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

© Copyright 2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/11/13</td>
<td>2012.4</td>
<td>• Updated Report Clock Interaction in Chapter 1, and added Report Datasheet in Chapter 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minor formatting and editing changes throughout.</td>
</tr>
<tr>
<td>11/16/2012</td>
<td>2012.3</td>
<td>• Minor formatting and editing changes throughout.</td>
</tr>
<tr>
<td>06/15/12</td>
<td>2012.2</td>
<td>• Initial Xilinx release.</td>
</tr>
</tbody>
</table>
# Table of Contents

Revision History ......................................................... 2

## Chapter 1: Design Analysis

### Introduction to Design Analysis ........................................ 6

### Reports and Messages .................................................. 6
  - Managing Messages ..................................................... 7

### Creating Reports ...................................................... 9
  - Report Sources .......................................................... 9
  - Generated Reports .................................................... 10
  - Vivado Synthesis Report .............................................. 11
  - Utilization Report .................................................... 11
  - Vivado Implementation Log .......................................... 12
  - IO Report ................................................................. 13
  - Clock Utilization Report ............................................. 14
  - Control Sets Report .................................................. 15
  - DRC Report ............................................................. 15
  - Power Report ........................................................... 16
  - Route Status Report .................................................. 16
  - WebTalk Report ....................................................... 16
  - On Demand Reports ................................................... 16
  - Report DRC ............................................................. 18
  - Elaborated Design ..................................................... 18
  - Synthesized Design and Implemented Design ....................... 18
  - Report Noise ........................................................... 20
  - Power Report ........................................................... 22

### Report Timing Summary ................................................ 25
  - Report Timing Summary Dialog Box .................................. 25
  - Details of the Timing Summary Report ................................ 33

### Report Clock Networks ................................................ 44

### Report Clock Interaction .............................................. 45
  - Report Clock Interaction Dialog Box .................................. 45
  - Details of the Clock Interaction Report ............................. 49

### Report Pulse Width .................................................... 53
  - Per Session Reports .................................................. 53

### Report Timing ........................................................... 54
  - Running Report Timing ................................................ 54
  - Targets Tab ............................................................. 56
  - Options Tab ............................................................. 57
  - Advanced Tab .......................................................... 59
  - Timer Settings Tab .................................................... 59
Chapter 1

Design Analysis

Introduction to Design Analysis

This chapter provides an introduction to:

- Understanding and analyzing designs in the Vivado™ Design Suite, including:
  - Reports and Messages
  - Design Rules Check (DRC) structure
- Understanding the design as it is implemented into the Xilinx FPGA device
- Analyzing the timing of the design
- Analyzing the results of the Vivado placer and the Vivado router
- Techniques for improving the design to meet its timing goals

For more information about managing windows and using the Vivado IDE, see the Vivado Design Suite User Guide: Using Vivado IDE (UG893) at the location cited in Appendix A, Additional Resources.

Reports and Messages

The Vivado tool 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 the tool at key steps in the design flow, and summarize specific information about the design. Messages are generated automatically by the Vivado tool 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)
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.

**Managing Messages**

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

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

There are two types of messages in the tool:

- Messages stored on disk
- Messages stored in memory

In the Messages window, the Vivado Integrated Design Environment (IDE) groups messages by the action that created the message. Use the command buttons on the toolbar menu to group the messages by message ID or file.

Some messages include hyperlinks to a file or a design element to help in debugging. Click the link to view the source. 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. The message ID identifies different messages, allowing them to be grouped and sorted. 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. They provide details of the design or process, but require no action on your part.
Table 1-1: Message Severities

<table>
<thead>
<tr>
<th>Icon</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon]</td>
<td>Status</td>
<td>Communicates general status of the design processing.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Info</td>
<td>General status of the process and feedback regarding design processing.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Warning</td>
<td>Design results may be sub-optimal because constraints or specifications may not be applied as intended.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Critical Warning</td>
<td>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.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Error</td>
<td>An issue that renders design results unusable and cannot be resolved without user intervention. The design flow stops.</td>
</tr>
</tbody>
</table>

You can filter messages by severity to enable or disable the display of specific message types. Click the checkbox next to a message severity in the header of the Messages window to enable or disable the display.

You can also change the severity of a specific message ID. You may reduce the severity of a message you do not believe is critical, or elevate a message you think demands more attention. To increase or reduce 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) at the location cited in Appendix A, Additional Resources.

Messages Created During Design Load

Messages are returned while loading one of the following designs:

- Elaborated Design
- Synthesized Design
- Implemented Design

The Message window reflects the messages from the design is loaded into memory. If there is a Critical Warning or an Error during design load, the Vivado IDE may automatically open the Messages window and filter out the other messages.
Creating Reports

Specific examples of reports include:

- Generated Reports
- Vivado Synthesis Report
- Control Sets Report
- DRC Report
- Route Status Report
- Timing Summary Report
- WebTalk Report

Report Sources

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.
- Other reports are generated when a design is loaded into memory.

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.
Generated Reports

Generated Reports include:

• Reports Generated During Synthesis
• Reports Generated During Placement
• Reports Generated During Route
• Reports Generated During Bitstream Generation

Reports Generated During Synthesis

Reports generated during synthesis include:

• Utilization Report (Post Synthesis)

Reports Generated During Placement

Reports generated during placement include:

• Vivado Implementation Log
• IO Report
• Clock Utilization Report
• Utilization Report (Post Placement)
• Control Sets Report

Reports Generated During Route

Reports generated during route include:

• Vivado Implementation Log
• WebTalk Report
• DRC Report
• Power Report
• Route Status Report
• Timing Summary Report

Reports Generated During Bitstream Generation

Output from bitstream generation is written to the Vivado Implementation Log file. The only report generated during bitstream generation is the WebTalk Report.
Vivado Synthesis Report

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.

Utilization Report

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, including:
  ° BUFGCTRL
  ° BUFR
  ° BUFHCE
  ° MMCME2_ADV
  ° PLL2_ADV
• Specific Device Resources, including:
  ° 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.

**Vivado 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.
IO Report

The IO Report replaces the ISE PAD file. The IO Report lists:

- Pin Number
  All the pins in the device
- Signal Name
  The name of the user IO 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
- IO Standard
  The IO standard for the User IO
  An asterisk (*) indicates that it is the default. This differs from the IO Ports window of the Vivado IDE.
- IO Bank Number
  The IO 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 IO Bank next to the given port

• Signal Integrity
  The Off-Chip Termination setting

Clock Utilization Report

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. 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.

**Control Sets Report**

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

**DRC Report**

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.
Creating Reports

Power Report

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.

Route Status Report

The Route Status Report displays:

- The number of nets in the design
- The number of nets that do not use routing resources outside of a tile, including:
  - CLB
  - BlockRam
  - IO Pad
- The number of nets that do need routing resources

For a design to be fully routed, this report should show:

```
  # of unrouted nets........... : 0 :
```

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:


On Demand Reports

To generate other reports, you must first load the design in memory. Commonly used reports are available from the Flow Navigator.
Elaborated Design

- Report DRC
- Report Noise

Synthesized and Implemented Design

- Report Timing Summary
- Report Clock Networks
- Report Clock Interaction
- Report DRC
- Report Noise
- Report Utilization
- Report Power

You can also select **Tools > Timing** to generate timing-related reports. For example, when a Synthesized Design is loaded in memory, you have access to:

- Report Timing
- Report Timing Summary
- Report Pulse Width
- Report Clock Interaction
- Report Datasheet

The following sections give more information on each of the reports, including:

- How to run the report.
- The information it provides.

*TIP: Not all reports are useful at all steps in the flow. RTL Analysis supports only a subset of the reports.*
Report DRC

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

Elaborated Design

The tool checks for DRCs related to IO and Clock Placement. The RTL netlist typically does not have all the IO 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 IO, 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 IO 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 1-3 shows the Vivado IDE graphical user interface form of Report DRC.

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.

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
```
Creating Reports

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 Noise

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.

![Run SSN Analysis](image)

*Figure 1-5: 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 on www.xilinx.com/support

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

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

Report Power is available when a Synthesized Design or anImplemented Design is open. The 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 1-7: Report Power
Analyzing the Power Report

Use the Report Power dialog box, shown in Figure 1-7, page 23, to analyze power based on:

- Settings
- Power total
- By hierarchy
- Per Voltage rail
- By block type

For more information on the power report and analyzing the results, see Power Analysis and Optimization (UG907) at the location cited in Appendix A, Additional Resources.

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 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.
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:

- The Flow Navigator, under Synthesis or Implementation, or
- **Tools > Timing > Report Timing Summary** in the main menu.

The equivalent Tcl command is `report_timing_summary`.

For more information on the options of this command, see the *Vivado Design Suite Tcl Command Reference Guide* (UG835) at the location cited in Appendix A, Additional Resources.

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 constraints.

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:

- **Report Timing Summary Options Tab**
- **Report Timing Summary 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 will include hyperlinks allowing 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`
Report Timing Summary Options Tab

Figure 1-9 shows the Options tab of the Report Timing Summary dialog box.

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, recovery/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 61.

  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_less_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)* at the location cited in Appendix A, Additional Resources.
Report Timing Summary Advanced Tab

Figure 1-10 shows the Advanced tab of the Report Timing Summary dialog box.

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`
Report Timing Summary

File Output

- **Write results to file**

  Writes 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 (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 the Report Timing Summary dialog box or the Tcl Console. 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 1-11 shows the Timer Settings tab of the Report Timing Summary dialog box.

Figure 1-11: Report Timing Summary Dialog Box: Timer Settings Tab

- **Interconnect**

  Controls whether net delays are calculated based on the estimated route distance between nodes, by the actual routed connection, or excludes net delay from timing. 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. This net is labeled unplaced in the timing path report.
  
  For placed but unrouted nodes, 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 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 Report Timing Summary contains the following sections:

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

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

The General Information section provides information about the following:

- 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

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

Figure 1-12, page 34, 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.

![Figure 1-12: Timing Summary Report Timer Settings](image-url)
Design Timing Summary

The Design Timing Summary section 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 1-13 shows an example of the Design Timing Summary section.

![Design Timing Summary](image)

**Figure 1-13:** Timing Summary of the Design

Setup

Setup includes 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.

- **Failing Endpoints**
  
  The total number of endpoints with a violation (WNS<0ns).

- **Total Number of Endpoints**
  
  The total number of endpoints analyzed.
Hold

Hold includes 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.

- **Failing Endpoints**
  
  The total number of endpoints with a violation (WHS < 0ns).

- **Total Number of Endpoints**
  
  The total number of endpoints analyzed.

Pulse Width

Pulse Width includes 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 WPWS value corresponds to the worst pulse width slack for all the checks listed above with min and max delays. The TWPS value is the sum of all WPWS violations, when considering only the worst violation of each timing path endpoint. Its value can only be 0ns or negative.

The three 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.

• **Failing Endpoints**

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

• **Total Number of Endpoints**

  The total number of endpoints analyzed.

**Clock Summary**

The Clock Summary section 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.

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

**Figure 1-14** shows an example of the Clock Summary section.
Check Timing

The Check Timing section 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.

To generate this 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 1-15, page 38, is as follows:

- **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_endpoints**
  Number of path endpoints 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 timing engine in order 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 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)* at the location cited in Appendix A, Additional Resources.
Intra-Clock Paths

The Intra Clock Paths section summarizes the worst slack and total violations of the timing paths with the same source and destination clocks.

Figure 1-16 shows an example of the Intra-Clock Paths section.

![Figure 1-16: Timing Summary Report Intra-Clock Paths](image)

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 1-17.

![Figure 1-17: Timing Summary Report Intra-Clock Paths Details](image)
Inter-Clock Paths

Similar to the Intra-Clock Paths section, the Inter-Clock Paths section summarizes the worst slack and total violations of timing paths with different source and destination clocks. See Figure 1-18.

![Timing Summary Report Inter-Clock Paths Details](image)

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

The Path Groups section displays default path groups and user-defined path groups. Figure 1-19 shows an example of the Path Groups summary table. To access this table, select Path Groups in the left pane.

![Timing Summary Report Path Groups](image)

**Note:** **async_default** is a path group automatically created by the Vivado 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 combination of source and destination clocks can be present in a path group.

**User-Ignored Paths**

The Timing Summary Report also 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Slack</th>
<th>From</th>
<th>To</th>
<th>Total Delay</th>
<th>Logic Delay</th>
<th>Net Delay</th>
<th>Logic %</th>
<th>Net %</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 725</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[14]/CLR</td>
<td>17.597</td>
<td>0.413</td>
<td>17.184</td>
<td>2.7</td>
<td>97.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 726</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[25]/CLR</td>
<td>17.458</td>
<td>0.471</td>
<td>16.980</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 727</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[26]/CLR</td>
<td>17.438</td>
<td>0.471</td>
<td>16.967</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 728</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[22]/CLR</td>
<td>17.438</td>
<td>0.471</td>
<td>16.967</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 729</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[23]/CLR</td>
<td>17.438</td>
<td>0.471</td>
<td>16.967</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 730</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[27]/CLR</td>
<td>17.438</td>
<td>0.471</td>
<td>16.967</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 731</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[24]/CLR</td>
<td>17.357</td>
<td>0.471</td>
<td>16.886</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 732</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[26]/CLR</td>
<td>17.357</td>
<td>0.471</td>
<td>16.886</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 733</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[25]/CLR</td>
<td>17.357</td>
<td>0.471</td>
<td>16.886</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 734</td>
<td>reset_reg_reg/C cpuEngine/or 1200_t/ttreg[30]/CLR</td>
<td>17.357</td>
<td>0.471</td>
<td>16.886</td>
<td>2.7</td>
<td>97.3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 1-20: Timing Summary Report - User-Ignored Paths*

**Unconstrained Paths**

The Unconstrained Paths section 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 1-21* shows an example of the Unconstrained Paths section summary.
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 4, 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 Violation

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

Figure 1-22 shows the Timing Summary window with only failing paths displayed.
Report Clock Networks

The Report Clock Network command can be run from:

- The Flow Navigator in the Vivado IDE, or
- The Tcl command:
  
  ```tcl
  report_clock_networks -name {network_1}
  ```

Report Clock Networks provides a tree view of the clock trees in the design. See Figure 1-23. Each clock tree shows the clock network from source to endpoint with the endpoints sorted by type.

![Clock Networks](image)

**Figure 1-23:** Clock Networks

The clock trees:

- Show clocks defined by the user or generated automatically by the tool.
- Report clocks from IO 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:

• **Tools > Timing > Report Clock Interaction** from the main menu, or
• Report Clock Interaction from the Netlist Analysis menu in the Flow Navigator, or
• The Tcl command:
  
  `report_clock_interaction -name clocks_1`

**Report Clock Interaction Dialog Box**

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

• **Options Tab**
• **Timer Settings Tab**

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

The equivalent Tcl option: `-name`
Options Tab

Figure 1-24 shows the Options tab of the Report Clock Interaction dialog box.

![Report Clock Interaction: Options Tab](image)

The Options tab contains the following fields:

- **Results name**
  
  Specifies the name for the returned results of the Report Clock Interaction command. The report opens with the specified name.

  Equivalent Tcl option: `-name`

- **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`
• **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**
  Writes 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 (1) the specified file is overwritten, or (2) new information is appended to an existing report.

  Equivalent Tcl option: `-append`

• **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.
Timer Settings Tab

Controls whether net delays are calculated based on the estimated route distance between nodes, by the actual routed connection, or excludes net delay from timing. 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. This net is labeled unplaced in the timing path report.

  For placed but unrouted nodes, 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 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 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 1-26, page 50.
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) at the location cited in Appendix A, Additional Resources.

- **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.

- **Green**
  
  Clock domains are fully constrained for cross-clock analysis.

- **Red**
  
  User-defined false_path or clock group constraints cover all paths crossing from the source clock to the destination clock.
• **Yellow (or Orange)**

  User-defined *false path* constraints cover *some* of the timing 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 timing of the paths between the domains. A green cell does not indicate that the timing is good, only that timing paths that cross clock domains are fully constrained.*

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 1-27**.

The table below the matrix displays the path with the worst case negative slack for each clock interaction. See **Figure 1-26, page 50**. This provides details not displayed in the matrix above. The table does not include paths from cells in which there are no cross-clock interactions (that is, the cells displayed in black on the matrix).

To sort the data in the table, select a column header to use as a key for sorting in increasing or decreasing amounts. Select once for up, once for down. Select again to turn off sorting.
and restore the table to its original state. Click **CTRL** to select additional columns to add secondary sorting criteria.

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.

The table columns are:

- **ID**
  A numeric ID for the path 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.

- **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 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.

- **Common Primary Clock**
  Whether the source and destination clocks of the timing path are defined by a common primary clock.
• Inter-Clock Constraints

Whether the clock domains are related, or are defined as false_paths, or constrained as asynchronous groups. Following are example definitions of these constraints:

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

A timing path, or multiple paths, can be selected from the table. Report Timing can be run from the popup menu. 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).

*Note:* ISE implementation calls this check Component Switching Limits.

To generate this report from the Tcl console, run:

```
report_pulse_width -name timing_1
```

---

**Per Session Reports**

The vivado.exe process creates the vivado.log (LOG) and vivado.jou (JOU) files in the directory you were in when you started the process. The vivado.exe process initializes the LOG and JOU files before the Vivado IDE starts. Select File > Open Log File and File > Open Journal File to view the LOG and JOU files at any time.
Report Timing

Read Report Timing to view specific timing paths at any point of the flow after synthesis when (1) You need to 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

Running Report Timing From the Menu

To run Report Timing from the Menu:


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.


To run Report Timing from the Timing Report and Timing Report Summary 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 options.
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.

• **Start Points (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 through points.

The Rise/Fall filter applies to the data edge.

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

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

• **End Points (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 1-29, page 56** 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 25.**
• **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**


• **Number of paths per endpoint**


• **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 a few 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**


• **Number of significant digits**


• **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**


**Timer Settings Tab**


**Reviewing Timing Path Details**

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

**Figure 1-30** shows the Report Timing window in which both min and max analysis (SETUP and HOLD) were selected, and N=4.

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Slack</th>
<th>From</th>
<th>To</th>
<th>Total Delay</th>
<th>Lo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 1</td>
<td>0.227</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.300</td>
<td></td>
</tr>
<tr>
<td>Path 2</td>
<td>0.233</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.383</td>
<td></td>
</tr>
<tr>
<td>Path 3</td>
<td>0.283</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.297</td>
<td></td>
</tr>
<tr>
<td>Path 4</td>
<td>0.316</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.264</td>
<td></td>
</tr>
<tr>
<td>Path 5</td>
<td>0.332</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.263</td>
<td></td>
</tr>
<tr>
<td>Path 6</td>
<td>0.339</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.201</td>
<td></td>
</tr>
<tr>
<td>Path 7</td>
<td>0.342</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.200</td>
<td></td>
</tr>
<tr>
<td>Path 8</td>
<td>0.362</td>
<td>cpuEngine/bram_reg/C</td>
<td>cpuEngine/bram_reg/C</td>
<td>7.224</td>
<td></td>
</tr>
</tbody>
</table>
```

**Figure 1-30:** Report Timing Window

Select any of these paths to view more details in the Path Properties window (Report tab).
Figure 1-31 shows a 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 4, 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 1-32, page 61.

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 Results 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**

![Report Datasheet Dialog Box Groups Tab](image)

*Figure 1-33: 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 tool 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 1-34** 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:

```tcl
set_property IDELAY_VALUE 13 [get_cells idelay_clk]
```
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 w.r.t. 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:

```bash
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.
**Note:** 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.
Logic Analysis

Introduction to Logic Analysis
This chapter contains information on:

- The elements in a netlist
- The netlist connectivity
- Searching for gates and primitive types
- Design flow in a chip
- Utilization
- Common design issues using DRC

RTL Analysis
For information about RTL Analysis, see the *Vivado Design Suite User Guide: System-Level Design Entry (UG895)* at the location cited in Appendix A, Additional Resources.
Netlist Window

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

![Netlist Window](image)

Figure 2-1: 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 edges. 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 ports 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 primitives folder if there are primitives at that level
- Any hierarchies instantiated at that level

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

For more information, see *Using the Netlist Window* in the *Vivado Design Suite User Guide: Using Vivado IDE (UG893)* at the location cited in Appendix A, Additional Resources.
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

![Instance Properties Window](image)

*Figure 2-2: Instance Properties Window*

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

---

**Hierarchy Window**

Explore the hierarchy to understand resource usage. To open the Hierarchy Window, select **Tools > Show Hierarchy**.

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 2-3 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 Utilization**.
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.
Viewing the Schematic

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.
For information about zooming and moving around the schematic, see the Vivado Design Suite User Guide: Using Vivado IDE (UG893) at the location cited in Appendix A, Additional Resources.

**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.
Viewing the Schematic

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 Vivado IDE (UG893) at the location cited in Appendix A, Additional Resources.

- 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.

![Schematic With Timing Path](image)

*Figure 2-7: Schematic With Timing Path*

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

![Timing Path with Select Primitive Parents](image)

*Figure 2-8: 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.
Searching for Objects Using Find

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

![Find Dialog Box]

Figure 2-9: Schematic With Timing Path Marked

Figure 2-10: 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 IOs
- 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.

For more information, see the Vivado Design Suite User Guide: Using the Tcl Scripting Capabilities (UG894) at the location cited in Appendix A, Additional Resources.
Design Data Flow – Top Level Floorplan

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 IO 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 understanding a full device.

![Place Pblocks Utilization](image)

The top level floorplan shows which blocks communicate with IOs (green lines). Nets shared between two Pblocks are bundled together. The bundles change size and color based on the number of shared nets. Two top level floorplans are shown below.
The Data Path Top Level Floorplan shows a data path. Each block communicates only to a drive block and a load block. The green lines show well-placed IOs that communicate with a single block.

**TIP:** This design is easy to 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.

**TIP:** This design is hard to 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.

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:

- IO
- 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
DRCs

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.

**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 IOs. 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.

Multiple Designs

A Design is an in-memory combination of a:

- Netlist
- Constraint set
- Part

The constraint set is a set of XDC constraint files with the target file marked for new constraints.

The tool supports multiple designs open at the same time. When analyzing code, it can be useful to have the Elaborated Design open at the same time as the Implemented Design. It is also possible to have multiple designs open at the same step of the flow.
The **Make Active** link appears in designs using the non-default settings. Click **Make Active** to update the defaults.

Use this feature to compare:

- Multiple different synthesis runs
- Constraint file iterations
- Different parts for the same netlist
- Two different Place and Router results

This feature works best on a 64-bit operating system. A simple way to open multiple designs is to open two runs at the same step. A more advanced method involves opening the single netlist in multiple windows:

1. Right click **Synthesized Design**.
2. Select **New Synthesized Design** from the popup menu.

Elaborated Design supports a similar method.
Multiple Windows

Use multiple monitors to float windows or designs. To open an entire Design in a different window, right click the blue Design bar -- for example, when you have multiple Synthesized Designs open and you want to view them simultaneously. Select Open in New Window from the popup menu.
For information on floating an individual window from a design, see *Using Windows in the Vivado Design Suite User Guide: Using the Vivado IDE (UG893)* at the location cited in Appendix A, Additional Resources.
Analyzing Implementation Results

Introduction to Analyzing Implementation Results

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

- Reviewing placement for hierarchical blocks
- IOs
- Looking at connectivity
- Cross probing between views
- Reviewing detailed routing

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) at the location cited in Appendix A, Additional Resources.

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, select Force up to date from the popup menu.

The Design Runs Window:

- Displays the message route_design Complete!
- Gives a brief summary.
- Reports any design error.
- Does not report whether the design met timing.
- Does not report Critical Warnings or other design issues.

If you are using the Vivado™ IDE project flow, review the Messages tab for your active implementation. Messages are grouped by each step in the run flow. The critical
information from the implementation steps appears in this view. See the message in the log file to view the context.

![Messages Window]

**Figure 3-2: Messages Grouped by Step**

If there are warnings or errors, review one of the reports listed above for more information. Some messages crossprobe back to the design elements related to the message.

To analyze the design in the context of the message, you might need to open either:

- Implemented Design (using the netlist at the end of implementation), or
- Netlist Design (using the netlist before implementation)

![Post Implementation Messages]

**Figure 3-3: Post Implementation Messages**
The router gives a message if a design met timing or not.

**IMPORTANT:** The router is not sign off timing. Only `report_timing_summary` is sign off timing.

Review the Timing Summary Report to determine if the design meets timing. If it did not meet timing, review the timing closure techniques discussed in Chapter 5, Closure Techniques.

If the Vivado IDE graphical user interface is open, in the non-project flow, messages generated during that executable session still display in the Messages Window. Otherwise, review the following:

- Vivado IDE log
- Notes, Warnings and Errors

If you are running without the Vivado IDE graphical user interface, run the Tcl `report_timing_summary` command to determine if the design met timing.

---

**More Analysis**

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.
Figure 3-4: Metrics
More Analysis

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 3-4, page 92. 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.
Highlight Placement

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

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

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

![Highlight Primitives Example](image)

*Figure 3-5: 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.

### Show 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. Use this technique to build up and see cones of logic inside the design.
Figure 3-6 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 tool tracks 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 3-7: Fixed Unfixed**

The IO and Block RAM placement is **Fixed**. The slice logic is **Unfixed**.
Fixing Logic

To fix logic:

- Enter placement into the XDC.
- Select logic in the Vivado IDE graphical user interface. From the popup, select \textbf{Fix Instances}, or
- Set the \texttt{is\_loc\_fixed} or \texttt{is\_bel\_fixed} properties in the Tcl console.

In the XDC, the constraints look like the following.

- For a Flip Flop with a LOC and BEL
  \begin{verbatim}
  set_property BEL DFF [get_cells 
    {cpuEngine/cpu_iwb_dat_i/buffer_fifo/infer_fifo.wr_addr_tmp_reg[9]}]
  set_property LOC SLICE_X7Y103 [get_cells 
    {cpuEngine/cpu_iwb_dat_i/buffer_fifo/infer_fifo.wr_addr_tmp_reg[9]}]
  \end{verbatim}

  Flip Flops can be placed with only a LOC.

- For a Block RAM
  \begin{verbatim}
  set_property LOC RAMB36_X0Y21 [get_cells 
    usbEngine1/dma_out/buffer_fifo/infer_fifo.block_ram_performance.fifo_ram_reg]
  \end{verbatim}

- For an IO Pin
  \begin{verbatim}
  set_property PACKAGE_PIN E23 [get_ports {DataIn_pad_0_i[5]}]
  \end{verbatim}

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 \textit{Go to instantiation} from the popup menu.
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 Routing

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

![Image of Viewing Routing](image)

**Figure 3-9**: 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.
Displaying Routing and Placement at Closer Zoom Levels

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

Figure 3-10: Abstract View

Figure 3-11: 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.

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

Figure 3-12: Layers
Navigating in 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.
Timing Analysis

Introduction to Timing Analysis

The Vivado™ Design Suite 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.

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.

**KEY CONCEPT:** 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_endpoint check).
• All input and output ports are constrained (no_input_delay, no_output_delay checks).

You can waive some of the missing constraints, at the risk of lowering the signoff quality of your design. You must also review the results of the other checks to make sure that they are expected, and that they do not affect the signoff quality.

Your Design Meets Timing

• Total Negative Slack (TNS) is 0ns.
  
  Covers max delay analysis.

• Total Hold Slack (THS) is 0ns.
  
  Covers min delay analysis.

• 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 4-1 highlights in green the information you must examine in order to verify that the timing signoff is clean.

Figure 4-1: Report Timing Summary Signoff in Vivado IDE

Figure 4-2 shows the Check Timing information to verify in the text report.

Figure 4-2: Check Timing Signoff in Text Report
Verifying Timing Signoff

Figure 4-3 shows the Design Timing Summary information to verify in the text report.

![Design Timing Summary](image)

**Figure 4-3:** 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 and recovery and 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 in Chapter 1, Design Analysis.

**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.
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.

Timing Path Summary

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 Timing Analysis in the Vivado Design Suite Using Constraints User Guide (UG903) at the location cited in Appendix A, Additional Resources.

Timing Path Summary Header Examples

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

```
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack (HST)</td>
<td>1.690ns</td>
</tr>
<tr>
<td>Source</td>
<td>c0/O/AOutReg_reg/CLK</td>
</tr>
<tr>
<td></td>
<td>(rising edge-triggered cell DSP48E1 clocked by bftClk)</td>
</tr>
<tr>
<td></td>
<td>(<a href="mailto:rise@0.000ns">rise@0.000ns</a> <a href="mailto:fall@8.500ns">fall@8.500ns</a> period=5.000ns)</td>
</tr>
<tr>
<td>Destination</td>
<td>transformLoop[0].c0/O/AOutReg_reg/C[16]</td>
</tr>
<tr>
<td></td>
<td>(rising edge-triggered cell DSP48E1 clocked by bftClk)</td>
</tr>
<tr>
<td></td>
<td>(<a href="mailto:rise@0.000ns">rise@0.000ns</a> <a href="mailto:fall@8.500ns">fall@8.500ns</a> period=5.000ns)</td>
</tr>
<tr>
<td>Path Group</td>
<td>bftClk</td>
</tr>
<tr>
<td>Path Type</td>
<td>Max at Slow Process Corner</td>
</tr>
<tr>
<td>Requirement</td>
<td>5.000ns</td>
</tr>
<tr>
<td>Data Path Delay</td>
<td>2.646ns (logic 0.55ns (20.97%) route 2.091ns (79.03%))</td>
</tr>
<tr>
<td>Logic Levels</td>
<td>1 (LUT2=1)</td>
</tr>
<tr>
<td>Clock Path Skew</td>
<td>-0.105ns (OCD - SCD + CFR)</td>
</tr>
<tr>
<td>Source Clock Delay (SCD)</td>
<td>4.123ns</td>
</tr>
<tr>
<td>Clock Rampism Removal (CFR)</td>
<td>0.223ns</td>
</tr>
<tr>
<td>Clock Uncertainty</td>
<td>0.061ns (([TSP/2 + TIP/2]+1/2 + DJ) / 2 + PE + UU</td>
</tr>
<tr>
<td>Total System Jitter (TJ)</td>
<td>0.107ns</td>
</tr>
<tr>
<td>Total Input Jitter (TIJ)</td>
<td>0.100ns</td>
</tr>
<tr>
<td>Discrete Jitter (DJ)</td>
<td>0.000ns</td>
</tr>
<tr>
<td>Phase Error (PE)</td>
<td>0.000ns</td>
</tr>
<tr>
<td>User Uncertainty (UU)</td>
<td>0.500ns</td>
</tr>
</tbody>
</table>
```

Figure 4-4: Timing Path Summary In Text Report
Figure 4-5 shows an example of the Timing Path Summary header in the Vivado IDE.

![Timing Path Summary in Vivado IDE](image)

### 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)**

  \[
  \text{slack} = \text{data required time} - \text{data arrival time}
  \]

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

  \[
  \text{slack} = \text{data arrival time} - \text{data required time}
  \]

  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.

  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)* at the location cited in Appendix A, Additional Resources.

• **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 timing engine.

• **Total System Jitter (TSJ)**

The combined system jitter applied to both source and destination clocks. To modify the system jitter globally, run the `set_system_jitter` XDC command. The virtual clocks are ideal and therefore do not have any system jitter.


• **Total Input Jitter (TIJ)**

The combined input jitter of both source and destination clocks. To define the input jitter for each primary clock individually, run the `set_input_jitter` XDC command. The Vivado IDE timing engine computes the generated clocks input jitter based on their
Reading a Timing Path Report

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)* at the location cited in Appendix A, Additional Resources.

- **Discrete Jitter (DJ)**
  
The amount of jitter introduced by hardware primitives such as MMCM or the PLL. The Vivado 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 and the PLL. The Vivado 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` XDC command.

  For more information on how to use this command, see the *Vivado Design Suite User Guide: Using Constraints (UG903)* at the location cited in Appendix A, Additional Resources.

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` XDC command 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` XDC command 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)* at the location cited in Appendix A, Additional Resources.

**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)
  
  \[
  \text{slack} = \text{data required time} - \text{data arrival time}
  \]

- For min delay analysis (hold/removal)
  
  \[
  \text{slack} = \text{data arrival time} - \text{data required time}
  \]
Timing Path Details In Text Report

Figure 4-6 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.

<table>
<thead>
<tr>
<th>Location</th>
<th>Delay type</th>
<th>Incרגm (ns)</th>
<th>Path (ns)</th>
<th>Netlist Resource(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W17</td>
<td>(clock bftClk rise edge)</td>
<td>0.000</td>
<td>0.000</td>
<td>bftClk</td>
</tr>
<tr>
<td>W17</td>
<td>net (fo=0)</td>
<td>0.000</td>
<td>0.000</td>
<td>bftClk</td>
</tr>
<tr>
<td>BUFCTRL_X0Y0</td>
<td>IBUF (Prop_ibuf_3,0)</td>
<td>0.586</td>
<td>0.586</td>
<td>bftClk_IBUF_inst/0</td>
</tr>
<tr>
<td>BUFCTRL_X0Y0</td>
<td>net (fo=1, routed)</td>
<td>3.001</td>
<td>2.769</td>
<td>bftClk_IBUF</td>
</tr>
<tr>
<td>BUFCTRL_X0Y0</td>
<td>BUFQ (Prop_bufq_1,0)</td>
<td>0.093</td>
<td>2.862</td>
<td>bftClk_IBUF_BUFQ_inst/0</td>
</tr>
<tr>
<td>X1Y28</td>
<td>net (fo=746, routed)</td>
<td>1.261</td>
<td>4.123</td>
<td>bftClk_IBUF_BUFQ</td>
</tr>
</tbody>
</table>

Figure 4-6: Timing Path Details In Text Report

Timing Path Details in Vivado IDE

The Timing Path Details in the Vivado IDE, as shown in Figure 4-7, page 116, shows the same information as is shown in the text report, seen in Figure 4-6.
Reading a Timing Path Report

The information on the path is presented 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 \((f_o)\) 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 selected for the path. It can be transformed 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.
Chapter 5

Closure Techniques

Introduction to Closure Techniques

This chapter discusses techniques for timing closure including:

• Refining timing constraints
• Floorplanning
• Making the implementation tools work harder
• Automatic and manual ways to improve design timing

Check Input Constraints and Sources

Ensure that design and timing inputs 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 can tune the gates to meet timing.

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

  Be sure that the clock trees are reasonable. Designs can hit long clock skew when one BUFG drives a second BUFG or a LUT. The extra clock skew can lead to tight hold timing constraints, degrading system timing. 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 FPGAs devices support twelve global clocks per region. There are additional limitations on the placement of clock trees.

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

If you are overconstraining the design, the tools will try to hit 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 false paths.

For XDC, the tool assumes that all cross clock domain crossings are real paths. This is a change from ISE and UCF. Other paths may not be reachable due to the structure of the control logic. Enter False Path Timing constraints.

Other control structures (for example, a clock enable driven by a state machine) generate multicycle paths. Data has multiple clock periods to get from the source to the destination. Enter Multicycle Timing constraints.

• 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. Two ways to try to reduce route delay are:
Increasing Tool Effort

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

Use the **HighEffort Strategy**. Implementation runs at a medium effort level by default. The `place_design` and `route_design` commands accept a `--effort_level high` switch. The tools will try different approaches to meet timing at the expense of runtime.

Consider using the **HighEffortPhysSynth** strategy. It turns up the effort level and turns on the optional `phys_opt_design` step. It works to improve timing at the expense of run time.

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

---

**Manual Floorplanning**

Floorplanning can help a design meet timing. Floorplan when a design:

- Does not meet timing consistently.
- 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 delay (TNS WNS) by reducing route delay. During implementation, the timing engine reports on Setup, Hold, and Component Switching Limits. 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 XST, use:
  - `–keep_hierarchy=no` (default) and `–netlist_hierarchy = rebuilt` (non-default), or
  - The PlanAhead™ tool defaults strategy
- For Vivado Synthesis use:
  - `–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.

**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 logic elements 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. For more information, see Multiple Windows.

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. logic 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 Element Placement**

Review the placement of the elements in the design. Compare two IO reports to review the IO placement and IO standards. Make sure all the IOs are placed. A simple search finds all IOs without fixed placement as shown in Figure 5-2.

![Figure 5-2: IO Is Not Fixed](image)

Consider placing all the clock primitives based on the run that met timing. The Clock Utilization Report lists the placement of the clock tree drivers, as shown in Figure 5-3.

![Figure 5-3: Clock Locations](image)
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 5-4: DSP or RAM](image)

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.
3. Select Fix Instances.
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:

- IOs
- Global Clock Resources
- BlockRam
- 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.
Hand Floorplanning

Consider gate 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. 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.

Hand Gate Placement

Hand gate 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 IO interface, or hand place Block RAMs and DSPs. Hand placement can be slow.

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

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 gates.
- Has comprehensive knowledge of the gates and timing paths.
- Generally does a good job of fine grain placement.

When floorplanning, you should have an idea of final pinout. It is useful to have the IOs fixed. The IOs can provide anchor points for starting the floorplan. Logic that communicates to IOs migrates towards the fixed pins.
**TIP:** Place blocks that communicate with IOs near their IOs. If the pinout is pulling a block apart, consider pinout or RTL modification.
Figure 5-6: IO Components Pulling Design Apart
The floorplan shown in Figure 5-6, page 129 may 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 above is represented by the XDC constraints:

```plaintext
create_Pblock Pblock_fftEngine
add_cells_to_Pblock [get_Pblocks Pblock_fftEngine] [get_cells -quiet [list usbEngine1]]
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {SLICE_X8Y105:SLICE_X23Y149}
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {DSP48_X0Y42:DSP48_X1Y59}
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {RAMB18_X0Y42:RAMB18_X1Y59}
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {RAMB36_X0Y21:RAMB36_X1Y29}
```

There is one line assigning the level of hierarchy to the Pblock. There are four resource types 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.

X-Ref Target - Figure 5-7

Figure 5-7: Pblock Grids

The resulting XDC constraints the simplified Pblock:

```plaintext
create_Pblock Pblock_fftEngine
add_cells_to_Pblock [get_Pblocks Pblock_fftEngine] [get_cells -quiet [list usbEngine1]]
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {RAMB18_X0Y42:RAMB18_X1Y59}
resize_Pblock [get_Pblocks Pblock_fftEngine] -add {RAMB36_X0Y21:RAMB36_X1Y29}
```

The Block RAMs are constrained in the device, but the slice logic is free to migrate to the IOs.
TIP: When floorplanning logic, be careful not to floorplan hierarchy in such a manner that it crosses the central config block.

Figure 5-8: Avoiding the Config Block
Stacked Silicon Interconnect (SSI)

There are extra considerations for Stacked Silicon Interconnect (SSI) parts. The SSI parts are made of multiple die or Super Logic Regions (SLRs), joined by an interposer. The interposer connections are called Super Long Lines (SLLs). There is some delay 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 critical timing paths inside an SLR.

The IOs are placed in the same SLR as the relevant IO interface. You should also 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.
Modify Routing

Depending on results, you may want to modify routes. The Device View allows you to Unroute, Route, and Fix Routing on any individual net. These commands are available from the context menu on a net.

Because only the commands applicable to a net are shown, not all commands are shown for all nets.

Figure 5-10: Net Context
Modify Logic

Properties on logical objects that are not Read Only can be modified after Implementation in the Vivado IDE graphical user interface as well as Tcl. Select the Attributes tab in the Properties window.

These properties can include everything from Block RAM INITs to the clock modifying properties on MMCMS. There is also a special dialog box to set or modify INIT on LUT objects. This dialog box allows you to specify the LUT equation and have the tool determine the appropriate INIT.

Figure 5-11: Property Modify
Using Modifications

Once modified, a checkpoint is the recommended way to capture the design in memory. Because logical changes are not back-annotated to the logical design, you must make the logical modifications in Source or XDC for them to impact the next run.
Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:


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

These documents provide supplemental material useful with this guide:

- Vivado Design Suite Tcl Command Reference Guide (UG835)
- Vivado Design Suite User Guide: Using the Tcl Scripting Capabilities (UG894)
- Vivado Design Suite User Guide: Implementation (UG904)
- Power Analysis and Optimization (UG907)
These documents are available at:

*Vivado Design Suite 2012.4 Documentation*  