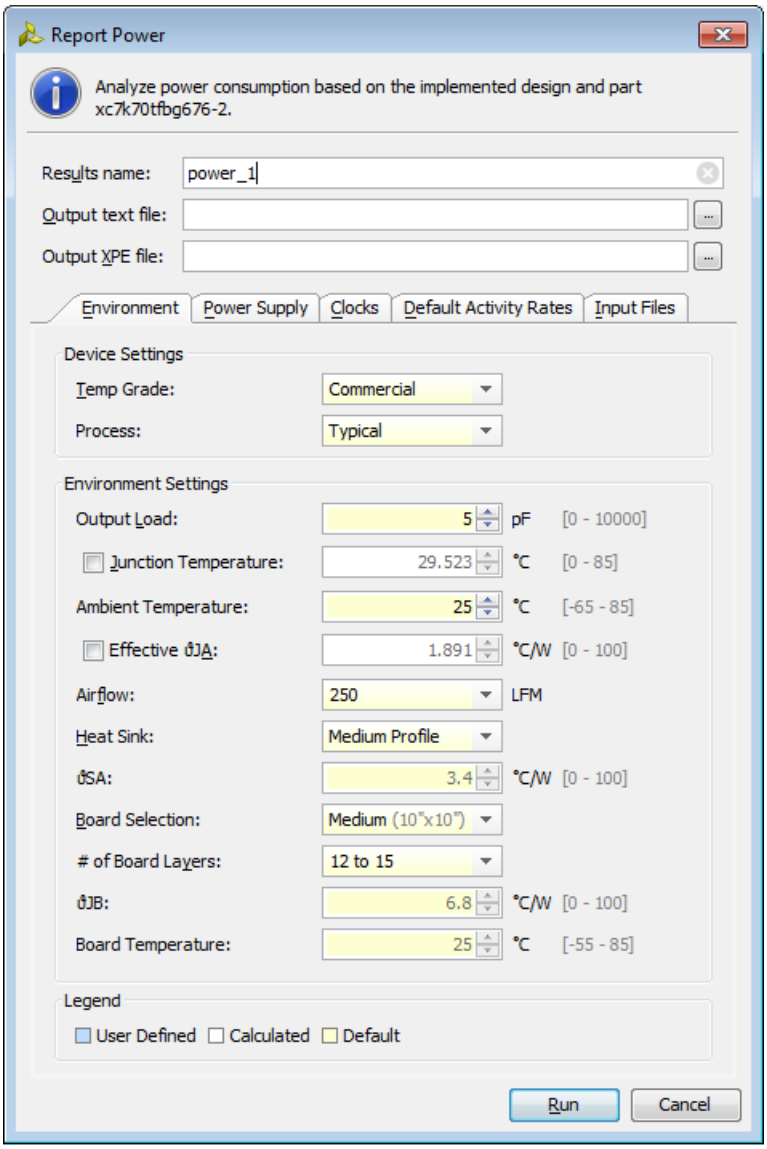


Figure 2-8: **Report Power Dialog Box**

Analyzing the Power Report

Use the Report Power dialog box (Figure 2-8, page 93) to analyze power based on:

- Settings
- Power total
- Hierarchy
- Voltage rail
- Block type

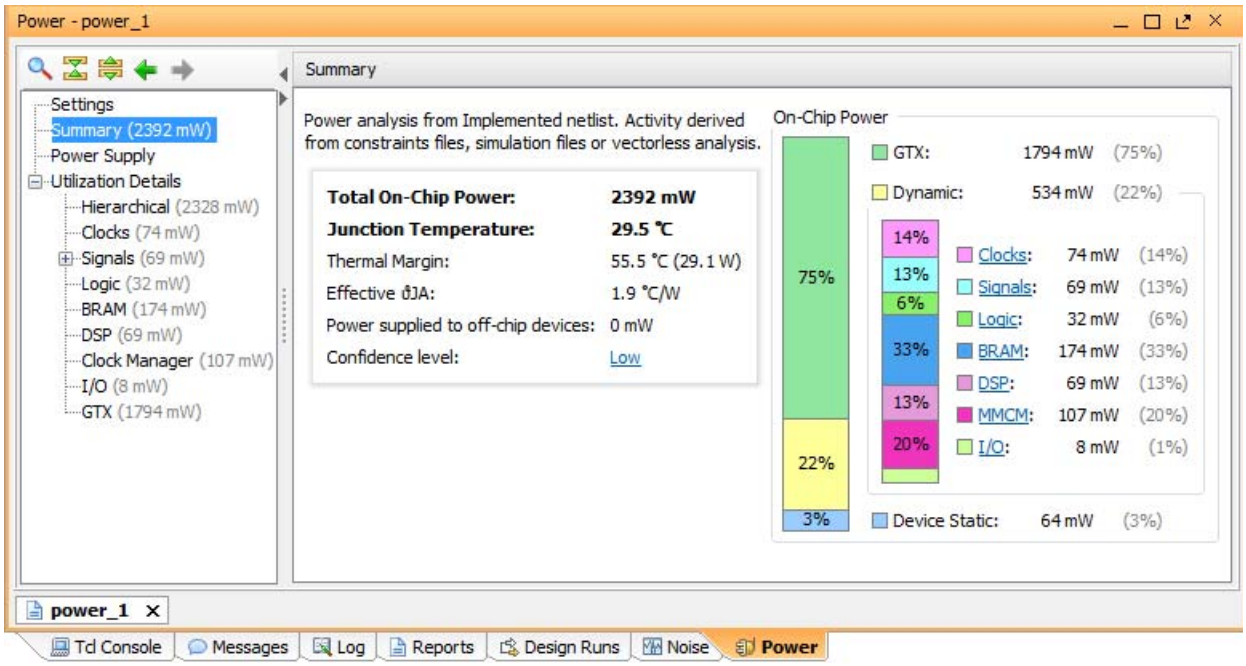


Figure 2-9: **Power Report**

For more information on the power report and analyzing the results, see the *Vivado Design Suite User Guide: Power Analysis and Optimization* (UG907) [Ref 8].

A text version of the power report is generated by default after route during the implementation process.

Reporting Power in a Non-Project Flow

In the non-project flow, **`report_power`** is available after **`link_design`** or **`synth_design`**. The report generated uses the available placement and routing to give more accurate power numbers. To generate this report from the Tcl Console or a script, run **`report_power.`**

Performing Timing Analysis

Introduction to Timing Analysis

The Xilinx® Vivado® Integrated Design Environment (IDE) provides several reporting commands to verify that your design meets all timing constraints and is ready to be loaded on the application board. Report Timing Summary is the timing signoff report, equivalent to TRCE in the ISE® Design Suite. Report Timing Summary provides a comprehensive overview of all the timing checks, and shows enough information to allow you to start analyzing and debugging any timing issue. For more information, see Chapter 1, Design Analysis Within the IDE.

You can generate this report in a window, write it to a file, or print it in your log file. Whenever Report Timing Summary shows that your design does not meet timing, or is missing some constraints, you can explore the details provided in the various sections of the summary and run more specific analysis.

The other timing reports provide more details on a particular situation or to scope the analysis to some logic by using filters.

Verifying Timing Signoff

Before going into the details of timing analysis, it is important to understand which part of the timing reports indicates that your design is ready to run in hardware.

IMPORTANT: *Timing signoff is a mandatory step in the analysis of the implementation results, once your design is fully placed and routed.*

By default, when using projects in the Vivado Design Suite, the runs automatically generate the text version of Report Timing Summary. You can also generate this report interactively after loading the post-implementation design checkpoint in memory.

Timing Signoff Criteria

Timing signoff is a combination of two criteria:

- Your design is fully constrained.
- Your design meets timing.

Your Design Is Fully Constrained

Review the Check Timing section to verify that your design is fully constrained. Check Timing must show that:

- All non-constant clock pins are reached by a defined clock (**no_clock check**).
- All internal path endpoints are timed (**unconstrained_internal_endpoint check**).
- All input and output ports are fully constrained (**no_input_delay**, **no_output_delay**, **partial_input_delay**, **partial_output_delay** checks).

The problems reported by the two following checks must be thoroughly reviewed and addressed:

- All generated clocks are not properly timed (**generated_clock** check). This occurs when there is no logical or active timing path between a generated clock and its master clock. It impacts skew computation and potentially slack violations.
- There are some combinatorial loops in the design (loops checks). The timing engine breaks combinatorial loops at random points which will reduce the coverage of the analysis.

The two last checks are informational and usually highlight overly-pessimistic timing situations:

- There are some combinatorial loops that propagate through a latch (**latch_loops check**). The Vivado IDE timing engine does not break **latch_loops**, but considers maximum time borrowing, which can lead to over-pessimistic latch timing analysis.
- The clocks present on any clock pin are not all exclusive (m**ultiple_clocks** check). This check does not need to be clean, because it usually highlights unrealistic timing paths that are timed while the rest of the design is still properly timed. In some cases, unrealistic timing paths can make it harder for the implementation tools to close timing.

TIP: *You can waive some of the missing constraints, at the risk of lowering the signoff quality of your design.*

Your Design Meets Timing

- Total Negative Slack (TNS) is 0ns.

  Covers max delay analysis (setup/recovery).

- Total Hold Slack (THS) is 0ns.

  Covers min delay analysis (hold/removal).

- Total Pulse Width Slack (TPWS) is 0ns.

  Equivalent to Component Pin Switching Limit in ISE. It is performed with both min and max delays.

The sum of TNS, THS, and TPWS is equivalent to the ISE final Timing Score.

Verifying Clean Timing Signoff

Figure 3-1, Report Timing Summary Signoff in Vivado IDE, highlights in green the information you must examine in order to verify that the timing signoff is clean.

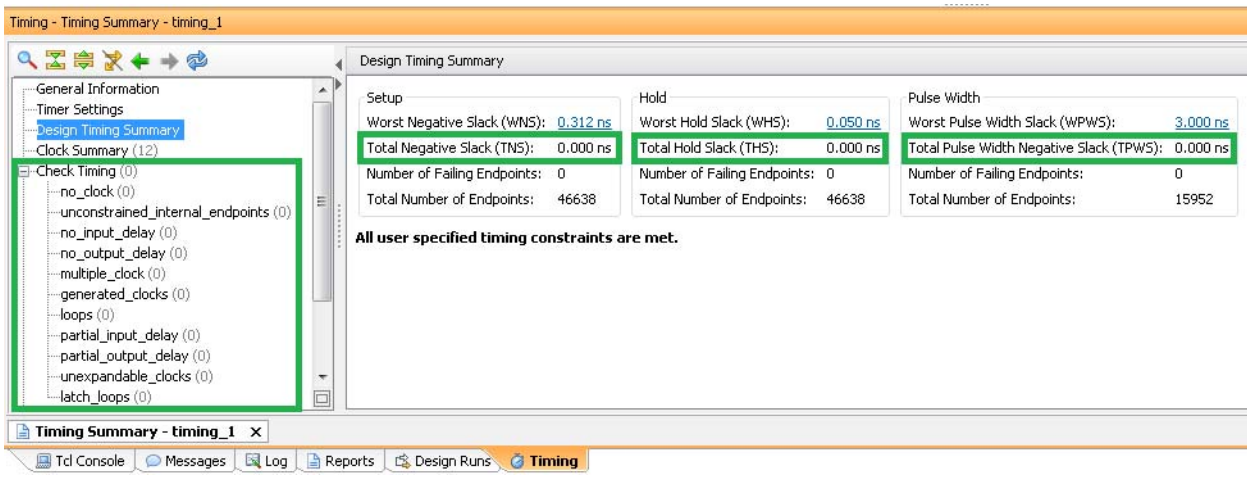


Figure 3-1: **Report Timing Summary Signoff in Vivado IDE**

Figure 3-2 shows the Check Timing information to verify in the text report.

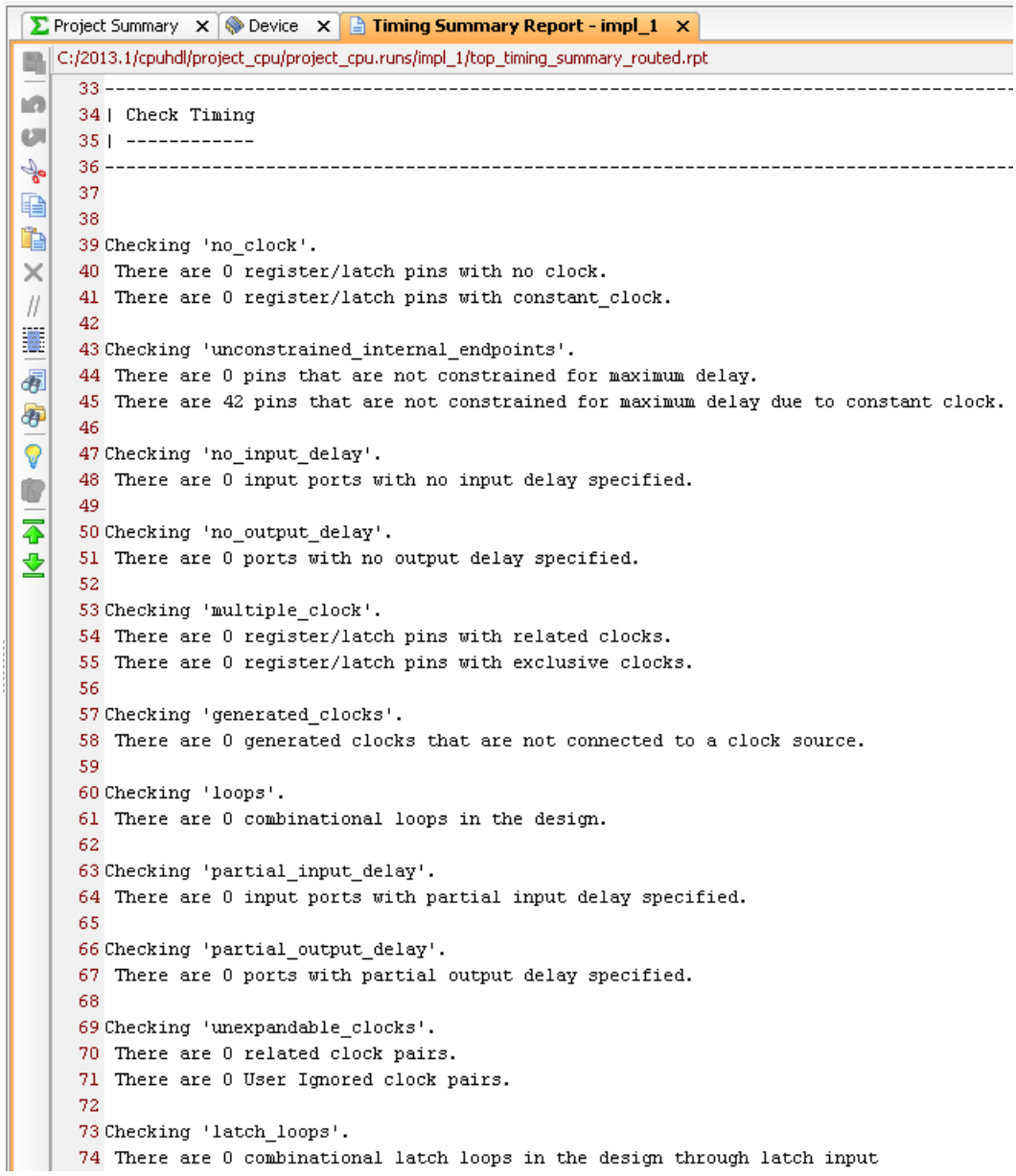


Figure 3-2: **Check Timing Signoff in Text Report**

Figure 3-3 shows the Design Timing Summary information to verify in the text report.

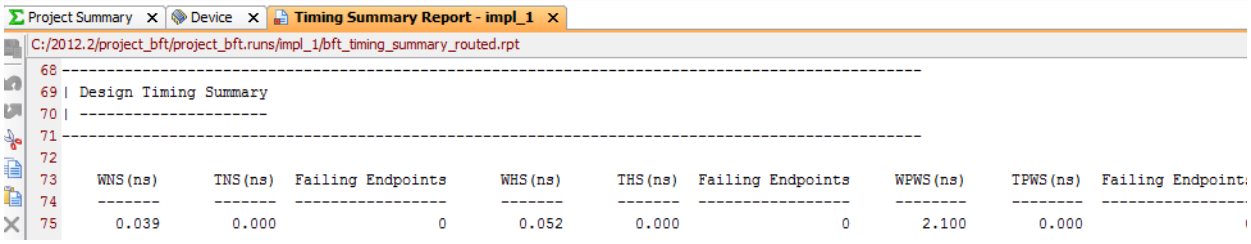


Figure 3-3: Design Timing Summary Signoff in Text Report

Investigating Timing Violations

In the Vivado IDE, slack violations are reported in red. Missing constraints are not highlighted with a particular color.

To investigate timing violations, review the following sections:

- The Intra-Clock Paths, Inter-Clock Paths and Path Groups (****async_default****) sections provide information on setup, recovery, hold, and removal violations.
- Intra-Clock Paths provides details on Pulse Width check violations.

For more information on the report windows, see Report Timing Summary, page 19, in Chapter 1, Design Analysis Within the IDE.

TIP: *To display violations only, click the **Show only failing checks** button.*

If you used the default options, the Timing Summary report includes the details of the N-worst paths for each clock pair and for each type of analysis.

- The GUI default for N is 10 (ten).
- The **report_timing_summary** command default for N is 1 (one).

To directly review the timing path details without running another report, double click the path. If not enough paths are reported, either (1) rerun Report Timing Summary with more paths (Tcl equivalent: **report_timing_summary -max_paths N**), or (2) run Report Timing on a particular clock pair or a particular timing path.

For more information on Tcl commands, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) [Ref 3], or type `<command> -help`.

Reading a Timing Path Report

The timing path report provides the information needed to understand what causes a timing violation. The following sections describe the Timing Path Report.

The Timing Path Summary displays the important information from the timing path details. You can review it to find out about the cause of a violation without having to analyze the details of the timing path. It includes slack, path requirement, datapath delay, cell delay, route delay, clock skew, and clock uncertainty. It does not provide any information about cell placement.

For more information about the terminology used for timing constraints and timing analysis, as well as learn how slack and path requirement are determined, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

Timing Path Summary Header Examples

Figure 3-4 shows an example of the Timing Path Summary Header in a text report.

```
Slack (MET) :             1.430ns  (required time - arrival time)
  Source:                 ingressFifoWrEn_reg/C
                            (rising edge-triggered cell FDRE clocked by wbClk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]/CE
                            (rising edge-triggered cell FDCE clocked by bftClk  {rise@0.000ns fall@2.500ns period=5.000ns})
  Path Group:             bftClk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            5.000ns
  Data Path Delay:        2.915ns  (logic 0.302ns (10.359%)  route 2.613ns (89.641%))
  Logic Levels:           1  (LUT2=1)
  Clock Path Skew:        -0.418ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    3.792ns = ( 8.792 - 5.000 )
    Source Clock Delay      (SCD):    4.210ns
    Clock Pessimism Removal (CPR):    0.000ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
  Clock Domain Crossing:  Inter clock paths are considered valid unless explicitly excluded by timing constraints such as set_
```

Figure 3-4: **Timing Path Summary Header in Text Report**

Figure 3-5 shows an example of the Timing Path Summary header in the Vivado IDE.

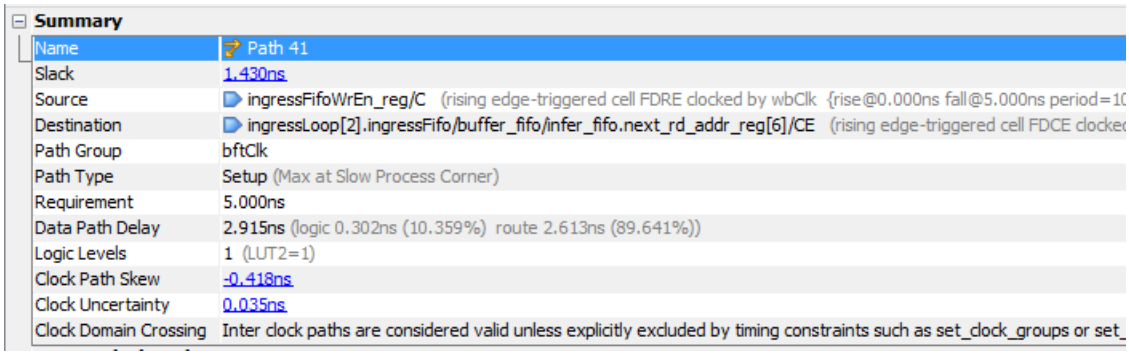


Figure 3-5: **Timing Path Summary Header in Vivado IDE**

Timing Path Summary Header Information

The Timing Path Summary header includes the following information:

- **Slack**

  A positive slack indicates that the path meets the path requirement, which is derived from the timing constraints. The Slack equation depends on the analysis performed.

- **Max delay analysis (setup/recovery)**

  ```
  slack = data required time - data arrival time
  ```

- **Min delay analysis (hold/removal)**

  ```
  slack = data arrival time - data required time
  ```

  Data required and arrival times are calculated and reported in the other sub-sections of the timing path report.

- **Source**

  The path startpoint and the source clock that launches the data. The startpoint is usually the clock pin of a sequential cell or an input port.

  When applicable, the second line displays the primitive and the edge sensitivity of the clock pin. It also provides the clock name and the clock edges definition (waveform and period).

- **Destination**

  The path endpoint and the destination clock that captures the data. The endpoint is usually the input data pin of the destination sequential cell or an output port. Whenever applicable, the second line displays the primitive and the edge sensitivity of the clock pin. It also provides the clock name and the clock edges definition (waveform and period).

- **Path Group**

  The timing group that the path endpoint belongs to. This is usually the group defined by the destination clock, except for asynchronous timing checks (recovery/removal) which are grouped in the **`async_default`** timing group. User-defined groups can also appear here. They are convenient for reporting purpose.

- **Path Type**

  The type of analysis performed on this path.

  - **Max** indicates that the maximum delay values are used to calculate the data path delay, which corresponds to setup and recovery analysis.
  - **Min** indicates that the minimum delay values are used to calculate the data path delay, which corresponds to hold and removal analysis.

  This line also shows which corner was used for the report: **Slow** or **Fast**.

- **Requirement**

  The timing path requirement, which is typically:

  - One clock period for setup/recovery analysis.
  - 0ns for hold/removal analysis, when the startpoint and endpoint are controlled by the same clock, or by clocks with no phase-shift.

  When the path is between two different clocks, the requirement corresponds to the smallest positive difference between any source and destination clock edges. This value is overridden by timing exception constraints such as multicycle path, max delay and min delay.

  For more information on how timing path requirement is derived from the timing constraints, see *Timing Analysis* in the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

- **Data Path Delay**

  Accumulated delay through the logic section of the path. The clock delay is excluded unless the clock is used as a data. The type of delay corresponds to what the **Path Type** line describes.

- **Logic Levels**

  The number of each type of primitives included in the data section of the path, excluding the startpoint and the endpoint cells.

- **Clock Path Skew**

  The insertion delay difference between the launch edge of the source clock and the capture edge of the destination clock, plus clock pessimism correction (if any).

- **Destination Clock Delay (DCD)**

  The accumulated delay from the destination clock source point to the endpoint of the path.

  - For max delay analysis (setup/recovery), the minimum cell and net delay values are used
  - For min delay analysis (hold/removal), the maximum delay values are used.

- **Source Clock Delay (SCD)**

  The accumulated delay from the clock source point to the startpoint of the path.

  - For max delay analysis (setup/recovery), the maximum cell and net delay values are used.
  - For min delay analysis (hold/removal), the minimum delay values are used.

- **Clock Pessimism Removal (CPR)**

  The absolute amount of extra clock skew introduced by the fact that source and destination clocks are reported with different types of delay even on their common circuitry.

  After removing this extra pessimism, the source and destination clocks do not have any skew on their common circuitry.

  For a routed design, the last common clock tree node is usually located in the routing resources used by the clock nets and is not reported in the path details.

- **Clock Uncertainty**

  The total amount of possible time variation between any pair of clock edges.

  The uncertainty comprises the computed clock jitter (system and discrete), the phase error introduced by certain hardware primitives and any clock uncertainty specified by the user in the design constraints (**`set_clock_uncertainty`**).

  The user clock uncertainty is additive to the uncertainty computed by the Vivado IDE timing engine.

- **Total System Jitter (TSJ)**

  The combined system jitter applied to both source and destination clocks. To modify the system jitter globally, use the **`set_system_jitter`** constraint. The virtual clocks are ideal and therefore do not have any system jitter. For more information on System Jitter, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

- **Total Input Jitter (TIJ)**

  The combined input jitter of both source and destination clocks.

  To define the input jitter for each primary clock individually, use the **`set_input_jitter`** constraint. The Vivado IDE timing engine computes the generated clocks input jitter based on their master clock jitter and the clocking resources traversed. By default, the virtual clocks are ideal and therefore do not have any jitter.

  For more information on jitter, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

- **Discrete Jitter (DJ)**

  The amount of jitter introduced by hardware primitives such as MMCM or PLL.

  The Vivado IDE timing engine computes this value based on the configuration of these cells.

- **Phase Error (PE)**

  The amount of phase variation between two clock signals introduced by hardware primitives such as MMCM or PLL.

  The Vivado IDE timing engine automatically provides this value and adds it to the clock uncertainty

- **User Uncertainty (UU)**

  The additional uncertainty specified by the **`set_clock_uncertainty`** constraint.

  For more information on how to use this command, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

Additional lines can appear in the Timing Path Summary depending on the timing constraints, the reported path, and the target device:

- **Inter-SLR Compensation**

  The additional margin required for safely reporting paths that cross SLR boundaries in Xilinx 7 series SSI devices only.

- **Input Delay**

  The input delay value specified by the **`set_input_delay`** constraint on the input port. This line does not show for paths that do not start from an input port.

- **Output Delay**

  The output delay value specified by the **`set_output_delay`** constraint on the output port. This line does not show for paths that do not end to an output port.

- **Timing Exception**

  The timing exception that covers the path. Only the exception with the highest precedence is displayed, as it is the only one affecting the timing path requirement.

  For more information on timing exceptions and their precedence rules, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 6].

Timing Path Details

The second half of the report provides more details on the cells, pins, ports and nets traversed by the path. It is separated into three sections:

- **Source Clock Path**

  The circuitry traversed by the source clock from its source point to the startpoint of the datapath. This section does not exist for a path starting from an input port.

- **Data Path**

  The circuitry traversed by the data from the startpoint to the endpoint.

- **Destination Clock Path**

  The circuitry traversed by the destination clock from its source point to the datapath endpoint clock pin.

The Source Clock Path and Data Path sections work together. They are always reported with the same type of delay:

- max delay for setup/recovery analysis
- min delay for hold/removal analysis

They share the accumulated delay which starts at the data launch edge time, and accumulates delay through both source clock and data paths. The final accumulated delay value is called the *data arrival time*.

The destination clock path is always reported with the opposite delay to the source clock and data paths. Its initial accumulated delay value is the time when the data capture edge is launched on the destination clock source point. The final accumulated delay value is called the *data required time*.

The final lines of the report summarize how the slack is computed.

- For max delay analysis (setup/recovery)

  ```
  slack = data required time - data arrival time
  ```

- For min delay analysis (hold/removal)

  ```
  slack = data arrival time - data required time
  ```

Timing Path Details In Text Report

Figure 3-6, Timing Path Details in Text Report, shows an example of the Source Clock, Data and Destination Clock Paths in the text report. Because the path is covered by a simple period constraint of 5ns, the source clock launch edge starts at 0ns and the destination clock capture edge starts at 5ns.

```
  Location         Delay type                Incr(ns)  Path(ns)    Netlist Resource(s)
-------------------------------------------------------------------    -------------------
                   (clock wbClk rise edge)    0.000     0.000 r
  V20                                         0.000     0.000 r  wbClk
                   net (fo=0)                 0.000     0.000    wbClk
  V20              IBUF (Prop_ibuf_I_O)       0.764     0.764 r  wbClk_IBUF_inst/O
                   net (fo=1, routed)         1.901     2.665    wbClk_IBUF
  BUFGCTRL_X0Y1    BUFG (Prop_bufg_I_O)       0.093     2.758 r  wbClk_IBUF_BUFG_inst/O
                   net (fo=704, routed)       1.452     4.210    wbClk_IBUF_BUFG
  SLICE_X6Y33                                                r  ingressFifoWrEn_reg/C
-------------------------------------------------------------------    -------------------
  SLICE_X6Y33      FDRE (Prop_fdre_C_Q)       0.259     4.469 r  ingressFifoWrEn_reg/Q
                   net (fo=24, routed)        2.286     6.756    ingressLoop[2].ingressFifo/buffer_fifo/I1
  SLICE_X40Y16     LUT2 (Prop_lut2_I0_O)      0.043     6.799 r  ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.two_rd_addr_reg[9]_i_1__1/O
                   net (fo=39, routed)        0.327     7.126    ingressLoop[2].ingressFifo/buffer_fifo/do_read
  SLICE_X37Y16                                               r  ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]/CE
-------------------------------------------------------------------    -------------------

                   (clock bftClk rise edge)   5.000     5.000 r
  W17                                         0.000     5.000 r  bftClk
                   net (fo=0)                 0.000     5.000    bftClk
  W17              IBUF (Prop_ibuf_I_O)       0.674     5.674 r  bftClk_IBUF_inst/O
                   net (fo=1, routed)         1.787     7.461    bftClk_IBUF
  BUFGCTRL_X0Y0    BUFG (Prop_bufg_I_O)       0.083     7.544 r  bftClk_IBUF_BUFG_inst/O
                   net (fo=730, routed)       1.248     8.792    ingressLoop[2].ingressFifo/buffer_fifo/bftClk_IBUF_BUFG
  SLICE_X37Y16                                               r  ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]/C
                   clock pessimism            0.000     8.792
                   clock uncertainty         -0.035     8.757
  SLICE_X37Y16     FDCE (Setup_fdce_C_CE)    -0.201     8.556    ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]
-------------------------------------------------------------------
                   required time                        8.556
                   arrival time                        -7.126
-------------------------------------------------------------------
                   slack                                1.430
```

Figure 3-6: **Timing Path Details in Text Report**

Timing Path Details in Vivado IDE

The Timing Path Details in the Vivado IDE, as shown in Figure 3-7, Timing Path Details in Text Report, page 106, shows the same information as is shown in the text report, seen in Figure 3-6, Timing Path Details in Text Report.

Source Clock Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock wbClk rise edge)	(r) 0.000	0.000		
	(r) 0.000	0.000	Site: V20	wbClk
net (fo=0)	0.000	0.000		wbClk
			Site: V20	wbClk_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 0.764	0.764	Site: V20	wbClk_IBUF_inst/O
net (fo=1, routed)	1.901	2.665		wbClk_IBUF
			Site: BUFGCTRL_X0Y1	wbClk_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.093	2.758	Site: BUFGCTRL_X0Y1	wbClk_IBUF_BUFG_inst/O
net (fo=704, routed)	1.452	4.210		wbClk_IBUF_BUFG
			Site: SLICE_X6Y33	ingressFifoWrEn_reg/C

Data Path

Delay Type	Delay	Cumulative	Location	Logical Resource
FDRE (Prop_fdre_C_Q)	(r) 0.259	4.469	Site: SLICE_X6Y33	ingressFifoWrEn_reg/Q
net (fo=24, routed)	2.286	6.756		ingressLoop[2].ingressFifo/buffer_fifo/I1
			Site: SLICE_X40Y16	ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.two_rd_addr_reg[9]_i_1
LUT2 (Prop_lut2_I0_O)	(r) 0.043	6.799	Site: SLICE_X40Y16	ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.two_rd_addr_reg[9]_i_1
net (fo=39, routed)	0.327	7.126		ingressLoop[2].ingressFifo/buffer_fifo/do_read
			Site: SLICE_X37Y16	ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]/CE
Arrival Time		7.126		

Destination Clock Path

Delay Type	Delay	Cumulative	Location	Logical Resource
(clock bftClk rise edge)	(r) 5.000	5.000		
	(r) 0.000	5.000	Site: W17	bftClk
net (fo=0)	0.000	5.000		bftClk
			Site: W17	bftClk_IBUF_inst/I
IBUF (Prop_ibuf_I_O)	(r) 0.674	5.674	Site: W17	bftClk_IBUF_inst/O
net (fo=1, routed)	1.787	7.461		bftClk_IBUF
			Site: BUFGCTRL_X0Y0	bftClk_IBUF_BUFG_inst/I
BUFG (Prop_bufg_I_O)	(r) 0.083	7.544	Site: BUFGCTRL_X0Y0	bftClk_IBUF_BUFG_inst/O
net (fo=730, routed)	1.248	8.792		ingressLoop[2].ingressFifo/buffer_fifo/bftClk_IBUF_BUFG
			Site: SLICE_X37Y16	ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]/C
clock pessimism	0.000	8.792		
clock uncertainty	-0.035	8.757		
FDCE (Setup_fdce_C_CE)	-0.201	8.556	Site: SLICE_X37Y16	ingressLoop[2].ingressFifo/buffer_fifo/infer_fifo.next_rd_addr_reg[6]
Required Time		8.556		

Figure 3-7: **Timing Path Details in Vivado IDE**

The information on the path is displayed in five columns:

- **Location**

  Where the cell or port is placed on the device.

- **Delay Type**

  The unisim primitive and the particular timing arc followed by the path. In case of a net, it shows the fanout (`fo`) and its status. A net can be:

  - **Unplaced**

    The driver and the load are not placed.

  - **Estimated**

    The driver or the load or both are placed. A partially routed net is also reported as estimated.

  - **Routed**

    The driver and the load are both placed, plus the net is fully routed.

- **Incr(ns) (text report) / Delay (IDE report)**

  The value of the incremental delay associated to a unisim primitive timing arc or a net. It can also show of a constraint such as input/output delay or clock uncertainty.

- **Path(ns) (text report) / Cumulative (IDE report)**

  The accumulated delay after each segment of the path. On a given line, its value is the accumulated value from the previous + the incremental delay of the current line.

- **Netlist Resource(s) (text report) / Logical Resource (IDE report)**

  The name of the netlist object traversed.

Each incremental delay is associated to an edge sense:

- **`r`** (rising), or
- **`f`** (falling)

The initial sense of the edge is determined by the launch or capture edge used for the analysis. It can be inverted by any cell along the path, depending on the nature of the timing arc. For example, a rising edge at the input of an inverter becomes a falling edge on the output.

The edge sense can be helpful in identifying that an overly-tight timing path requirement comes from a clock edge inversion along the source or destination clock tree.

XILINX

Chapter 4

Design Closure Techniques

Introduction to Design Closure Techniques

This chapter discusses techniques for timing closure in the Xilinx® Vivado® Integrated Design Environment (IDE) including:

- Checking Constraints and Sources
- Increasing Tool Effort
- Floorplanning
- Modifying Routing

Checking Constraints and Sources

Ensure that design and timing constraints are reasonable.

- Be sure that you have a good netlist.

  If you are using Vivado Synthesis, add synthesis-specific timing constraints to a synthesis XDC file. Synthesis is timing-driven, and optimizes the logic to meet timing.

- Manually review the clock trees in the Schematic window or the Clock Networks report.

  Be sure that the clock trees are reasonable. Designs can have a large clock skew when one BUFG drives a second BUFG or a LUT. The extra clock skew can lead to small to no design margin for meeting setup checks, or to a large amount of hold fixing, which will increase the routing resource utilization and potentially the congestion level. The placer DRCs issue warnings for some clock tree issues.

- Consider the clocking resources when laying out the pinout and floorplanning the design.

  The clock regions in Xilinx® 7 series FPGA devices support twelve global clocks per region. There are additional limitations on the placement of clock trees. For more information on Xilinx 7 Series FPGA Clocking, see *7 Series Clocking Resources Guide*

(UG472) [Ref 9].

- Be sure that the clock periods are the ones the design needs to meet.

  If you are overconstraining the design, the tools attempt to meet artificially tight timing constraints at the cost of runtime. Overconstraining can lead to timing failures and higher power consumption.

  If the design is failing to meet artificially tight constraints, try the real constraints. You can modify timing constraints in the Vivado IDE without changing the placement and routing. Rerun **`report_timing_summary`** with the real clock periods.

- Determine if the failing paths are multicycle paths or paths not functionally active.

  The Vivado IDE times all logical paths of your design, including the ones between asynchronous clocks unless you add appropriate timing exceptions. This is a change from ISE® Design Suite and UCF. Some paths may never be active due to the structure of the control logic. You can use the **`set_false_path`** constraint to ignore slack computation on them.

  Other control structures (for example, a clock enable driven by a state machine) generate multicycle paths. If data has multiple clock periods to get from the source to the destination, enter Multicycle Timing constraints. In XDC, setup and hold multicycle delays are entered separately.

- Rerun timing after Implementation with the improved timing constraints. You do not need to rerun implementation to view how changing the timing graph changes design timing. Review the next set of timing paths and continue to refine the timing constraints as needed.
- If you are still not meeting timing after modifying the timing constraints, rerun implementation. Implementation is timing-driven and can now focus on the real problem areas. If the original design had high Total Hold Slack (THS) in the router, you must rerun implementation.

TIP: *Use **Save Constraints As** to create a new constraint set, preserving your original constraints.*

- Review the timing path.

  Ensure that clock skew and jitter are reasonable.

- Review the logic.

  What is the logic delay compared to the period? If logic delay is a high percentage of the period you must resynthesize the design to reduce logic delay.

- Review the Route Delay

  If the route delay is a high percentage of the period, try reducing route delay. Two ways to try to reduce route delay are:

  - Rewrite the RTL.

    If you see high fanout nets, consider replicating the drivers. Take logic placement into consideration when re-factoring the high fanout net. Each driver should communicate with localized loads rather than loads spread out across the device.

    Apply synthesis attributes such as `DONT_TOUCH` or `KEEP` to prevent register pruning on replicated registers.

  - Floorplan.

    Consider floorplanning if route delay is a high percentage of the timing path. The goal is to improve the timing of the critical paths by reducing route delay. Floorplanning does not change the logic that makes up the critical path. Review how the blocks in the design interconnect while floorplanning.

    You may want to restructure the RTL to better work inside the device. The synthesis tools do not know about placement. When they replicate high fanout nets, they generally do not take placement into account.

Manually replicating and rewiring can lead to a better design. For example, if you have two memory interfaces that go on opposite corners of the chip, you might want to replicate the control signals in the RTL source. You might need to use synthesis attributes to stop optimization of logically equivalent registers.

Increasing Tool Effort

Ask the tools to work harder. Sometimes you can close timing on a design merely by assigning more CPU cycles to implementation.

The **Vivado Implementation Defaults** strategy balances run time and performance. Some run time expensive algorithms are not used. To turn on these algorithms, and physical synthesis, use the **Performance_Explore** strategy. For the largest parts (using SSI technology) consider **Performance_ExploreSLLs**.

There are other strategies to help with faster run time or congestion. The strategies with a SLL suffix are tuned for the parts using SSI technology.

Run the **Flow > Create Runs** command to create and launch multiple runs using the different strategies.

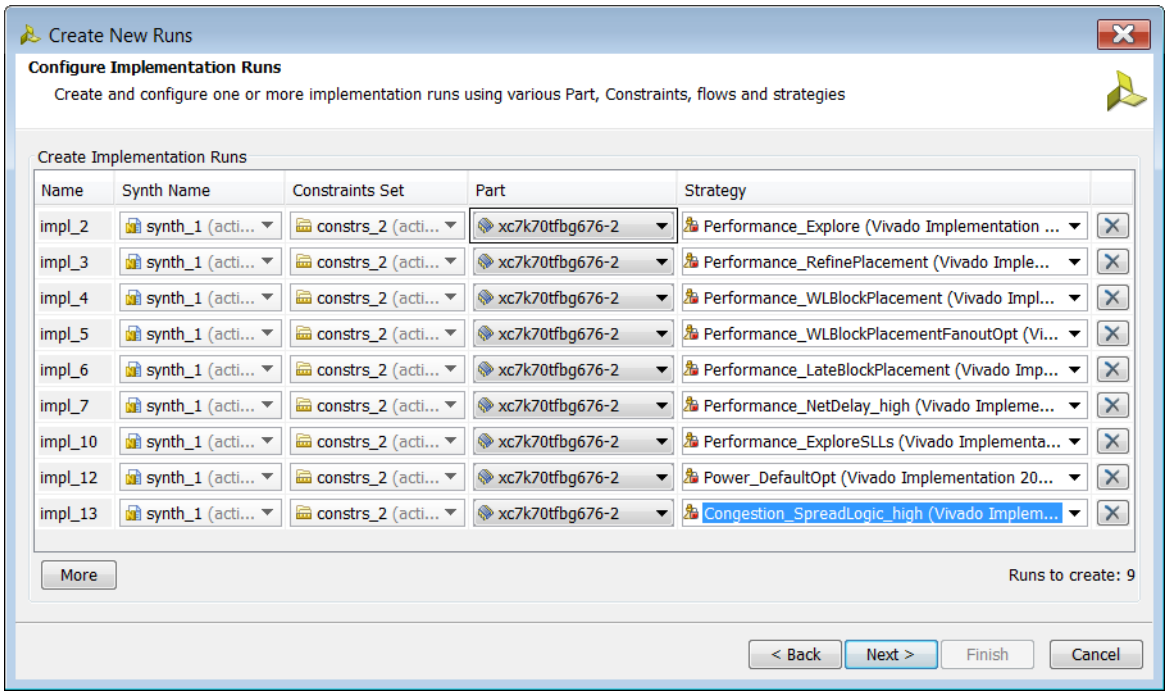


Figure 4-1: **Multiple Runs**

Review the implementation results to find the strategy that works best for the design. Re-use the strategy for later runs.

TIP: *The optimal strategy can change between designs and software releases.*

Floorplanning

This section discusses Floorplanning and includes:

About Floorplanning

Floorplanning can help a design meet timing. Xilinx recommends that you floorplan when a design does not meet timing consistently, or has never met timing.

Floorplanning is also helpful when you are working with design teams, and consistency is most important.

Floorplanning can improve the setup slack (TNS, WNS) by reducing the average route delay. During implementation, the timing engine works on resolving the worst setup violations and all the hold violations. Floorplanning can only improve setup slack.

Manual floorplanning is easiest when the netlist has hierarchy. Design analysis is much slower when synthesis flattens the entire netlist. Set up synthesis to generate a hierarchical netlist. For Vivado synthesis use:

- **synth_design -flatten_hierarchy rebuilt**

  or

- The **Vivado Synthesis Defaults** strategy

Large hierarchical blocks with intertwined logical paths can be difficult to analyze. It is easier to analyze a design in which separate logical structures are in lower sub-hierarchies. Consider registering all the outputs of a hierarchical module. It is difficult to analyze the placement of paths that trace through multiple hierarchical blocks.

Understanding Floorplanning Basics

Not every design will always meet timing. You may have to guide the tools to a solution. Floorplanning allows you to guide the tools, either through high-level hierarchy layout, or through detailed gate placement.

You will achieve the greatest improvements by fixing the worst problems or the most common problems. For example if there are outlier paths that have significantly worse slack, or high levels of logic, fix those paths first. The **Tools > Timing > Create Slack Histogram** command can provide a view of outlier paths. Alternatively, if the same timing endpoint appears in several negative slack paths, improving one of the paths might result in similar improvements for the other paths on that endpoint.

Consider floorplanning to increase performance by reducing route delay or increasing logic density on a non-critical block. Logic density is a measure of how tightly the logic is packed onto the chip.

Floorplanning can help you meet a higher clock frequency and improve consistency in the results.

There are multiple approaches to floorplanning, each with its advantages and disadvantages.

Detailed Gate-Level Floorplanning

Detailed gate-level floorplanning involves placing individual leaf cells in specific sites on the device.

Advantages of Detailed Gate-Level Floorplanning

- Detailed gate-level floorplanning works with hand routing nets.
- Detailed gate-level floorplanning can extract the most performance out of the device.

Disadvantages of Detailed Gate-Level Floorplanning

- Detailed gate-level floorplanning is time consuming.
- Detailed gate-level floorplanning requires extensive knowledge of the device and design.
- Detailed gate-level floorplanning may need to be redone if the netlist changes.

RECOMMENDED: *Use detailed gate-level floorplanning as a last resort.*

Information Re-Use

Re-use information from a design that met timing. Use this flow if the design does not consistently meet timing. To re-use information:

1. Open two implementation runs:
   a. One for a run that *is* meeting timing.
   b. One for a run that is *not* meeting timing.

TIP: *On a computer with multiple monitors, select* ***Open Implementation in New Window*** *to open a design in a new window.*

2. Look for the differences between the two designs.
   a. Identify some failing timing paths from **`report_timing_summary`**.
   b. On the design that is meeting timing, run **`report_timing`** in **`min_max`** mode to time those same paths on the design that meets timing.
3. Compare the timing results:
   a. Clock skew
   b. Datapath delay
   c. Placement
   d. Route delays
4. If there are differences in the amount of logic delay between path end points, revisit the synthesis runs.

Review I/O and Cell Placement

Review the placement of the cells in the design. Compare two I/O reports to review the I/O placement and I/O standards. Make sure all the I/Os are placed. A simple search finds all

I/Os without fixed placement as shown in Figure 4-2, I/O Is Not Fixed.

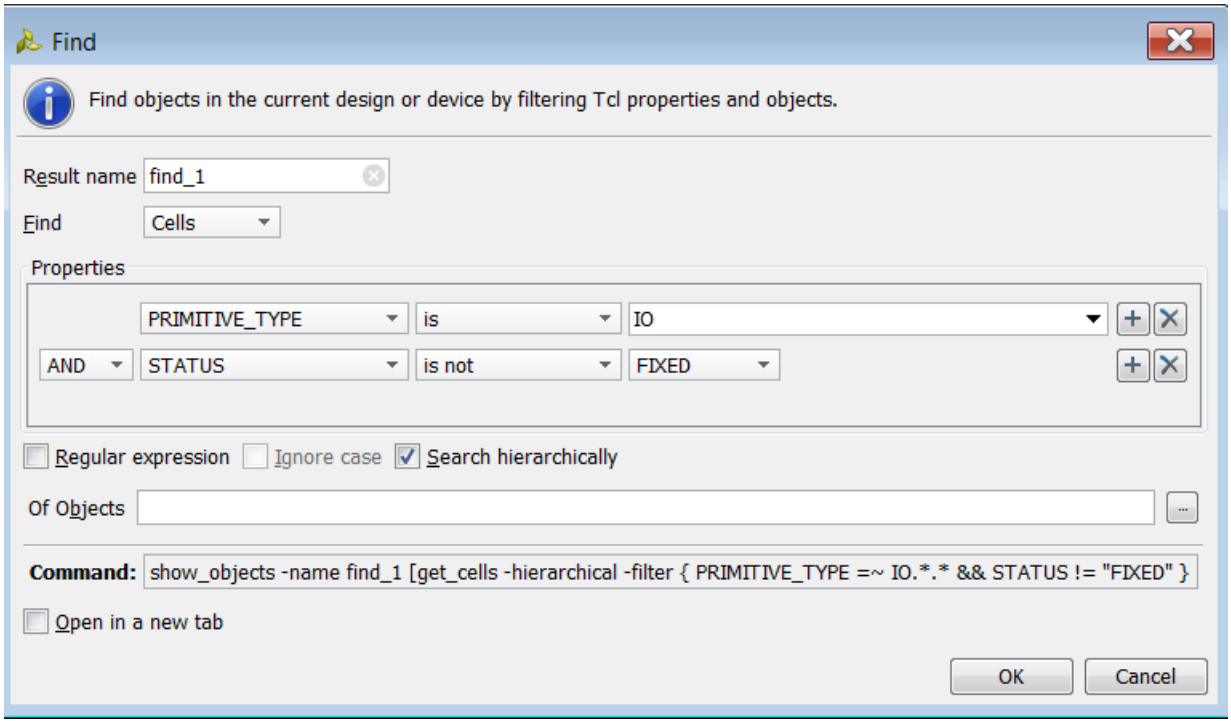


Figure 4-2: **I/O Is Not Fixed**

If clock skew has changed between the runs, consider re-using the clock primitive placement from the run that met timing. The Clock Utilization Report lists the placement of the clock tree drivers, as shown in Figure 4-3, Clock Locations.

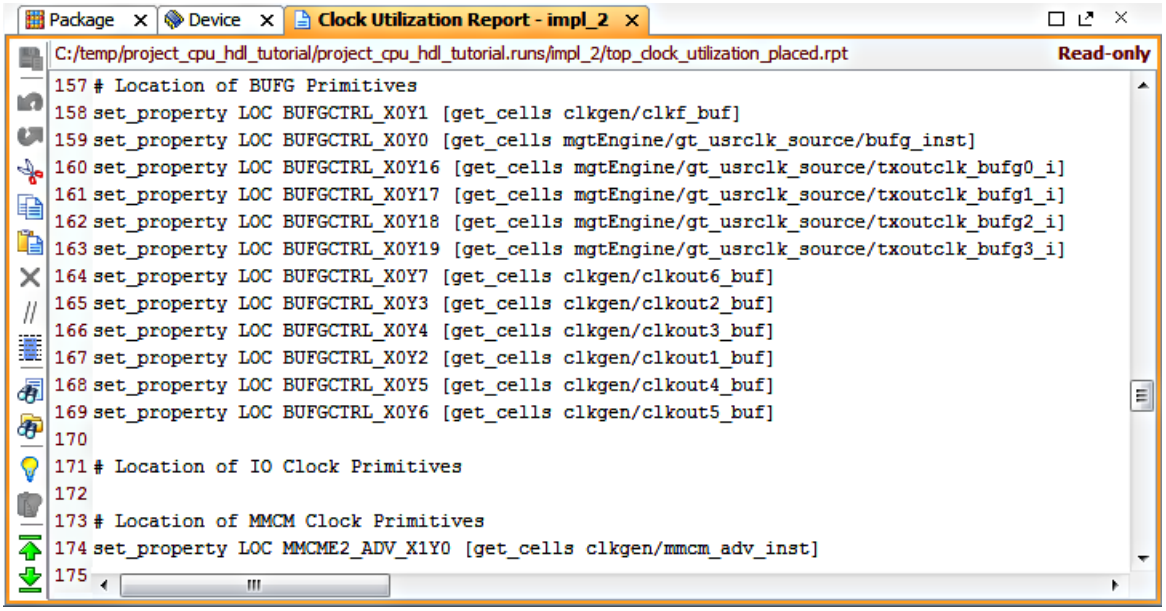


Figure 4-3: **Clock Locations**

The LOC constraints can easily be copied into your XDC constraints file.

Many designs have met timing by reusing the placement of the Block RAMs and DSPs. Select **Edit > Find** to list the instances.

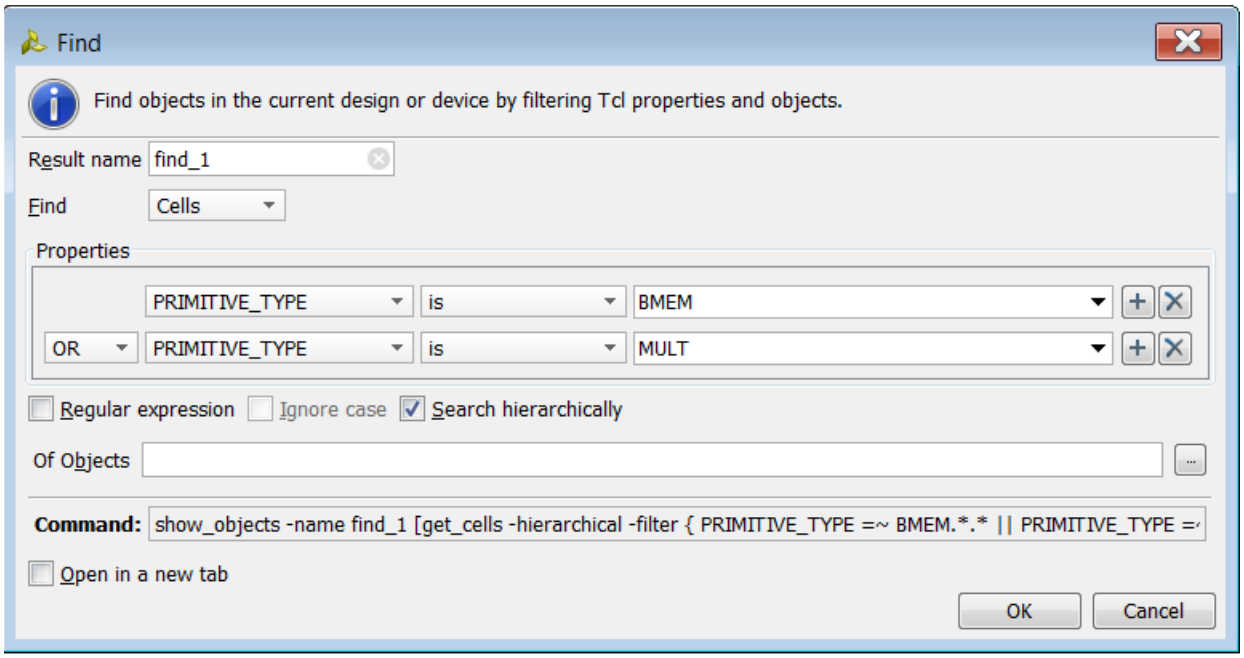


Figure 4-4: **DSP or RAM**

Adding Placement Constraints

Fix the logic to add the placement constraints to your XDC.

1. Select the macros from the find results.
2. Right click and select **Fix Cells**.

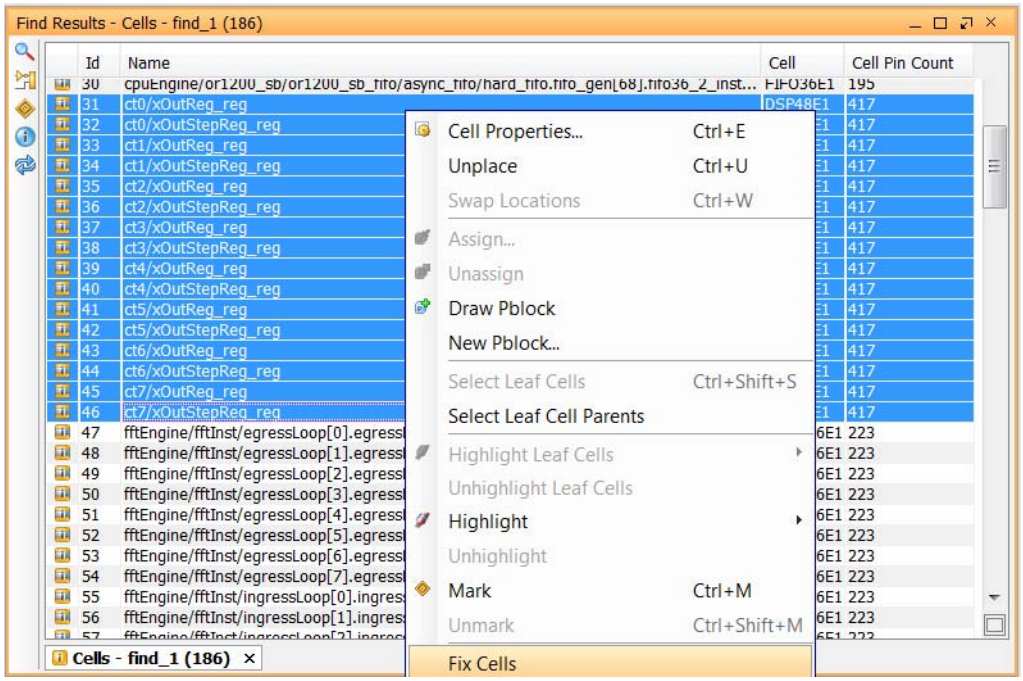


Figure 4-5: **Selecting the Logic to Fix**

RECOMMENDED: *Analyze the placement based on hierarchy name and highlight before fixing the placement.*

Re-Using Placement

It is fairly easy to re-use the placement of:

- I/Os
- Global Clock Resources
- BlockRam macros
- DSP macros

Re-using this placement helps to reduce the variability in results from one netlist revision to the next. These primitives generally have stable names. The placement is usually easy to maintain.

TIP: *Do not reuse the placement of general slice logic. Do not re-use the placement for sections of the design that are likely to change.*

Floorplanning Techniques

Consider gate-level floorplanning for a design that has never met timing, and in which changing the netlist or the constraints are not good options.

RECOMMENDED: *Try hierarchical floorplanning before considering gate level floorplanning.*

Hierarchical Floorplanning

Hierarchical floorplanning allows you to place one or more levels of hierarchy in a region on the chip. This region provides guidance to the placer at a global level, and the placer does the detailed placement.

Hierarchical floorplanning has the following advantages over gate-level floorplanning:

- Hierarchical floorplan creation is fast compared to gate-level floorplanning. A good floorplan can improve timing. The floorplan is resistant to design change.
- The level of hierarchy acts as a container for all the gates. It will generally work if the netlist changes.

In hierarchical floorplanning:

- Identify the lower levels of hierarchy that contain the critical path.
- Use the top level floorplan to identify where to place them.
- Implementation places individual cells.
- Has comprehensive knowledge of the cells and timing paths.
- Generally does a good job of fine grain placement.

Manual Cell Placement

Manual cell placement can obtain the best performance from a device. When using this technique, designers generally use it only on a small block of the design. They may hand place a small amount of logic around a high speed I/O interface, or hand place Block RAMs and DSPs. Manual placement can be slow.

All floorplanning techniques can require significant engineering time. They may require floorplan iterations. If any of the cell names change, the floorplan constraints must be updated.

When floorplanning, you should have an idea of final pinout. It is useful to have the I/Os fixed. The I/Os can provide anchor points for starting the floorplan. Logic that communicates to I/Os migrates towards the fixed pins.

TIP: *Place blocks that communicate with I/Os near their I/Os. If the pinout is pulling a block apart, consider pinout or RTL modification.*

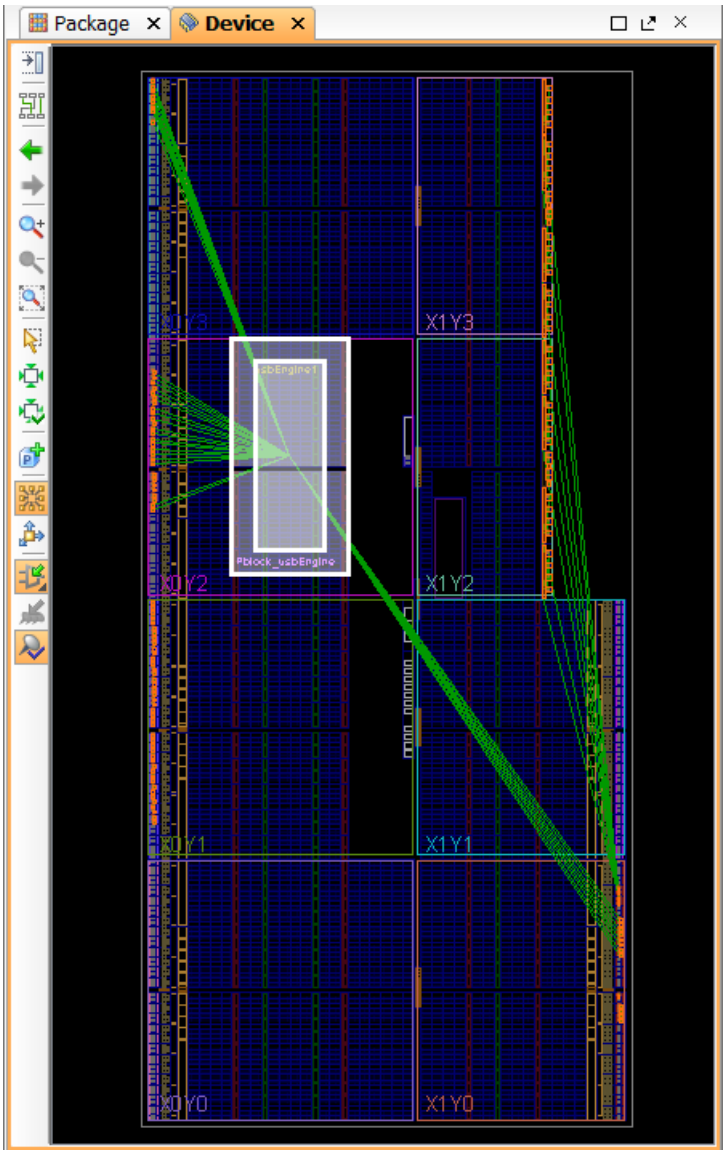


Figure 4-6: **I/O Components Pulling Design Apart**

The floorplan shown in Figure 4-6 might not help timing. Consider splitting the block apart, changing the source code, or constraining only the Block RAMs and DSPs. Also consider unplacing I/O registers if external timing requirements allow.

The Pblock mentioned in this section is represented by the XDC constraints:

```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list
usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add {SLICE_X8Y105:SLICE_X23Y149}
resize_pblock [get_pblocks Pblock_usbEngine] -add {DSP48_X0Y42:DSP48_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB36_X0Y21:RAMB36_X1Y29}
```

The first line creates the Pblock. The second line (**`add_cells_to_pblock`**) assigns the level of hierarchy to the Pblock. There are four resource types (SLICE, DSP48, RAMB18, RAMB36) each with its own grid. Logic that is not constrained by a grid can go anywhere in the device. To constrain just the Block RAMs in the level of hierarchy, disable the other Pblock grids.

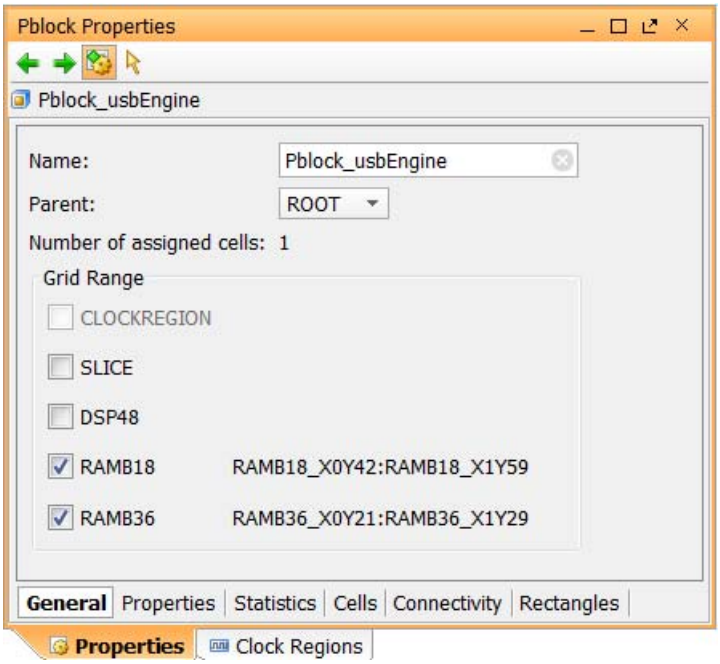


Figure 4-7: **Pblock Grids**

The resulting XDC commands define the simplified Pblock:

```
create_pblock Pblock_usbEngine
add_cells_to_pblock [get_pblocks Pblock_usbEngine] [get_cells -quiet [list
usbEngine1]]
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB18_X0Y42:RAMB18_X1Y59}
resize_pblock [get_pblocks Pblock_usbEngine] -add {RAMB36_X0Y21:RAMB36_X1Y29}
```

The Block RAMs are constrained in the device, but the slice logic is free to be placed anywhere on the device.

TIP: *When placing Pblocks, be careful not to floorplan hierarchy in such a manner that it crosses the central config block.*

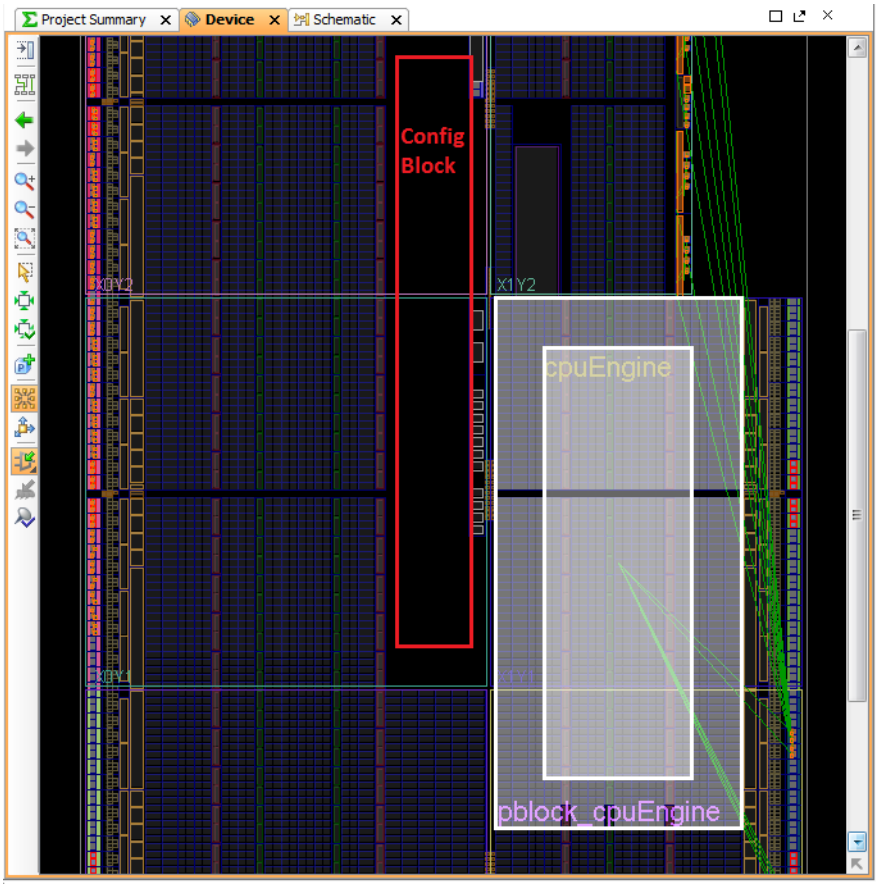


Figure 4-8: **Avoiding the Config Block**

Using Pblock-Based Floorplanning

When you integrate RTL into a design, it helps to visualize the design inside the device. Graphically seeing how the blocks interconnect between themselves and the I/O pinout after synthesis helps you to understand your design.

To view the interconnect, generate a top level floorplan using Pblocks on upper levels of hierarchy. To break apart the top level RTL into Pblocks, select **Tools > Floorplanning > Auto Create Pblocks**.

To place the blocks in the device, select **Tools > Floorplanning > Place Pblocks**. The tool sizes the Pblocks based on the slice count and target utilization

Pblocks can be more than one hundred percent full during analysis, but not during implementation. Overfilling the Pblock makes them smaller on the device. This is a useful technique for getting an overview of the relative size of your design top-level blocks, and how they will occupy the device.

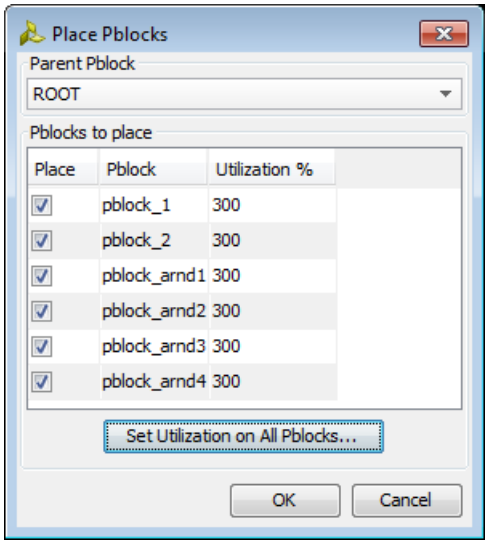


Figure 4-9: **Place Pblocks Utilization**

Top-Level Floorplan

The top-level floorplan shows which blocks communicate with I/Os (green lines). Nets connecting two Pblocks are bundled together. The bundles change size and color based on the number of shared nets. Two top-level floorplans are shown in Figure 4-10, Data Path Top Level Floorplan, and Figure 4-11, Control Path Floorplan.

The Data Path Top Level Floorplan shows how the data flows between the top-level blocks of the design. Each block communicates only with two neighbors. The green lines show well-placed I/Os that communicate with a single block.

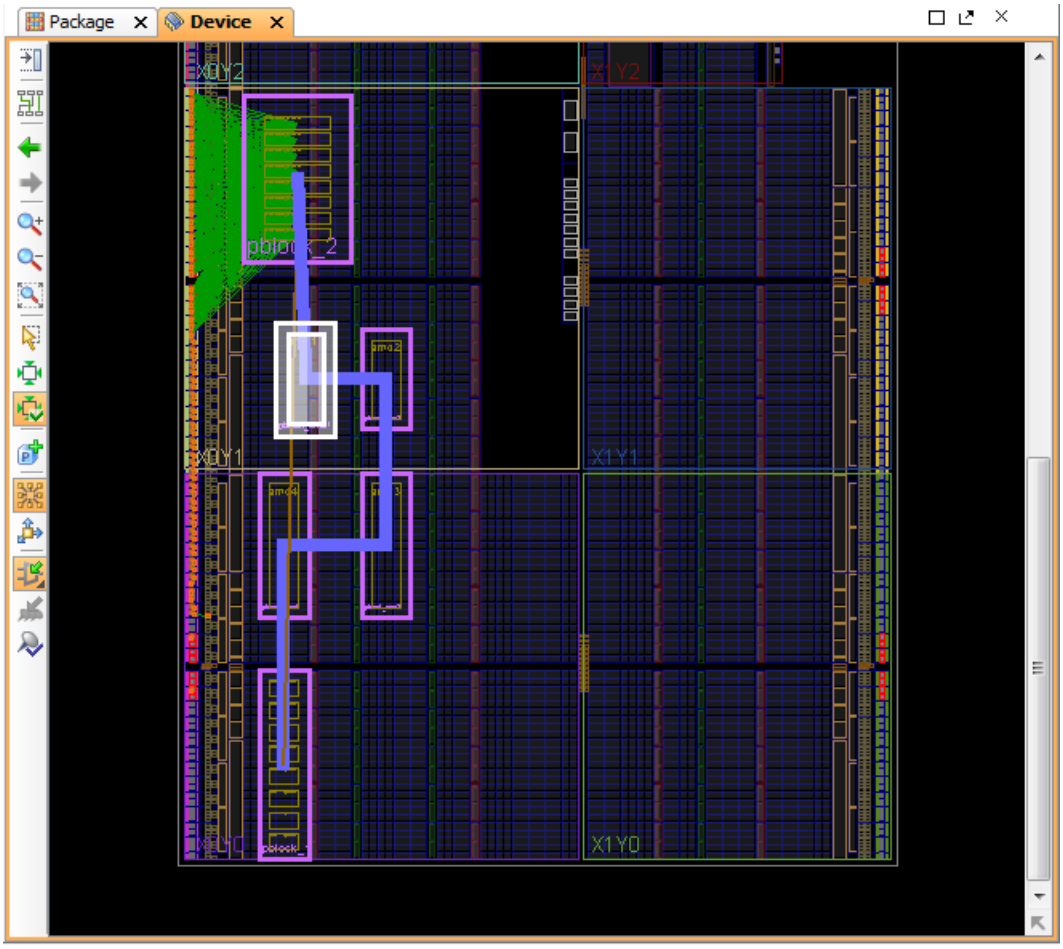


Figure 4-10: **Data Path Top Level Floorplan**

The Control Path Floorplan displays a design in which all the blocks communicate with a central block. The largest connection is between the central block and the block in the bottom right. The central block must spread out around the design to communicate with all the other loads.

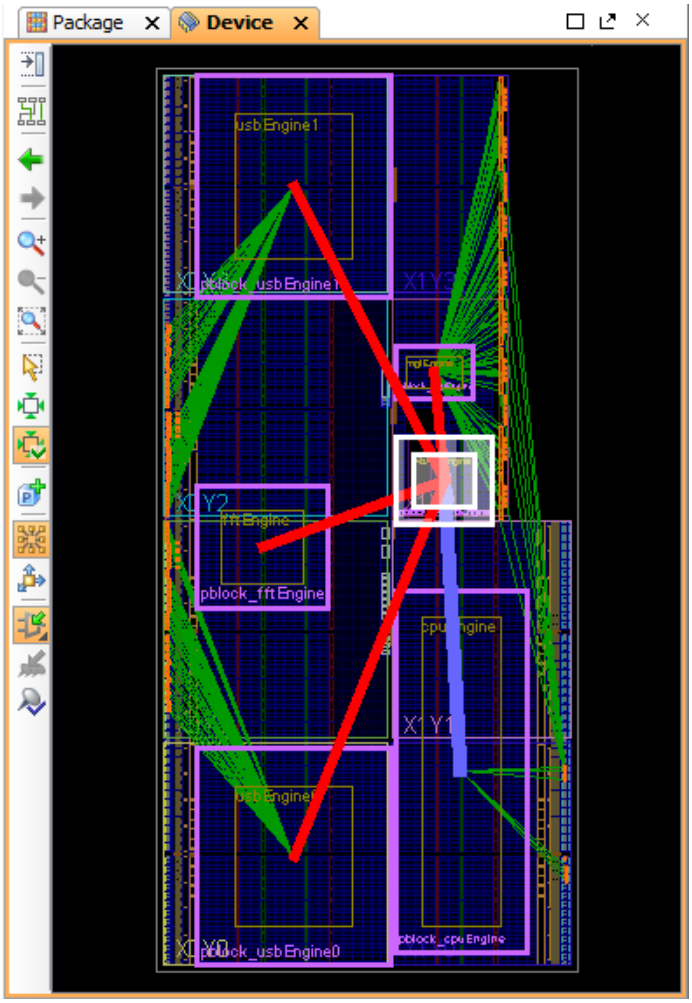


Figure 4-11: **Control Path Floorplan**

Reviewing the Floorplan

Consider device resources when reviewing the floorplan. The Pblock sizing does not take into account specialized device resources such as:

- Block RAM
- DSP48s
- MGTs
- ClockBuffers

TIP: *Review the blocks with the floorplan and utilization in mind.*

Locking Specific Logic to Device Sites

You can place cells on specific locations on the FPGA device, such as placing all the I/O ports on a Xilinx 7 series FPGA design. Xilinx recommends that you place the I/Os **before** attempting to close timing.

The I/O placement can impact the cell placement in the FPGA fabric. Hand placing other cells in the fabric can help provide some consistency to clock logic and macro placement, with the goal of more consistent implementation runs.

Table 4-1: **Constraints Used to Place Logic**

Constraint	Use	Notes
LOC	Places a gate or macro at a specific site.	SLICE sites have subsites called BEL sites.
BEL	Specifies the subsite in the slice to use for a basic element.	

Fixed and Unfixed Cells

Fixed and Unfixed apply to placed cells. They describe the way in which the Vivado tools view placed cells in the design.

For more information about Fixed and Unfixed Cells, refer to the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 7].

RECOMMENDED: *After the I/Os are placed, use a hierarchical Pblock floorplan as a starting point for user-controlled placement. Hand placing logic should be used when Pblocks have been found not to work.*

Floorplanning With Stacked Silicon Interconnect (SSI) Devices

There are extra considerations for Stacked Silicon Interconnect (SSI) parts. The SSI parts are made of multiple Super Logic Regions (SLRs), joined by an interposer. The interposer connections are called Super Long Lines (SLLs). There is some delay penalty when crossing from one SLR to another.

Keep the SLRs in mind when structuring the design, generating a pinout, and floorplanning. Minimize SLL crossings by keeping logic cells of critical timing paths inside a single SLR.

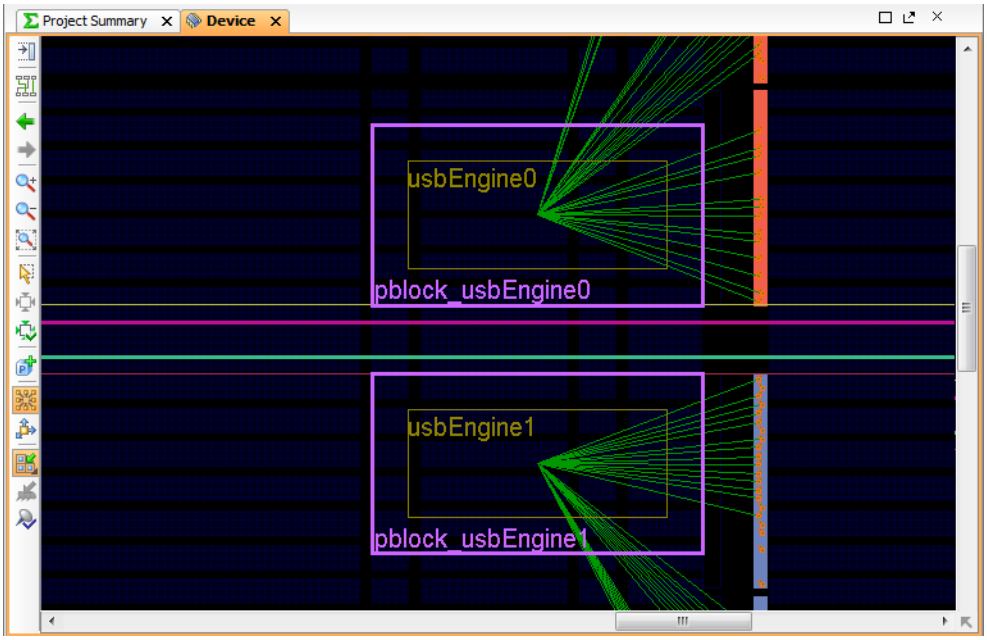


Figure 4-12: **Minimize SLR Crossings**

The I/Os must be placed in the same SLR as the relevant I/O interface circuitry. You must also carefully consider clock placement when laying out logic for SSI parts.

RECOMMENDED: *Let the placer try an automatic placement of the logic into the SSI parts before doing extensive partitioning. Analyzing the automatic placement may suggest floorplanning approaches you were not considering.*

Modifying Routing

You might occasionally want to modify routing. Most often, this involves a design with a few small timing failures, and in which other, easier approaches to timing closure have failed. The Vivado Design Suite allows you to modify and fix routes.

For more information, see "Modifying Placement, Routing and Logic" in the *Vivado Design Suite User Guide: Implementation* (UG904) [Ref 7].

Appendix A

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

The following Vivado® Design Suite documents provide supplemental material useful with this guide.

1. *Vivado Design Suite User Guide: Using the Vivado IDE* (UG893)
2. *Vivado Design Suite User Guide: Using Tcl Scripting* (UG894)
3. *Vivado Design Suite Tcl Command Reference Guide* (UG835)
4. *Vivado Design Suite User Guide: System-Level Design Entry* (UG895)
5. *Design Methodology Handbook for Xilinx FPGAs* (UG949)
6. *Vivado Design Suite User Guide: Using Constraints* (UG903)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. *Vivado Design Suite User Guide: Power Analysis and Optimization* (UG907)
9. *7 Series Clocking Resources Guide* (UG472)

Other Vivado Design Suite Documents

- *Vivado Design Suite Tutorial: Design Flows Overview* (UG888)
- *Vivado Design Suite Video Tutorials* (http://www.xilinx.com/training/vivado/index.htm)
- *All Vivado Design Suite Documentation* (www.xilinx.com/support/documentation/dt_vivado2014-1.htm)

Please Read: Important Legal Notices