



# Intel<sup>®</sup> Quartus<sup>®</sup> Prime Pro Edition User Guide

---

## Design Optimization

Updated for Intel<sup>®</sup> Quartus<sup>®</sup> Prime Design Suite: **20.3**



[Subscribe](#)

[Send Feedback](#)

**UG-20133 | 2020.09.28**

Latest document on the web: [PDF](#) | [HTML](#)



# Contents

---

- 1. Design Optimization Overview..... 6**
  - 1.1. Device Considerations..... 6
    - 1.1.1. Device Migration Considerations..... 6
  - 1.2. Required Settings for Initial Compilation..... 7
    - 1.2.1. Initial I/O Assignment Guidelines..... 7
    - 1.2.2. Initial Timing Constraint Guidelines..... 7
  - 1.3. Optimization Trade-Offs and Limitations..... 8
    - 1.3.1. Area Reduction Trade-Offs..... 8
    - 1.3.2. Critical Path Delay Reduction Trade-Offs..... 8
    - 1.3.3. Power Consumption Reduction Trade-Offs..... 9
    - 1.3.4. Compilation Time Trade-Offs..... 9
  - 1.4. Design Visualization and Optimization Tools..... 9
    - 1.4.1. Design Visualization Tools..... 9
    - 1.4.2. Design Optimization Tools..... 10
  - 1.5. Design Optimization Overview Revision History..... 11
- 2. Optimizing the Design Netlist..... 13**
  - 2.1. When to Use the Netlist Viewers: Analyzing Design Problems ..... 13
  - 2.2. Intel Quartus Prime Design Flow with the Netlist Viewers..... 14
  - 2.3. RTL Viewer Overview..... 15
    - 2.3.1. Maximizing Readability in RTL Viewer..... 16
    - 2.3.2. Running the RTL Viewer..... 16
  - 2.4. Technology Map Viewer Overview..... 16
  - 2.5. Netlist Viewer User Interface..... 17
    - 2.5.1. Netlist Navigator Pane..... 20
    - 2.5.2. Properties Pane..... 20
    - 2.5.3. Netlist Viewers Find Pane..... 22
  - 2.6. Schematic View..... 22
    - 2.6.1. Display Schematics in Multiple Tabbed View..... 22
    - 2.6.2. Schematic Symbols..... 23
    - 2.6.3. Select Items in the Schematic View..... 27
    - 2.6.4. Shortcut Menu Commands in the Schematic View..... 27
    - 2.6.5. Filtering in the Schematic View..... 28
    - 2.6.6. View Contents of Nodes in the Schematic View..... 28
    - 2.6.7. Moving Nodes in the Schematic View..... 30
    - 2.6.8. View LUT Representations in the Technology Map Viewer..... 31
    - 2.6.9. Zoom Controls..... 31
    - 2.6.10. Navigating with the Bird's Eye View..... 32
    - 2.6.11. Partition the Schematic into Pages..... 32
    - 2.6.12. Follow Nets Across Schematic Pages..... 33
    - 2.6.13. Maintaining Selection in the Resource Property Viewer..... 33
  - 2.7. Cross-Probing to a Source Design File and Other Intel Quartus Prime Windows..... 34
  - 2.8. Cross-Probing to the Netlist Viewers from Other Intel Quartus Prime Windows..... 35
  - 2.9. Viewing a Timing Path..... 36
  - 2.10. Optimizing the Design Netlist Revision History..... 36
- 3. Netlist Optimizations and Physical Synthesis..... 38**
  - 3.1. Physical Synthesis Optimizations..... 38



3.1.1. Enabling Physical Synthesis Optimization.....	39
3.1.2. Physical Synthesis Options.....	39
3.2. Applying Netlist Optimizations.....	39
3.2.1. WYSIWYG Primitive Resynthesis.....	40
3.3. Scripting Support.....	41
3.3.1. Synthesis Netlist Optimizations.....	41
3.3.2. Physical Synthesis Optimizations.....	42
3.4. Netlist Optimizations and Physical Synthesis Revision History.....	42
<b>4. Area Optimization.....</b>	<b>44</b>
4.1. Resource Utilization Information.....	44
4.1.1. Flow Summary Report.....	44
4.1.2. Fitter Reports.....	45
4.1.3. Analysis and Synthesis Reports.....	45
4.1.4. Compilation Messages.....	45
4.2. Optimizing Resource Utilization.....	46
4.2.1. Resource Utilization Issues Overview.....	46
4.2.2. I/O Pin Utilization or Placement.....	46
4.2.3. Logic Utilization or Placement.....	47
4.2.4. Routing.....	51
4.3. Scripting Support.....	53
4.3.1. Initial Compilation Settings.....	54
4.3.2. Resource Utilization Optimization Techniques.....	54
4.4. Area Optimization Revision History.....	55
<b>5. Timing Closure and Optimization.....</b>	<b>56</b>
5.1. Optimize Multi Corner Timing.....	56
5.2. Optimize Critical Paths.....	56
5.2.1. Viewing Critical Paths.....	57
5.3. Optimize Critical Chains.....	57
5.3.1. Viewing Critical Chains.....	57
5.4. Design Evaluation for Timing Closure.....	58
5.4.1. Review Messages.....	58
5.4.2. Evaluate Fitter Netlist Optimizations.....	58
5.4.3. Evaluate Optimization Results.....	58
5.4.4. Evaluate Resource Usage.....	58
5.4.5. Evaluate Other Reports and Adjust Settings Accordingly.....	62
5.4.6. Evaluate Clustering Difficulty.....	64
5.4.7. Revise and Recompile.....	64
5.5. Timing Optimization.....	64
5.5.1. Correct Design Assistant Rule Violations.....	64
5.5.2. Implement Fast Forward Timing Closure Recommendations.....	66
5.5.3. View Timing Optimization Advisor.....	68
5.5.4. Review Timing Path Details.....	69
5.5.5. Try Optional Fitter Settings.....	88
5.5.6. Back-Annotate Optimized Assignments.....	89
5.5.7. Optimize Settings with Design Space Explorer II.....	91
5.5.8. I/O Timing Optimization Techniques.....	93
5.5.9. Register-to-Register Timing Optimization Techniques.....	98
5.5.10. Metastability Analysis and Optimization Techniques.....	112
5.6. Periphery to Core Register Placement and Routing Optimization .....	112



- 5.6.1. Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box..... 113
- 5.6.2. Setting Periphery to Core Optimizations in the Assignment Editor..... 114
- 5.6.3. Viewing Periphery to Core Optimizations in the Fitter Report..... 114
- 5.7. Scripting Support..... 115
  - 5.7.1. Initial Compilation Settings.....116
  - 5.7.2. I/O Timing Optimization Techniques .....116
  - 5.7.3. Register-to-Register Timing Optimization Techniques.....117
- 5.8. Timing Closure and Optimization Revision History..... 117
- 6. Analyzing and Optimizing the Design Floorplan..... 121**
  - 6.1. Design Floorplan Analysis in the Chip Planner.....121
    - 6.1.1. Starting the Chip Planner..... 122
    - 6.1.2. Chip Planner GUI Components..... 122
    - 6.1.3. Viewing Design Elements in the Chip Planner..... 123
    - 6.1.4. Exploring Paths in the Chip Planner.....131
    - 6.1.5. Viewing Assignments in the Chip Planner.....134
    - 6.1.6. Viewing High-Speed and Low-Power Tiles in the Chip Planner..... 135
  - 6.2. Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner.....135
    - 6.2.1. Viewing Design Connectivity and Hierarchy..... 136
  - 6.3. Using Logic Lock Regions in the Chip Planner..... 138
    - 6.3.1. Viewing Connections Between Logic Lock Regions in the Chip Planner.....138
    - 6.3.3. Logic Lock Regions..... 138
    - 6.3.4. Attributes of a Logic Lock Region..... 139
    - 6.3.5. Migrating Assignments between Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition..... 139
    - 6.3.6. Creating Logic Lock Regions.....139
    - 6.3.7. Customizing the Shape of Logic Lock Regions.....142
    - 6.3.8. Placing Device Resources into Logic Lock Regions..... 144
    - 6.3.9. Hierarchical Regions..... 148
    - 6.3.10. Additional Intel Quartus Prime Logic Lock Design Features..... 148
    - 6.3.11. Logic Lock Regions Window.....148
    - 6.3.12. Snapping to a Region..... 149
  - 6.4. Using User-Defined Clock Regions in the Chip Planner..... 150
    - 6.4.1. Creating Clock Assignments with the Chip Planner..... 151
    - 6.4.2. Resizing a Clock Assignment..... 152
    - 6.4.3. Moving a Clock Assignment..... 152
    - 6.4.4. Deleting a Clock Region Assignment..... 152
    - 6.4.5. Assigning a Clock Signal to a Clock Region.....153
    - 6.4.6. Clock Assignment Properties..... 153
  - 6.5. Scripting Support..... 154
    - 6.5.1. Creating Logic Lock Assignments with Tcl commands..... 154
    - 6.5.2. Assigning Virtual Pins with a Tcl command..... 155
    - 6.5.3. Logic Lock Region Assignment Examples..... 155
  - 6.6. Analyzing and Optimizing the Design Floorplan Revision History..... 156
- 7. Using the ECO Compilation Flow..... 159**
  - 7.1. ECO Compilation Flow.....159
  - 7.2. ECO Tcl Script Example..... 160
  - 7.3. Viewing ECO Compilation Reports.....161
  - 7.4. ECO Commands.....162



7.4.1. ECO Command Quick Reference.....	163
7.4.2. make_connection.....	163
7.4.3. remove_connection.....	164
7.4.4. modify_lutmask.....	165
7.4.5. adjust_pll_refclk.....	165
7.4.6. modify_io_slew_rate.....	166
7.4.7. modify_io_current_strength.....	166
7.4.8. modify_io_delay_chain.....	166
7.4.9. create_new_node.....	167
7.4.10. remove_node.....	168
7.4.11. place_node.....	168
7.4.12. unplace_node.....	169
7.4.13. create_wirelut.....	169
7.5. ECO Command Limitations.....	170
7.6. Interactive ECO Fitting.....	171
7.6.1. eco_load_design and eco_commit_design Commands.....	171
7.7. Using the ECO Compilation Flow Revision History.....	172
<b>8. Intel Quartus Prime Pro Edition Design Optimization User Guide Archives.....</b>	<b>173</b>
<b>A. Intel Quartus Prime Pro Edition User Guides.....</b>	<b>174</b>



## 1. Design Optimization Overview

---

The early stages of FPGA design development typically focus on meeting timing requirement, resource usage, and power consumption goals. After meeting those basic goals, you can focus on optimizing performance.

Optimization of FPGA design performance requires a multi-dimensional approach to meet the design goals, while also reducing area, critical path delay, power consumption, and runtime. The Intel® Quartus® Prime software supports various tools and techniques for performance optimization.

This chapter describes the techniques and tools in the Intel Quartus Prime software that you can use to meet your design goals and achieve the highest design performance in Intel FPGAs.

### Related Information

[Compilation Flows, Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)

### 1.1. Device Considerations

All Intel FPGAs have a unique timing model that contains delay information for all physical elements in the device, such as combinational adaptive logic modules, memory blocks, interconnects, and registers. The delays encompass all valid combinations of operating conditions for the target FPGA. Additionally, the device size and package determine pin-out and the resource availability. When selecting the target Intel FPGA device for your design, consider the performance specifications of the device.

#### 1.1.1. Device Migration Considerations

If you anticipate a change to the target device later in the design cycle, plan for the migration from the beginning of cycle. This strategy helps to minimize changes to the design at a later stage.

When choosing a design's target device in the Intel Quartus Prime software, you can see a list of compatible devices by clicking the **Migration Devices** button in the **Device** dialog box.

### Related Information

[Migration Devices Dialog Box](#)  
In *Intel Quartus Prime Help*



## 1.2. Required Settings for Initial Compilation

Compilation results can vary significantly, depending on the assignments and settings that you choose. In the Intel Quartus Prime software, the default values for timing settings provide the best trade-off between compilation time, resource utilization, and timing performance. You can adjust these settings if you want to further optimize for compile time, resource utilization, or timing performance. Before compiling a design in the Intel Quartus Prime software, consider the following guidelines for specifying initial settings.

### 1.2.1. Initial I/O Assignment Guidelines

In a FPGA design, I/O standards and drive strengths affect I/O timing. Follow these guidelines when specifying initial I/O assignments:

- When specifying I/O assignments, make sure that the Intel Quartus Prime software is using an accurate I/O timing delay for timing analysis and Fitter optimizations.
- If the PCB layout does not indicate pin locations, then leave the pin locations unconstrained. This technique allows the Compiler to search for the best layout. Otherwise, make pin assignments to constrain the compilation appropriately.

#### Related Information

##### [I/O Planning Overview](#)

In *Intel Quartus Prime Pro Edition User Guide: Design Constraints*

### 1.2.2. Initial Timing Constraint Guidelines

For best results, use realistic timing requirements for initial compilation. Applying more stringent timing requirements than the design requires can cause the Compiler to increase performance at the expense of resource usage, power utilization, or compilation time.

Specifying comprehensive timing requirement settings helps you to achieve the best results for the following reasons:

- Comprehensive timing assignments enable the Compiler to work hardest to optimize the performance of the timing-critical parts of the design. This optimization can also save area or power utilization in non-critical parts of the design.
- If enabled, the Intel Quartus Prime software can perform physical synthesis optimizations based on the comprehensive timing requirements.

Following compilation and timing analysis, the Compilation Report reports whether the design meets the timing requirements. You can then use the Intel Quartus Prime Timing Analyzer reporting commands to provide detailed information about all timing paths.

#### Related Information

- [Using the Intel Quartus Prime Timing Analyzer](#)  
In *Intel Quartus Prime Pro Edition User Guide: Timing Analyzer*
- [Intel Quartus Prime Timing Analyzer Cookbook](#)

## 1.3. Optimization Trade-Offs and Limitations

Design optimization is often a trade-off between performance, resource usage, power utilization, and compilation time. You can apply various settings to achieve the correct balance of these factors for your design goals.

**Table 1. Design Optimization Trade-Off Examples**

Trade-off	Comments
Resource usage and critical path timing.	Certain techniques (such as logic duplication) can improve timing performance at the cost of increased area.
Power requirements can result in area and timing trade-offs.	For example, reducing the number of available high-speed tiles, or attempting to shorten high-power nets at the expense of critical path nets.
System cost and time-to-market considerations can affect the choice of device.	For example, a device with a higher speed grade or more clock networks can facilitate timing closure at the expense of higher power consumption and system cost.

Finally, constraints that are too stringent can produce a situation with no possible solution for the selected device. If the Fitter cannot resolve a design due to resource limitations, timing constraints, or power constraints, consider rewriting parts of the HDL code.

### 1.3.1. Area Reduction Trade-Offs

By default, the Intel Quartus Prime Fitter might physically spread a design over the entire device to meet the set timing constraints. If you prefer to optimize your design to use the smallest area, you can change this behavior. If you require reduced area, you can enable certain physical synthesis options to modify your netlist to create a more area-efficient implementation, but at the cost of increased runtime and decreased performance.

#### Related Information

- [Area Optimization](#) on page 44
- [Netlist Optimizations and Physical Synthesis](#) on page 38

### 1.3.2. Critical Path Delay Reduction Trade-Offs

To meet complex timing requirements involving multiple clocks, routing resources, and area constraints, the Intel Quartus Prime software offers a close interaction between synthesis, floorplan editing, place-and-route, and timing analysis processes.

By default, the Intel Quartus Prime Fitter works to meet the timing requirements, and stops when the requirements are met. Therefore, specifying realistic constraints is crucial for timing closure.

Under-constrained designs can lead to sub-optimal results. For over-constrained designs, the Fitter might over-optimize non-critical paths at the expense of true critical paths. In addition, area and compilation time may also increase.

For designs with high resource usage, the Intel Quartus Prime Fitter might have trouble finding a legal placement. In such circumstances, the Fitter automatically modifies settings to try to trade off performance for area.





The Intel Quartus Prime Fitter offers advanced options that can help improve the design performance when you properly set constraints. Use the Timing Optimization Advisor to determine which options are best suited for the design.

In high-density FPGAs, routing accounts for a major part of critical path timing. Because of this, duplicating or retiming logic can allow the Fitter to reduce delay on critical paths. The Intel Quartus Prime software offers push-button netlist optimizations and physical synthesis options that can improve design performance at the expense of considerable increases of compilation time and area. Turn on only those options that help you keep reasonable compilation times and resource usage. Alternately, you can modify the HDL to manually duplicate or adjust the timing logic.

#### Related Information

[Optimize Critical Paths](#) on page 56

### 1.3.3. Power Consumption Reduction Trade-Offs

The Intel Quartus Prime software has features that help reduce design power consumption. The power optimization options control the power-driven compilation settings for Synthesis and the Fitter. You can adjust these settings

#### Related Information

[Power Optimization](#)

In *Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization*

### 1.3.4. Compilation Time Trade-Offs

Many Fitter settings influence compilation time. Most of the default settings in the Intel Quartus Prime software are set for reduced compilation time. You can modify these settings for your project requirements to trade-off longer compilation time for increased performance.

The Intel Quartus Prime software supports parallel compilation in computers with multiple processors. This technique can reduce compilation times by up to 15%.

#### Related Information

[Reducing Compilation Time, Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)

## 1.4. Design Visualization and Optimization Tools

The Intel Quartus Prime software provides various tools to help you visualize and optimize the design settings and constraints for the best design implementation.

### 1.4.1. Design Visualization Tools

The Intel Quartus Prime software provides tools that display different graphical representations of your design to help you visualize and optimize placement, connectivity, and routing congestion at various stages of the design cycle.

**Table 2. Design Visualization Tools**

Tool	Description
RTL Viewer	Provides a schematic representation of the design before synthesis and place-and-route.
Technology Map Viewer	Provides a schematic representation of the design implementation in the selected device architecture after synthesis and place-and-route. Optionally, you can include timing information.
Chip Planner	Allows you to make floorplan assignments, perform power analysis, and visualize critical paths and routing congestion.
Interface Planner	Simplifies the planning of accurate constraints for physical implementation. Use Interface Planner to prototype interface implementations, plan clocks, and rapidly define a legal device floorplan.
State Machine Viewer	Presents a high-level, graphical view of finite state machines in your design. The viewer displays the states and their related transitions, as well as a state transition table with condition equations for state transitions, and encoding information for each state.
Design Partition Planner	Displays design entities, I/O banks, connectivity, design hierarchy, and design partition membership. Design Partition Planner can assist you in visualizing a design's structure for creating effective design partitions.
Design Partition Planner and Chip Planner	Allows you to partition and layout the design at a higher level chip view.

**Related Information**

- [Design Floorplan Analysis in the Chip Planner](#) on page 121
- [Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner](#) on page 135
- [RTL Viewer Overview](#) on page 15
- [Technology Map Viewer Overview](#) on page 16
- [State Machine Viewer Overview](#)
- [Interface Planning, Intel Quartus Prime Pro Edition User Guide: Design Constraints](#)

**1.4.2. Design Optimization Tools**

The Intel Quartus Prime software provides tools help you identify design RTL and project settings that potentially limit performance.

**Table 3. Design Optimization Tools**

Tool	Description	To Access
Design Assistant	Automatically reports any violations against a standard set of Intel FPGA-recommended design guidelines, as <a href="#">Correct Design Assistant Rule Violations</a> on page 64 describes.	<b>Assignments &gt; Settings &gt; Design Assistant Rule Settings</b>
Fast Forward Timing Closure Recommendations	Fast Forward compilation generates design recommendations to help you to break performance bottlenecks and maximize use of Hyper-Registers to drive the highest performance in Intel Stratix® 10 and Intel Agilex™ designs, as <a href="#">Implement Fast Forward Timing Closure Recommendations</a> on page 66 describes.	On the Compilation Dashboard, click <b>Fast Forward Timing Closure Recommendations</b> .
<i>continued...</i>		



Tool	Description	To Access
Design Space Explorer II	Provides an easy and efficient way to run seed sweeps with different combinations of design settings and constraints to identify the optimal combination for your design, as <a href="#">Optimize Settings with Design Space Explorer II</a> on page 91 describes.	<b>Tools &gt; Launch Design Space Explorer II</b>
Assignment Back-Annotation Dialog Box	Back-annotation copies the last compilation's resource assignments to preserve your optimized implementation in subsequent compilations, as <a href="#">Back-Annotate Optimized Assignments</a> on page 89 describes.	<b>Assignments &gt; Back-Annotate Assignments</b>
Design Advisors	Provide recommendations about your current project settings and constraints. Consider the recommendations to optimize fitting, meet timing or power requirements, or improve the design performance, as <a href="#">View Timing Optimization Advisor</a> on page 68 describes.	Click <b>Tools &gt; Advisors &gt;</b> <ul style="list-style-type: none"> <li>• <b>Timing Optimization Advisor</b></li> <li>• <b>Power Optimization Advisor</b></li> <li>• <b>Compilation Time Advisor</b></li> </ul>

### Related Information

- [Fast Forward Compilation, Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)
- [Design Assistant, Intel Quartus Prime Pro Edition User Guide: Design Recommendations](#)

## 1.5. Design Optimization Overview Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2020.09.28	20.3.0	<ul style="list-style-type: none"> <li>• Reorganized "Design Optimization Tools" section.</li> <li>• Added references to Design Assistant, Fast Forward Timing Closure Recommendations, and assignment back-annotation to "Design Optimization Tools" topic.</li> <li>• Added Interface Planner and State Machine Viewer to list of "Design Visualization Tools".</li> <li>• Reworded titles in "Optimization Trade-Offs and Limitations" section for greater clarity.</li> </ul>
2018.05.07	18.0.0	<ul style="list-style-type: none"> <li>• General topic reorganization.</li> <li>• Added how DSE II works, and the main steps to follow when performing a design exploration.</li> </ul>
2017.11.06	17.1.0	<ul style="list-style-type: none"> <li>• Added mention to the Design Partition Planner in Design Analysis topic.</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>• Implemented Intel rebranding.</li> </ul>
2016.05.03	16.0.0	Removed statements about serial equivalence when using multiple processors.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
2014.12.15	14.1.0	<ul style="list-style-type: none"> <li>• Updated location of Fitter Settings, Analysis &amp; Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.</li> <li>• Updated DSE II content.</li> </ul>
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Minor changes for HardCopy.
		<i>continued...</i>



Document Version	Intel Quartus Prime Version	Changes
May 2013	13.0.0	Added the information about initial compilation requirements. This section was moved from the Area Optimization chapter of the Intel Quartus Prime Handbook. Minor updates to delineate division of Timing and Area optimization chapters.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.3	Template update.
December 2010	10.0.2	Changed to new document template. No change to content.
August 2010	10.0.1	Corrected link
July 2010	10.0.0	Initial release. Chapter based on topics and text in Section III of volume 2.

## 2. Optimizing the Design Netlist

---

This chapter describes how you can use the Intel Quartus Prime Netlist Viewers to analyze and debug your designs.

As FPGA designs grow in size and complexity, the ability to analyze, debug, optimize, and constrain your design is critical. With today's advanced designs, several design engineers are involved in coding and synthesizing different design blocks, making it difficult to analyze and debug the design. The Intel Quartus Prime RTL Viewer and Technology Map Viewer provide powerful ways to view your initial and fully mapped synthesis results during the debugging, optimization, and constraint entry processes.

### Related Information

- [Intel Quartus Prime Design Flow with the Netlist Viewers](#) on page 14
- [RTL Viewer Overview](#) on page 15
- [Technology Map Viewer Overview](#) on page 16
- [Filtering in the Schematic View](#) on page 28
- [Viewing a Timing Path](#) on page 36

### 2.1. When to Use the Netlist Viewers: Analyzing Design Problems

You can use the Netlist Viewers to analyze and debug your design. The following simple examples show how to use the RTL Viewer and Technology Map Viewer to analyze problems encountered in the design process.

Using the RTL Viewer is a good way to view your initial synthesis results to determine whether you have created the necessary logic, and that the logic and connections have been interpreted correctly by the software. You can use the RTL Viewer to check your design visually before simulation or other verification processes. Catching design errors at this early stage of the design process can save you valuable time.

If you see unexpected behavior during verification, use the RTL Viewer to trace through the netlist and ensure that the connections and logic in your design are as expected. Viewing your design helps you find and analyze the source of design problems. If your design looks correct in the RTL Viewer, you know to focus your analysis on later stages of the design process and investigate potential timing violations or issues in the verification flow itself.

You can use the Technology Map Viewer to look at the results at the end of Analysis and Synthesis. If you have compiled your design through the Fitter stage, you can view your post-mapping netlist in the Technology Map Viewer (Post-Mapping) and your post-fitting netlist in the Technology Map Viewer. If you perform only Analysis and Synthesis, both the Netlist Viewers display the same post-mapping netlist.

In addition, you can use the RTL Viewer or Technology Map Viewer to locate the source of a particular signal, which can help you debug your design. Use the navigation techniques described in this chapter to search easily through your design. You can trace back from a point of interest to find the source of the signal and ensure the connections are as expected.

The Technology Map Viewer can help you locate post-synthesis nodes in your netlist and make assignments when optimizing your design. This functionality is useful when making a multicycle clock timing assignment between two registers in your design. Start at an I/O port and trace forward or backward through the design and through levels of hierarchy to find nodes of interest, or locate a specific register by visually inspecting the schematic.

Throughout your FPGA design, debug, and optimization stages, you can use all of the netlist viewers in many ways to increase your productivity while analyzing a design.

**Related Information**

- [Intel Quartus Prime Design Flow with the Netlist Viewers](#) on page 14
- [RTL Viewer Overview](#) on page 15
- [Technology Map Viewer Overview](#) on page 16

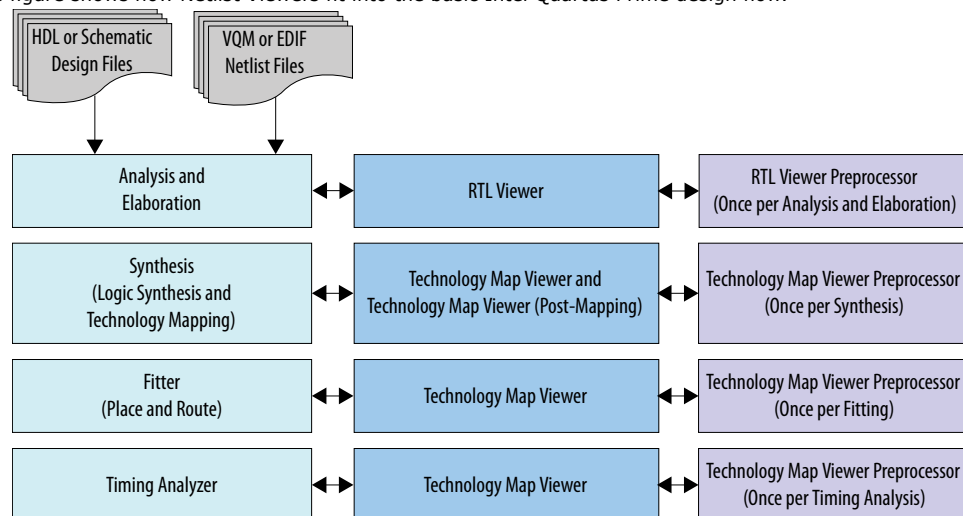
## 2.2. Intel Quartus Prime Design Flow with the Netlist Viewers

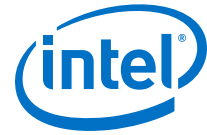
When you first open one of the Netlist Viewers after compiling the design, a preprocessor stage runs automatically before the Netlist Viewer opens.

Click the link in the preprocessor process box to go to the **Settings > Compilation Process Settings** page where you can turn on the **Run Netlist Viewers preprocessing during compilation** option. If you turn this option on, the preprocessing becomes part of the full project compilation flow and the Netlist Viewer opens immediately without displaying the preprocessing dialog box.

**Figure 1. Intel Quartus Prime Design Flow Including the RTL Viewer and Technology Map Viewer**

This figure shows how Netlist Viewers fit into the basic Intel Quartus Prime design flow.





Before the Netlist Viewer can run the preprocessor stage, you must compile your design:

- To open the RTL Viewer first perform Analysis and Elaboration.
- To open the Technology Map Viewer (Post-Fitting) or the Technology Map Viewer (Post-Mapping), first perform Analysis and Synthesis.

The Netlist Viewers display the results of the last successful compilation.

- Therefore, if you make a design change that causes an error during Analysis and Elaboration, you cannot view the netlist for the new design files, but you can still see the results from the last successfully compiled version of the design files.
- If you receive an error during compilation and you have not yet successfully run the appropriate compilation stage for your project, the Netlist Viewer cannot be displayed; in this case, the Intel Quartus Prime software issues an error message when you try to open the Netlist Viewer.

*Note:* If the Netlist Viewer is open when you start a new compilation, the Netlist Viewer closes automatically. You must open the Netlist Viewer again to view the new design netlist after compilation completes successfully.

## 2.3. RTL Viewer Overview

The RTL Viewer allows you to view a register transfer level (RTL) graphical representation of Intel Quartus Prime Pro Edition synthesis results or third-party netlist files in the Intel Quartus Prime software.

You can view results after Analysis and Elaboration for designs that use any supported Intel Quartus Prime design entry method, including Verilog HDL Design Files (.v), SystemVerilog Design Files (.sv), VHDL Design Files (.vhdl), AHDL Text Design Files (.tdf), or schematic Block Design Files (.bdf).

You can also view the hierarchy of atom primitives (such as device logic cells and I/O ports) for designs that generate Verilog Quartus Mapping File (.vqm) or Electronic Design Interchange Format (.edf) files through a synthesis tool.

The RTL Viewer displays a schematic view of the design netlist after Analysis and Elaboration or after the Intel Quartus Prime software performs netlist extraction, but before technology mapping and synthesis or fitter optimizations. This view a preliminary pre-optimization design structure and closely represents the original source design.

- For designs synthesized with the Intel Quartus Prime Pro Edition synthesis, this view shows how the Intel Quartus Prime software interprets the design files.
- For designs synthesized with a third-party synthesis tool, this view shows the netlist that the synthesis tool generates.

To run the RTL Viewer for an Intel Quartus Prime project, first analyze the design to generate an RTL netlist. To analyze the design and generate an RTL netlist, click **Processing > Start > Start Analysis & Elaboration**. You can also perform a full compilation on any process that includes the initial Analysis and Elaboration stage of the Intel Quartus Prime compilation flow.

To open the RTL Viewer, click **Tools > Netlist Viewers > RTL Viewer**.

### Related Information

[Netlist Viewer User Interface](#) on page 17

## 2.3.1. Maximizing Readability in RTL Viewer

While displaying a design, the RTL Viewer optimizes the netlist to maximize readability:

- Removes logic with no fan-out (unconnected output) or fan-in (unconnected inputs) from the display.
- Hides default connections such as  $V_{CC}$  and GND.
- Groups pins, nets, wires, module ports, and certain logic into buses where appropriate.
- Groups constant bus connections.
- Displays values in hexadecimal format.
- Converts NOT gates into bubble inversion symbols in the schematic.
- Merges chains of equivalent combinational gates into a single gate; for example, a 2-input AND gate feeding a 2-input AND gate is converted to a single 3-input AND gate.

## 2.3.2. Running the RTL Viewer

To run the RTL Viewer for an Intel Quartus Prime project:

1. Analyze the design to generate an RTL netlist by clicking **Processing > Start > Start Analysis & Elaboration**.

You can also perform a full compilation on any process that includes the initial Analysis and Elaboration stage of the Intel Quartus Prime compilation flow.

2. Open the RTL Viewer by clicking **Tools > Netlist Viewers > RTL Viewer**.

## 2.4. Technology Map Viewer Overview

The Intel Quartus Prime Technology Map Viewer provides a technology-specific, graphical representation of FPGA designs after Analysis and Synthesis or after the Fitter maps the design into the target device.

The Technology Map Viewer shows the hierarchy of atom primitives (such as device logic cells and I/O ports) in the design. For supported device families, you can also view internal registers and look-up tables (LUTs) inside logic cells (LCELLs), and registers in I/O atom primitives.

Where possible, the Intel Quartus Prime software maintains the port names of each hierarchy throughout synthesis. However, the software may change or remove port names from the design. For example, the software removes ports that are unconnected or driven by GND or  $V_{CC}$  during synthesis. If a port name changes, the software assigns a related user logic name in the design or a generic port name such as IN1 or OUT1.

You can view Intel Quartus Prime technology-mapped results after synthesis, fitting, or timing analysis. To run the Technology Map Viewer for an Intel Quartus Prime project, on the **Processing** menu, point to **Start** and click **Start Analysis & Synthesis** to synthesize and map the design to the target technology. At this stage,





The Technology Map Viewer shows the same post-mapping netlist as the Technology Map Viewer (Post-Mapping). You can also perform a full compilation, or any process that includes the synthesis stage in the compilation flow.

For designs that completed the Fitter stage, the Technology Map Viewer shows how the Fitter changed the netlist through physical synthesis optimizations, while the Technology Map Viewer (Post-Mapping) shows the post-mapping netlist. If you have completed the Timing Analysis stage, you can locate timing paths from the Timing Analyzer report in the Technology Map Viewer.

To open the Technology Map Viewer, click **Tools > Netlist Viewers > Technology Map Viewer (Post-Fitting)** or **Technology Map Viewer (Post Mapping)**.

#### Related Information

- [Viewing a Timing Path](#) on page 36
- [View Contents of Nodes in the Schematic View](#) on page 28
- [Netlist Viewer User Interface](#) on page 17

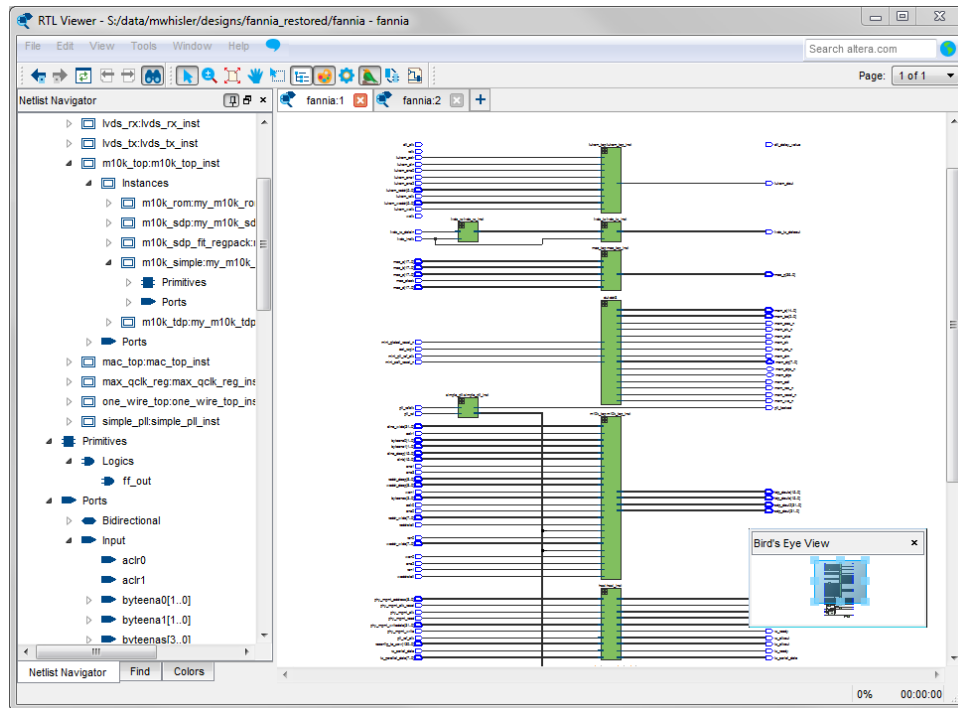
## 2.5. Netlist Viewer User Interface

The Netlist Viewer is a graphical user-interface for viewing and manipulating nodes and nets in the netlist.

The RTL Viewer and Technology Map Viewer each consist of these main parts:

- The **Netlist Navigator** pane—displays a representation of the project hierarchy.
- The **Find** pane—allows you to find and locate specific design elements in the schematic view.
- The **Properties** pane displays the properties of the selected block when you select **Properties** from the shortcut menu.
- The schematic view—displays a graphical representation of the internal structure of the design.

Figure 2. RTL Viewer



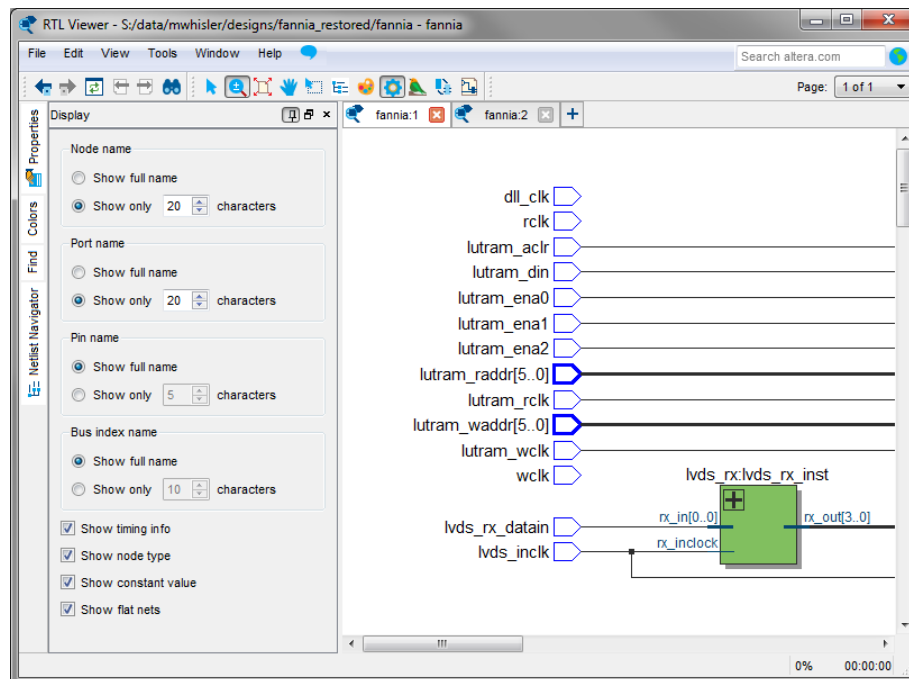
Netlist Viewers also contain a toolbar that provides tools to use in the schematic view.

- Use the **Back** and **Forward** buttons to switch between schematic views. You can go forward only if you have not made any changes to the view since going back. These commands do not undo an action, such as selecting a node. The Netlist Viewer caches up to ten actions including filtering, hierarchy navigation, netlist navigation, and zoom actions.
- The **Refresh** button to restore the schematic view and optimizes the layout. **Refresh** does not reload the database if you change the design and recompile.
- Click the **Find** button opens and closes the **Find** pane.
- Click the **Selection Tool** and **Zoom Tool** buttons to alternate between the selection mode and zoom mode.
- Click the **Fit in Page** button resets the schematic view to encompass the entire design.
- Use the **Hand Tool** to change the focus of the viewer without changing the perspective.
- Click the **Area Selection Tool** to drag a selection box around ports, pins, and nodes in an area.
- Click the **Netlist Navigator** button to open or close the **Netlist Navigator** pane.
- Click the **Color Settings** button to open the **Colors** pane where you can customize the Netlist Viewer color scheme.



- Click the **Display Settings** button to open the **Display** pane where you can specify the following settings:
  - **Show full name** or **Show only <n> characters**. You can specify this separately for **Node name**, **Port name**, **Pin name**, or **Bus name**.
  - Turn **Show timing info** on or off.
  - Turn **Show node type** on or off.
  - Turn **Show constant value** on or off.
  - Turn **Show flat nets** on or off.

Figure 3. Display Settings



- The **Bird's Eye View** button opens the **Bird's Eye View** window which displays a miniature version of the design and allows you to navigate within the design and adjust the magnification in the schematic view quickly.
- The **Show/Hide Instance Pins** button can alternate the display of instance pins not displayed by functions such as cross-probing between a Netlist Viewer and Timing Analyzer. You can also use this button to hide unconnected instance pins when filtering a node results in large numbers of unconnected or unused pins. The Netlist Viewer hides Instance pins by default.
- If the Netlist Viewer display encompasses several pages, the **Show Netlist on One Page** button resizes the netlist view to a single page. This action can make netlist tracing easier.

You can have only one RTL Viewer, one Technology Map Viewer (Post-Fitting), and one Technology Map Viewer (Post-Mapping) window open at the same time, although each window can show multiple pages, each with multiple tabs. For example, you cannot have two RTL Viewer windows open at the same time.

### Related Information

- [RTL Viewer Overview](#) on page 15
- [Technology Map Viewer Overview](#) on page 16
- [Netlist Navigator Pane](#) on page 20
- [Netlist Viewers Find Pane](#) on page 22
- [Properties Pane](#) on page 20

## 2.5.1. Netlist Navigator Pane

The **Netlist Navigator** pane displays the entire netlist in a tree format based on the hierarchical levels of the design. Each level groups similar elements into subcategories.

The **Netlist Navigator** pane allows you to traverse through the design hierarchy to view the logic schematic for each level. You can also select an element in the **Netlist Navigator** to highlight in the schematic view.

*Note:* The **Netlist Navigator** pane does not list nodes inside atom primitives.

For each module in the design hierarchy, the **Netlist Navigator** pane displays the applicable elements listed in the following table. Click the “+” icon to expand an element.

**Table 4. Netlist Navigator Pane Elements**

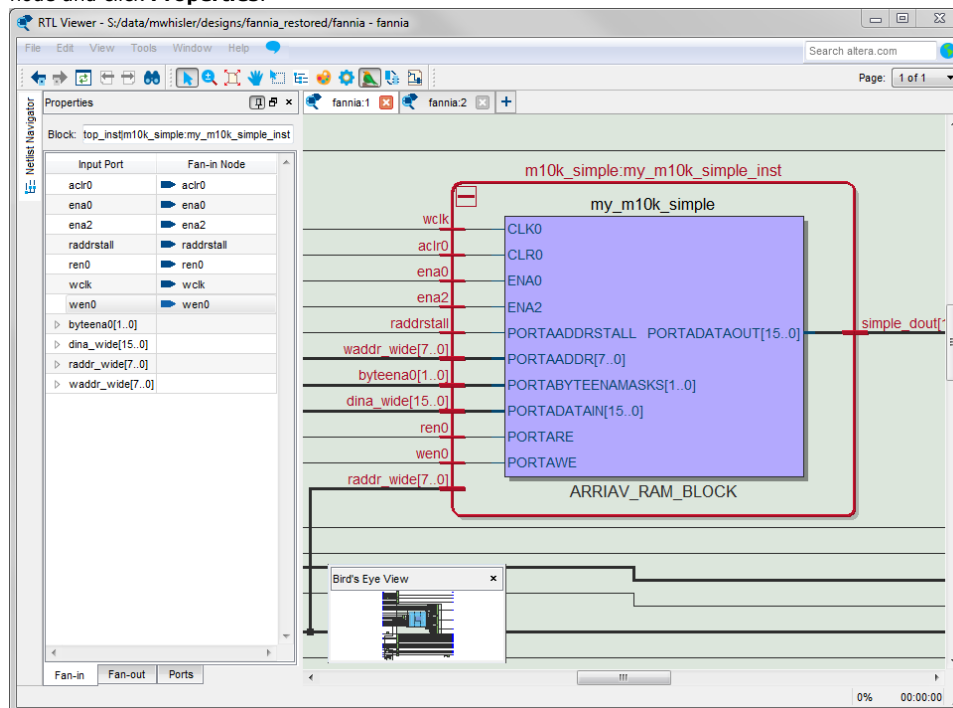
Elements	Description
Instances	Modules or instances in the design that can be expanded to lower hierarchy levels.
Primitives	<p>Low-level nodes that cannot be expanded to any lower hierarchy level. These primitives include:</p> <ul style="list-style-type: none"> <li>• Registers and gates that you can view in the RTL Viewer when using Intel Quartus Prime Pro Edition synthesis.</li> <li>• Logic cell atoms in the Technology Map Viewer or in the RTL Viewer when using a VQM or EDIF from third-party synthesis software</li> </ul> <p>In the Technology Map Viewer, you can view the internal implementation of certain atom primitives, but you cannot traverse into a lower-level of hierarchy.</p>
Ports	<p>The I/O ports in the current level of hierarchy.</p> <ul style="list-style-type: none"> <li>• Pins are device I/O pins when viewing the top hierarchy level and are I/O ports of the design when viewing the lower-levels.</li> <li>• When a pin represents a bus or an array of pins, expand the pin entry in the list view to see individual pin names.</li> </ul>

## 2.5.2. Properties Pane

You can view the properties of an instance or primitive with the **Properties** pane.

**Figure 4. Properties Pane**

To view the properties of an instance or primitive in the RTL Viewer or Technology Map Viewer, right-click the node and click **Properties**.



The **Properties** pane contains tabs with the following information about the selected node:

- The **Fan-in** tab displays the **Input port** and **Fan-in Node**.
- The **Fan-out** tab displays the **Output port** and **Fan-out Node**.
- The **Parameters** tab displays the **Parameter Name** and **Values** of an instance.
- The **Ports** tab displays the **Port Name** and **Constant** value (for example,  $V_{CC}$  or GND). The following table lists the possible values of a port:

**Table 5. Possible Port Values**

Value	Description
$V_{CC}$	The port is not connected and has $V_{CC}$ value (tied to $V_{CC}$ )
GND	The port is not connected and has GND value (tied to GND)
--	The port is connected and has value (other than $V_{CC}$ or GND)
Unconnected	The port is not connected and has no value (hanging)

If the selected node is an atom primitive, the **Properties** pane displays a schematic of the internal logic.

### 2.5.3. Netlist Viewers Find Pane

You can narrow the range of the search process by setting the following options in the **Find** pane:

- Click **Browse** in the **Find** pane to specify the hierarchy level of the search. In the **Select Hierarchy Level** dialog box, select the particular instance you want to search.
- Turn on the **Include subtentities** option to include child hierarchies of the parent instance during the search.
- Click **Options** to open the **Find Options** dialog box. Turn on **Instances**, **Nodes**, **Ports**, or any combination of the three to further refine the parameters of the search.

When you click the **List** button, a progress bar appears below the **Find** box.

All results that match the criteria you set are listed in a table. When you double-click an item in the table, the related node is highlighted in red in the schematic view.

## 2.6. Schematic View

The schematic view is shown on the right side of the RTL Viewer and Technology Map Viewer. The schematic view contains a schematic representing the design logic in the netlist. This view is the main screen for viewing your gate-level netlist in the RTL Viewer and your technology-mapped netlist in the Technology Map Viewer.

The RTL Viewer and Technology Map Viewer attempt to display schematic in a single page view by default. If the schematic crosses over to several pages, you can highlight a net and use connectors to trace the signal in a single page.

### 2.6.1. Display Schematics in Multiple Tabbed View

The RTL Viewer and Technology Map Viewer support multiple tabbed views.

With multiple tabbed view, schematics can be displayed in different tabs. Selection is independent between tabbed views, but selection in the tab in focus is synchronous with the Netlist Navigator pane.

To create a new blank tab, click the **New Tab** button at the end of the tab row . You can now drag a node from the **Netlist Navigator** pane into the schematic view.

Right-click in a tab to see a shortcut menu to perform the following actions:

- Create a blank view with **New Tab**
- Create a **Duplicate Tab** of the tab in focus
- Choose to **Cascade Tabs**
- Choose to **Tile Tabs**
- Choose **Close Tab** to close the tab in focus
- Choose **Close Other Tabs** to close all tabs except the tab in focus

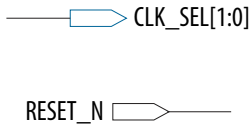
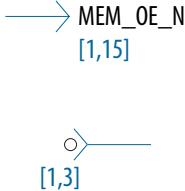
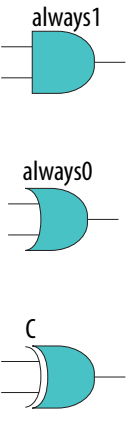
## 2.6.2. Schematic Symbols

The symbols for nodes in the schematic represent elements of your design netlist. These elements include input and output ports, registers, logic gates, Intel primitives, high-level operators, and hierarchical instances.

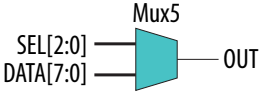
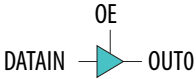
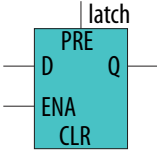
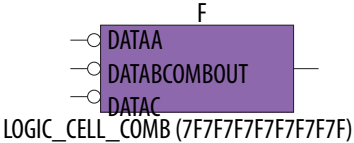
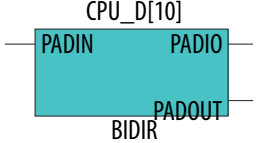
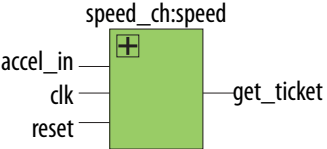
**Note:** The logic gates and operator primitives appear only in the RTL Viewer. Logic in the Technology Map Viewer is represented by atom primitives, such as registers and LCELLs.

**Table 6. Symbols in the Schematic View**

This table lists and describes the primitives and basic symbols that you can display in the schematic view of the RTL Viewer and Technology Map Viewer.

Symbol	Description
<p>I/O Ports</p> 	<p>An input, output, or bidirectional port in the current level of hierarchy. A device input, output, or bidirectional pin when viewing the top-level hierarchy. The symbol can also represent a bus. Only one wire is shown connected to the bidirectional symbol, representing the input and output paths. Input symbols appear on the left-most side of the schematic. Output and bidirectional symbols appear on the right-most side of the schematic.</p>
<p>I/O Connectors</p> 	<p>An input or output connector, representing a net that comes from another page of the same hierarchy. To go to the page that contains the source or the destination, double-click the connector to jump to the appropriate page.</p>
<p>OR, AND, XOR Gates</p> 	<p>An OR, AND, or XOR gate primitive (the number of ports can vary). A small circle (bubble symbol) on an input or output port indicates the port is inverted.</p>
<p>MULTIPLEXER</p>	<p>A multiplexer primitive with a selector port that selects between port 0 and port 1. A multiplexer with more than two inputs is displayed as an operator.</p>

*continued...*

Symbol	Description
	
<p>BUFFER</p> 	<p>A buffer primitive. The figure shows the tri-state buffer, with an inverted output enable port. Other buffers without an enable port include LCELL, SOFT, and GLOBAL. The NOT gate and EXP expander buffers use this symbol without an enable port and with an inverted output port.</p>
<p>LATCH</p> 	<p>A latch/DFF (data flipflop) primitive. A DFF has the same ports as a latch and a clock trigger. The other flipflop primitives are similar:</p> <ul style="list-style-type: none"> <li>• DFFE (data flipflop with enable and asynchronous load) primitive with additional <code>ALOAD</code> asynchronous load and <code>ADATA</code> data signals</li> <li>• DFFEAS (data flipflop with enable and synchronous and asynchronous load), which has <code>ASDATA</code> as the secondary data port</li> </ul>
<p>Atom Primitive</p> 	<p>An atom primitive. The symbol displays the atom name, the port names, and the atom type. The blue shading indicates an atom primitive for which you can view the internal details.</p>
<p>Other Primitive</p> 	<p>Any primitive that does not fall into the previous categories. Primitives are low-level nodes that cannot be expanded to any lower hierarchy. The symbol displays the port names, the primitive or operator type, and its name.</p>
<p>Instance</p> 	<p>An instance in the design that does not correspond to a primitive or operator (a user-defined hierarchy block). The symbol displays the port name and the instance name.</p>
<p>Encrypted Instance</p>	<p>A user-defined encrypted instance in the design. The symbol displays the instance name. You cannot open the schematic for the lower-level hierarchy, because the source design is encrypted.</p>

*continued...*



Symbol	Description
<p>RAM</p>	<p>A synchronous memory instance with registered inputs and optionally registered outputs. The symbol shows the device family and the type of memory block. This figure shows a true dual-port memory block in a Stratix M-RAM block.</p>
<p>Constant</p>	<p>A constant signal value that is highlighted in gray and displayed in hexadecimal format by default throughout the schematic.</p>

**Table 7. Operator Symbols in the RTL Viewer Schematic View**

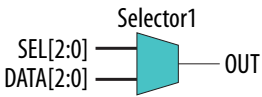
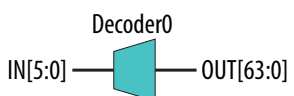
The following lists and describes the additional higher level operator symbols in the RTL Viewer schematic view.

Symbol	Description
	<p>An adder operator:  <math>OUT = A + B</math></p>

*continued...*

Symbol	Description
	<p>A multiplier operator:  <math>OUT = A \times B</math></p>
	<p>A divider operator:  <math>OUT = A / B</math></p>
	<p>Equals</p>
	<p>A left shift operator:  <math>OUT = (A \ll COUNT)</math></p>
	<p>A right shift operator:  <math>OUT = (A \gg COUNT)</math></p>
	<p>A modulo operator:  <math>OUT = (A \% B)</math></p>
	<p>A less than comparator:  <math>OUT = (A &lt; B : A &gt; B)</math></p>
	<p>A multiplexer:  <math>OUT = DATA [SEL]</math>            The data range size is <math>2^{\text{sel range size}}</math></p>

*continued...*

Symbol	Description
	<p>A selector: A multiplexer with one-hot select input and more than two input signals</p>
	<p>A binary number decoder:  <math>OUT = (binary\_number(IN) == x)</math>  for x = 0 to  <math>x = 2^{(n+1)} - 1</math></p>

#### Related Information

- [Partition the Schematic into Pages](#) on page 32
- [Follow Nets Across Schematic Pages](#) on page 33

### 2.6.3. Select Items in the Schematic View

To select an item in the schematic view, ensure that the **Selection Tool** is enabled in the Netlist Viewer toolbar. Click an item in the schematic view to highlight in red.

Select multiple items by pressing the Shift key while selecting with the mouse.

Items selected in the schematic view are automatically selected in the **Netlist Navigator** pane. The folder then expands automatically if it is required to show the selected entry; however, the folder does not collapse automatically when you deselected the entries.

When you select a hierarchy box, node, or port in the schematic view, the Schematic View highlights the item in red, but not the connecting nets. When you select a net (wire or bus) in the schematic view, the Schematic View highlights all connected nets in red.

Once you select an item, you can perform different actions on it based on the contents of the shortcut menu which appears when you right-click your selection.

#### Related Information

[Netlist Navigator Pane](#) on page 20

### 2.6.4. Shortcut Menu Commands in the Schematic View

When you right-click a selected instance or primitive in the schematic view, the Netlist Viewer displays a shortcut menu.

If the selected item is a node, you see the following options:

- Click **Expand to Upper Hierarchy** to displays the parent hierarchy of the node in focus.
- Click **Copy ToolTip** to copy the selected item name to the clipboard. This command does not work on nets.
- Click **Hide Selection** to remove the selected item from the schematic view. This command does not delete the item from the design, merely masks it in the current view.
- Click **Filtering** to display a sub-menu with options for filtering your selection.

### 2.6.5. Filtering in the Schematic View

Filtering allows you to filter out nodes and nets in a netlist to view only the logic elements of interest to you.

You can filter a netlist by selecting hierarchy boxes, nodes, or ports of a node, that are part of the path you want to see. The following filter commands are available:

- **Sources**—displays the sources of the selection.
- **Destinations**—displays the destinations of the selection.
- **Sources & Destinations**—displays the sources and destinations of the selection.
- **Selected Nodes**—displays only the selected nodes.
- **Between Selected Nodes**—displays nodes and connections in the path between the selected nodes.
- **Bus Index**—Displays the sources or destinations for one or more indexes of an output or input bus port.
- **Filtering Options**—Displays the **Filtering Options** dialog box:
  - **Stop filtering at register**—Turning on this option directs the Netlist Viewer to filter out to the nearest register boundary.
  - **Filter across hierarchies**—Turning on this option directs the Netlist Viewer to filter across hierarchies.
  - **Maximum number of hierarchy levels**—Sets the maximum number of hierarchy levels that the schematic view can display.

To filter a netlist, select a hierarchy box, node, port, net, or state node, right-click in the window, point to **Filter** and click the appropriate filter command. The Netlist Viewer generates a new page showing the netlist that remains after filtering.

### 2.6.6. View Contents of Nodes in the Schematic View

In the RTL Viewer and the Technology Map Viewer, you can view the contents of nodes to see their underlying implementation details.

You can view LUTs, registers, and logic gates. You can also view the implementation of RAM and DSP blocks in certain devices in the RTL Viewer or Technology Map Viewer. In the Technology Map Viewer, you can view the contents of primitives to see their underlying implementation details.

**Figure 5. Wrapping and Unwrapping Objects**

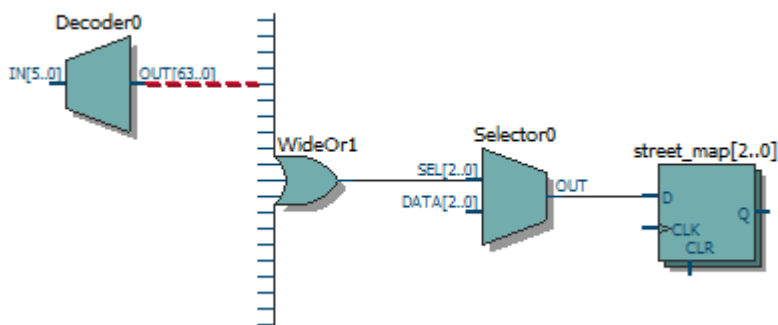
If you can unwrap the contents of an instance, a plus symbol appears in the upper right corner of the object in the schematic view. To wrap the contents (and revert to the compact format), click the minus symbol in the upper right corner of the unwrapped instance.



*Note:* In the schematic view, the internal details in an atom instance cannot be selected as individual nodes. Any mouse action on any of the internal details is treated as a mouse action on the atom instance.

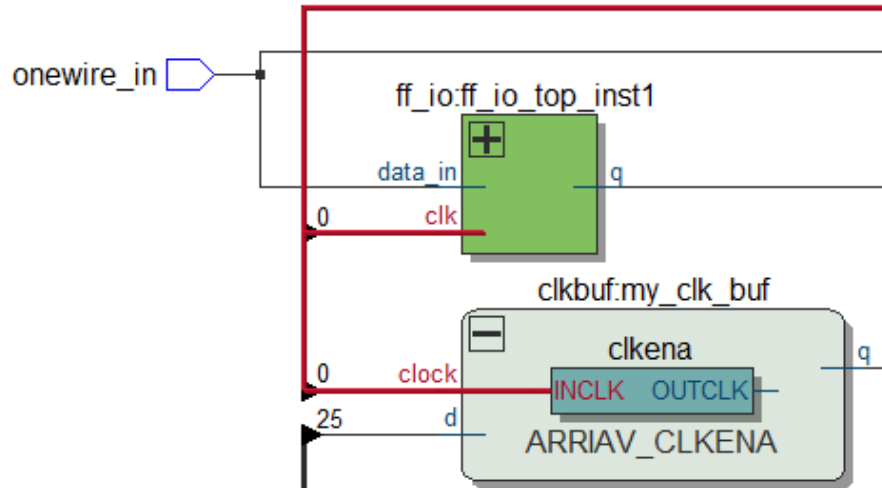
**Figure 6. Nodes with Connections Outside the Hierarchy**

In some cases, the selected instance connects to something outside the visible level of the hierarchy in the schematic view. In this case, the net appears as a dotted line. Double-click the dotted line to expand the view to display the destination of the connection .



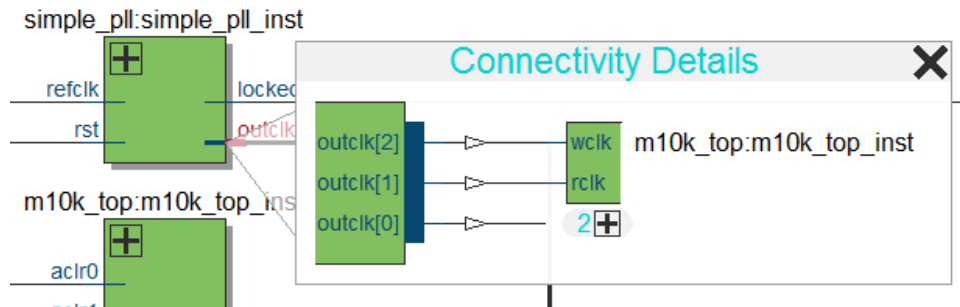
**Figure 7. Display Nets Across Hierarchies**

In cases where the net connects to an instance outside the hierarchy, you can select the net, and unwrap the node to see the destination ports.



**Figure 8. Show Connectivity Details**

You can select a bus or bus pin and click **Connectivity Details** in the context menu for that object.



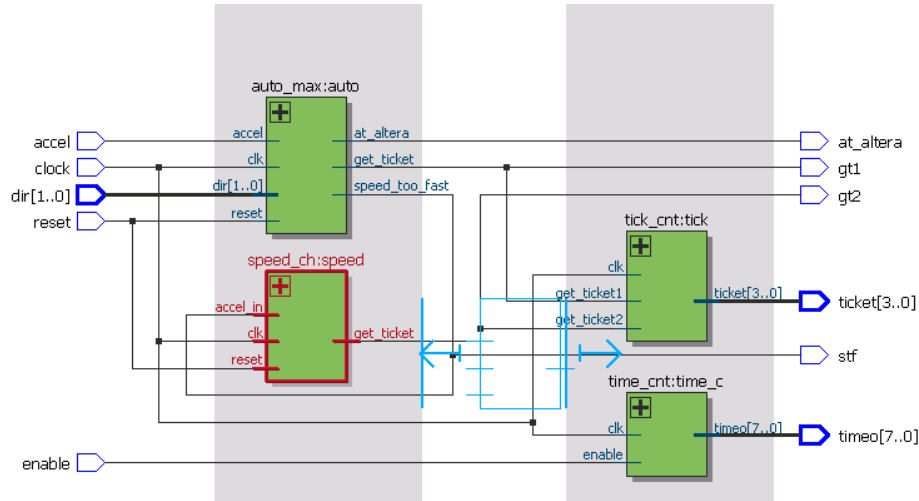
You can double-click objects in the **Connectivity Details** window to navigate to them quickly. If the plus symbol appears, you can further unwrap objects in the view. This can be very useful when tracing a signal in a complex netlist.

### 2.6.7. Moving Nodes in the Schematic View

Rearrange items in the schematic view by dragging to destination.

To move a node from one area of the netlist to another, select the node and hold down the Shift key. Legal placements appear as shaded areas within the hierarchy. Click to drop the selected node.

**Figure 9. Legal Placement when Moving Nodes**



To restore the schematic view to its default arrangement, right-click and click **Refresh**.

### 2.6.8. View LUT Representations in the Technology Map Viewer

You can view different representations of a LUT by right-clicking the selected LUT and clicking **Properties**.

You can view the LUT representations in the following three tabs in the **Properties** dialog box:

- The **Schematic** tab—the equivalent gate representations of the LUT.
- The **Truth Table** tab—the truth table representations.

#### Related Information

[Properties Pane](#) on page 20

### 2.6.9. Zoom Controls

Use the Zoom Tool in the toolbar, or mouse gestures, to control the magnification of your schematic on the View menu.

By default, the Netlist Viewer displays most pages sized to fit in the window. If the schematic page is very large, the schematic is displayed at the minimum zoom level, and the view is centered on the first node. Click **Zoom In** to view the image at a larger size, and click **Zoom Out** to view the image (when the entire image is not displayed) at a smaller size. The **Zoom** command allows you to specify a magnification percentage (100% is considered the normal size for the schematic symbols).

You can use the Zoom Tool on the Netlist Viewer toolbar to control magnification in the schematic view. When you select the Zoom Tool in the toolbar, clicking in the schematic zooms in and centers the view on the location you clicked. Right-click in the schematic to zoom out and center the view on the location you clicked. When you

select the Zoom Tool, you can also zoom into a certain portion of the schematic by selecting a rectangular box area with your mouse cursor. The schematic is enlarged to show the selected area.

Within the schematic view, you can also use the following mouse gestures to zoom in on a specific section:

- **zoom in**—Dragging a box around an area starting in the upper-left and dragging to the lower right zooms in on that area.
- **zoom -0.5**—Dragging a line from lower-left to upper-right zooms out 0.5 levels of magnification.
- **zoom 0.5**—Dragging a line from lower-right to upper-left zooms in 0.5 levels of magnification.
- **zoom fit**—Dragging a line from upper-right to lower-left fits the schematic view in the page.

#### Related Information

[Filtering in the Schematic View](#) on page 28

### 2.6.10. Navigating with the Bird's Eye View

To open the Bird's Eye View, on the View menu, click **Bird's Eye View**, or click the **Bird's Eye View** icon in the toolbar.

Viewing the entire schematic can be useful when debugging and tracing through a large netlist. The Intel Quartus Prime software allows you to quickly navigate to a specific section of the schematic using the Bird's Eye View feature, which is available in the RTL Viewer and Technology Map Viewer.

The Bird's Eye View shows the current area of interest:

- Select an area by clicking and dragging the indicator or right-clicking to form a rectangular box around an area.
- Click and drag the rectangular box to move around the schematic.
- Resize the rectangular box to zoom-in or zoom-out in the schematic view.

### 2.6.11. Partition the Schematic into Pages

For large design hierarchies, the RTL Viewer and Technology Map Viewer partition your netlist into multiple pages in the schematic view.

When a hierarchy level is partitioned into multiple pages, the title bar for the schematic window indicates which page is displayed and how many total pages exist for this level of hierarchy. The schematic view displays this as **Page** <current page number> **of** <total number of pages>.

#### Related Information

[Netlist Viewer User Interface](#) on page 17





## 2.6.12. Follow Nets Across Schematic Pages

Input and output connector symbols indicate nodes that connect across pages of the same hierarchy. Double-click a connector to trace the net to the next page of the hierarchy.

**Note:** After you double-click to follow a connector port, the Netlist Viewer opens a new page, which centers the view on the particular source or destination net using the same zoom factor as the previous page. To trace a specific net to the new page of the hierarchy, Intel recommends that you first select the necessary net, which highlights it in red, before you double-click to navigate across pages.

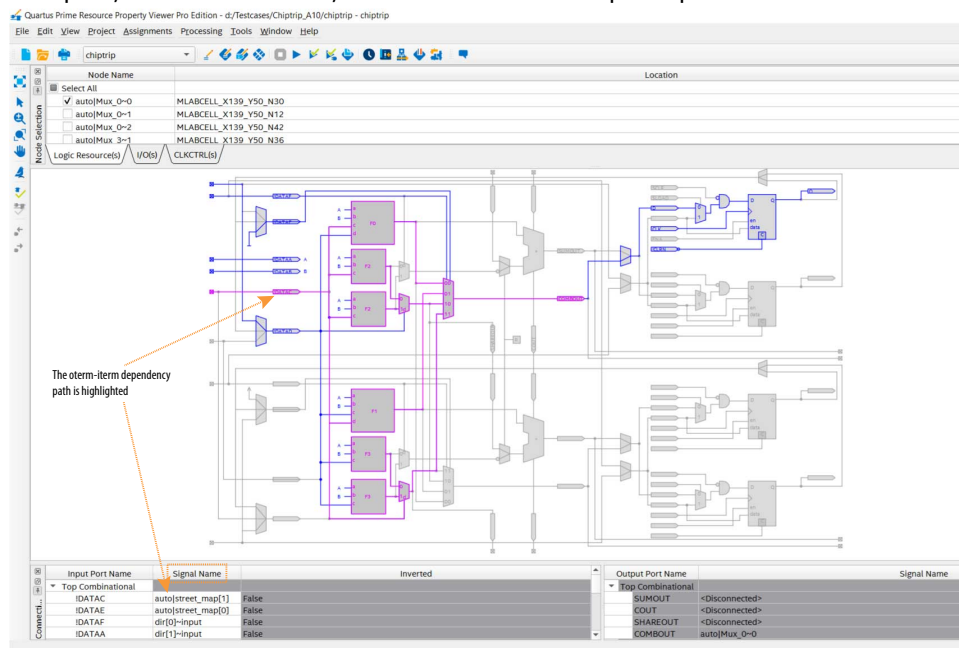
### Related Information

[Schematic Symbols](#) on page 23

## 2.6.13. Maintaining Selection in the Resource Property Viewer

Prior to 19.1 release, when you hovered the mouse over a port in the Resource Property Viewer to visualize the item-to-item dependency path of a node, the dependency path would get highlighted. However, the highlight would disappear once you moved the mouse to another point.

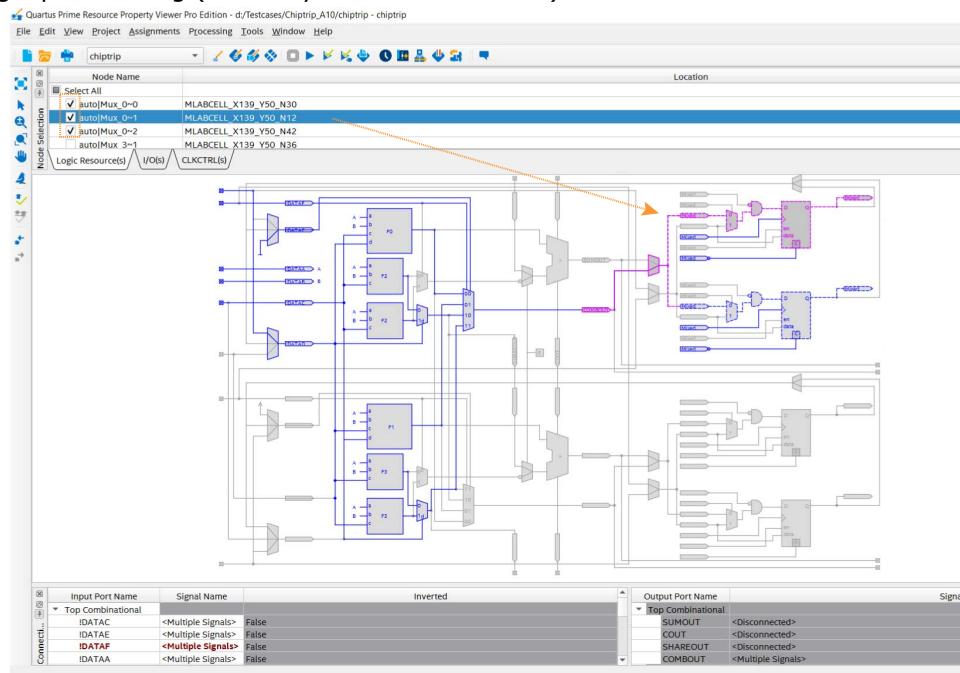
From 19.1 release, when you click on a port (in the schematic) or the signal name (in the Connectivity pane), the path is selected. Each node keeps only a dependency path. This dependency path is cleared when you recompile, close the project, select another port, run a new locate, or left-click on a non-port space.



Note:

- Every double-click on a node in the Chip Planner to open the Resource Property Viewer is considered as a new locate.
- For the path-tracing purpose, using the **Go To Destination Node** or **Go To Source Node** option adds more nodes to the Node Selection table, and does not clear the path-dependency selection (as opposed to every new locate, for example, double-clicking on nodes in the Chip Planner).
- The dependency path selection or highlight is not applicable to Node Properties in the Chip Planner.

When you select a port or a signal name of the multiple selection, the dependency path is updated only if the port is used. If the port is unused, the dependency path selection is cleared. However, the Resource Property Viewer does not display the dependency path selection for multiple node selections. It serves the objective of single path tracing (with only one selection color).



## 2.7. Cross-Probing to a Source Design File and Other Intel Quartus Prime Windows

The RTL Viewer and Technology Map Viewer allow you to cross-probe to the source design file and to various other windows in the Intel Quartus Prime software.

You can select one or more hierarchy boxes, nodes, state nodes, or state transition arcs that interest you in the Netlist Viewer and locate the corresponding items in another applicable Intel Quartus Prime software window. You can then view and make changes or assignments in the appropriate editor or floorplan.



To locate an item from the Netlist Viewer in another window, right-click the items of interest in the schematic or state diagram, point to **Locate**, and click the appropriate command. The following commands are available:

- **Locate in Assignment Editor**
- **Locate in Pin Planner**
- **Locate in Chip Planner**
- **Locate in Resource Property Editor**
- **Locate in Technology Map Viewer**
- **Locate in RTL Viewer**
- **Locate in Design File**

The options available for locating an item depend on the type of node and whether it exists after placement and routing. If a command is enabled in the menu, it is available for the selected node. You can use the **Locate in Assignment Editor** command for all nodes, but assignments might be ignored during placement and routing if they are applied to nodes that do not exist after synthesis.

The Netlist Viewer automatically opens another window for the appropriate editor or floorplan and highlights the selected node or net in the newly opened window. You can switch back to the Netlist Viewer by selecting it in the Window menu or by closing, minimizing, or moving the new window.

## 2.8. Cross-Probing to the Netlist Viewers from Other Intel Quartus Prime Windows

You can cross-probe to the RTL Viewer and Technology Map Viewer from other windows in the Intel Quartus Prime software. You can select one or more nodes or nets in another window and locate them in one of the Netlist Viewers.

You can locate nodes between the RTL Viewer and Technology Map Viewer, and you can locate nodes in the RTL Viewer and Technology Map Viewer from the following Intel Quartus Prime software windows:

- Project Navigator
- Timing Closure Floorplan
- Chip Planner
- Resource Property Editor
- Node Finder
- Assignment Editor
- Messages Window
- Compilation Report
- Timing Analyzer (supports the Technology Map Viewer only)

To locate elements in the Netlist Viewer from another Intel Quartus Prime window, select the node or nodes in the appropriate window; for example, select an entity in the **Entity** list on the **Hierarchy** tab in the Project Navigator, or select nodes in the Timing Closure Floorplan, or select node names in the **From** or **To** column in the Assignment Editor. Next, right-click the selected object, point to **Locate**, and click



**Locate in RTL Viewer** or **Locate in Technology Map Viewer**. After you click this command, the Netlist Viewer opens, or is brought to the foreground if the Netlist Viewer is open.

**Note:** The first time the window opens after a compilation, the preprocessor stage runs before the Netlist Viewer opens.

The Netlist Viewer shows the selected nodes and, if applicable, the connections between the nodes. The display is similar to what you see if you right-click the object, then click **Filter > Selected Nodes** using **Filter across hierarchy**. If the nodes cannot be found in the Netlist Viewer, a message box displays the message: **Can't find requested location**.

## 2.9. Viewing a Timing Path

After completing a full design compilation, including the timing analyzer stage, you can see a visual representation of a timing path cross-probe from a timing report. For details about generating the timing report, refer to the *Intel Quartus Prime Pro Edition User Guide: Timing Analyzer*.

When you locate the timing path from the Timing Analyzer to the Technology Map Viewer, the interconnect and cell delay associated with each node appears on top of the schematic symbols. The total slack of the selected timing path appears in the Page Title section of the schematic.

1. To open the report from the **Compilation Report** Table of Contents, click **Timing Analyzer GUI > Report Timing**, and double-click the timing corner.
2. To open the report from the **Timing Analyzer**, open the **Report Timing** folder in the **Report** pane, and double-click the timing corner.
3. In the **Summary of Paths** tab, right-click a row in the table and select **Locate Path > Locate in Technology Map Viewer**. In the Technology Map Viewer, the schematic page displays the nodes along the timing path with a summary of the total delay.

### Related Information

[Report Timing \(Dialog Box\)](#)

In *Intel Quartus Prime Pro Edition User Guide: Timing Analyzer*

## 2.10. Optimizing the Design Netlist Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2019.07.01	19.1	Added <i>Maintaining Selection in the Resource Property Viewer</i> topic explaining how the item-term dependency is maintained in the schematic view.
2018.09.24	18.1.0	<ul style="list-style-type: none"> <li>• Added link to <i>Viewing a Timing Path</i>.</li> <li>• Removed reference to unsupported CARRY buffer from "Schematic Symbols" topic.</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>• Implemented Intel rebranding.</li> </ul>
2016.05.03	16.0.0	Removed Schematic Viewer topic.
<i>continued...</i>		



Document Version	Intel Quartus Prime Version	Changes
2015.11.02	15.1.0	Added Schematic Viewer topic for viewing stage snapshots. Added information for the following new features and feature updates: <ul style="list-style-type: none"> <li>• Nets visible across hierarchies</li> <li>• Connection Details</li> <li>• Display Settings</li> <li>• Hand Tool</li> <li>• Area Selection Tool</li> <li>• New default behavior for Show/Hide Instance Pins (default is now off)</li> </ul>
2014.06.30	14.0.0	Added Show Netlist on One Page and show/Hide Instance Pins commands.
November 2013	13.1.0	Removed HardCopy device information. Reorganized and migrated to new template. Added support for new Netlist viewer.
November 2012	12.1.0	Added sections to support Global Net Routing feature.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Changed to new document template.
July 2010	10.0.0	<ul style="list-style-type: none"> <li>• Updated screenshots</li> <li>• Updated chapter for the Intel Quartus Prime software version 10.0, including major user interface changes</li> </ul>
November 2009	9.1.0	<ul style="list-style-type: none"> <li>• Updated devices</li> <li>• Minor text edits</li> </ul>
March 2009	9.0.0	<ul style="list-style-type: none"> <li>• Chapter 13 was formerly Chapter 12 in version 8.1.0</li> <li>• Updated Figure 13-2, Figure 13-3, Figure 13-4, Figure 13-14, and Figure 13-30</li> <li>• Added "Enable or Disable the Auto Hierarchy List" on page 13-15</li> <li>• Updated "Find Command" on page 13-44</li> </ul>
November 2008	8.1.0	Changed page size to 8.5" × 11"
May 2008	8.0.0	<ul style="list-style-type: none"> <li>• Added Arria GX support</li> <li>• Updated operator symbols</li> <li>• Updated information about the radial menu feature</li> <li>• Updated zooming feature</li> <li>• Updated information about probing from schematic to Signal Tap Analyzer</li> <li>• Updated constant signal information</li> <li>• Added .png and .gif to the list of supported image file formats</li> <li>• Updated several figures and tables</li> <li>• Added new sections "Enabling and Disabling the Radial Menu", "Changing the Time Interval", "Changing the Constant Signal Value Formatting", "Logic Clouds in the RTL Viewer", "Logic Clouds in the Technology Map Viewer", "Manually Group and Ungroup Logic Clouds", "Customizing the Shortcut Commands"</li> <li>• Renamed several sections</li> <li>• Removed section "Customizing the Radial Menu"</li> <li>• Moved section "Grouping Combinational Logic into Logic Clouds"</li> <li>• Updated document content based on the Intel Quartus Prime software version 8.0</li> </ul>

## 3. Netlist Optimizations and Physical Synthesis

The Intel Quartus Prime software offers netlist optimizations during synthesis, and physical synthesis optimization during fitting, that can improve the performance of your design. Synthesis netlist optimizations operate with the atom netlist of your design, which describes a design in terms of specific primitives. This chapter provides guidelines for applying synthesis and physical synthesis optimization settings.

You can access a range of global synthesis and physical synthesis optimization settings from the **Compiler Settings** page:

**Table 8. Synthesis Netlist Optimization and Physical Synthesis Options**

Options	Location/Description
Enable synthesis netlist optimization settings	Enable synthesis optimization options (for example, <b>Synthesis Effort</b> ) in the <b>Advanced Synthesis Settings</b> dialog box. Click <b>Assignments &gt; Settings &gt; Compiler Settings &gt; Advanced Settings (Synthesis)</b> to access these options.
Enable physical synthesis options	Enable physical synthesis options (for example, <b>Advanced Physical Synthesis</b> ) in the <b>Advanced Fitter Settings</b> dialog box. Click <b>Assignments &gt; Settings &gt; Compiler Settings &gt; Advanced Settings (Fitter)</b> to access these settings.

**Note:** Because the node names for primitives in the design can change when you use physical synthesis optimizations, you should evaluate whether your design depends on fixed node names. If you use a verification flow that might require fixed node names, such as the Signal Tap Logic Analyzer, formal verification, or the Logic Lock based optimization flow (for legacy devices), disable physical synthesis options.

### 3.1. Physical Synthesis Optimizations

The Intel Quartus Prime Fitter places and routes the logic cells to ensure critical portions of logic are close together and use the fastest possible routing resources. However, routing delays are often a significant part of the typical critical path delay. Physical synthesis optimizations take into consideration placement information, routing delays, and timing information to determine the optimal placement. The Fitter then focuses timing-driven optimizations at those critical parts of the design. The tight integration of the synthesis and fitting processes is known as physical synthesis.

The following sections describe the physical synthesis optimizations available in the Intel Quartus Prime software, and how they can help improve performance and fitting for the selected device.

#### Related Information

[Compiler Settings Page \(Settings Dialog Box\)](#)  
In *Intel Quartus Prime Help*



### 3.1.1. Enabling Physical Synthesis Optimization

Physical synthesis optimization improves circuit performance by performing combinational and sequential optimization and register duplication.

To enable physical synthesis options:

1. Click **Assignments** > **Settings** > **Compiler Settings**.
2. To enable retiming, combinational optimization, and register duplication, click **Advanced Settings (Fitter)**. Next, enable **Physical Synthesis**.
3. View physical synthesis results in the **Netlist Optimizations** report.

### 3.1.2. Physical Synthesis Options

The Intel Quartus Prime software provides physical synthesis optimization options to improve fitting results. To access these options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

*Note:* To disable global physical synthesis optimizations for specific elements of your design, assign the **Netlist Optimizations** logic option to **Never Allow** to the specific nodes or entities.

**Table 9. Physical Synthesis Options**

Option	Description
<b>Advanced Physical Synthesis</b>	Uses the physical synthesis engine to perform combinational and sequential optimization during fitting to improve circuit performance.
<b>Netlist Optimizations</b>	You can use the Assignment Editor to apply the <b>Netlist Optimizations</b> logic option. Use this option to disable physical synthesis optimizations for parts of your design.
<b>Allow Register Duplication</b>	Allows the Compiler to duplicate registers to improve design performance. When you enable this option, the Compiler copies registers and moves some fan-out to this new node. This optimization improves routability and can reduce the total routing wire in nets with many fan-outs. If you disable this option, this disables optimizations that retime registers. This setting affects Analysis & Synthesis and the Fitter.
<b>Allow Register Merging</b>	Allows the Compiler to remove registers that are identical to other registers in the design. When you enable this option, in cases where two registers generate the same logic, the Compiler deletes one register, and the remaining registers fan-out to the deleted register's destinations. This option is useful if you want to prevent the Compiler from removing intentional use of duplicate registers. If you disable register merging, the Compiler disables optimizations that retime registers. This setting affects Analysis & Synthesis and the Fitter.

## 3.2. Applying Netlist Optimizations

The improvement in performance when using netlist optimizations is design dependent. If you have restructured your design to balance critical path delays, netlist optimizations might yield minimal improvement in performance.

You may have to experiment with available options to see which combination of settings works best for a particular design. Refer to the messages in the compilation report to see the magnitude of improvement with each option, and to help you decide whether you should turn on a given option or specific effort level.

Turning on more netlist optimization options can result in more changes to the node names in the design; bear this in mind if you are using a verification flow, such as the Signal Tap Logic Analyzer or formal verification that requires fixed or known node names.

To find the best results, you can use the Intel Quartus Prime Design Space Explorer II (DSE) to apply various sets of netlist optimization options.

#### Related Information

[Optimize Settings with Design Space Explorer II](#) on page 91

### 3.2.1. WYSIWYG Primitive Resynthesis

For designs synthesized with a third-party tool, the **Perform WYSIWYG primitive resynthesis** option allows you to apply optimizations to the synthesized netlist.

The **Perform WYSIWYG primitive resynthesis** option directs the Intel Quartus Prime software to un-map the logic elements (LEs) in an atom netlist to logic gates, and then re-map the gates back to Intel-specific primitives. Third-party synthesis tools generate either an `.edf` or `.vqm` atom netlist file using Intel-specific primitives. When you turn on the **Perform WYSIWYG primitive resynthesis** option, the Intel Quartus Prime software uses device-specific techniques during the re-mapping process. This feature re-maps the design using the **Optimization Technique** specified for your project (**Speed**, **Area**, or **Balanced**).

The **Perform WYSIWYG primitive resynthesis** option unmaps and remaps only logic cells, also referred to as LCELL or LE primitives, and regular I/O primitives (which may contain registers). Double data rate (DDR) I/O primitives, memory primitives, digital signal processing (DSP) primitives, and logic cells in carry chains are not remapped. This process does not process logic specified in an encrypted `.vqm` file or an `.edf` file, such as third-party intellectual property (IP).

The **Perform WYSIWYG primitive resynthesis** option can change node names in the `.vqm` file or `.edf` file from your third-party synthesis tool, because the primitives in the atom netlist are broken apart and then re-mapped by the Intel Quartus Prime software. The re-mapping process removes duplicate registers. Registers that are not removed retain the same name after re-mapping.

Any nodes or entities that have the **Netlist Optimizations** logic option set to **Never Allow** are not affected during WYSIWYG primitive resynthesis. You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. This option disables WYSIWYG resynthesis for parts of your design.

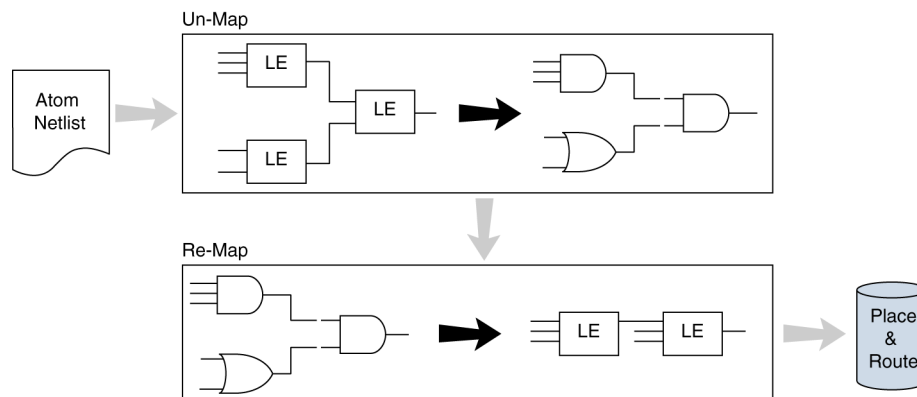
#### Note:

Primitive node names are specified during synthesis. When netlist optimizations are applied, node names might change because primitives are created and removed. HDL attributes applied to preserve logic in third-party synthesis tools cannot be maintained because those attributes are not written into the atom netlist, which the Intel Quartus Prime software reads.

If you use the Intel Quartus Prime software to synthesize your design, you can use the **Preserve Register (preserve)** and **Keep Combinational Logic (keep)** attributes to maintain certain nodes in the design.



Figure 10. Intel Quartus Prime Flow for WYSIWYG Primitive Resynthesis



### 3.3. Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section on either an instance or global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <QSF variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <QSF variable name> <value> \ -to <instance name>
```

#### Related Information

- [Command Line Scripting](#)
- [Tcl Scripting](#)
- [API Functions for Tcl](#)  
In *Intel Quartus Prime Help*
- [Intel Quartus Prime Pro Edition Settings File Reference Manual](#)  
For information about all settings and constraints in the Intel Quartus Prime software.

#### 3.3.1. Synthesis Netlist Optimizations

The project .qsf file preserves the settings that you specify in the GUI. Alternatively, you can edit the .qsf directly. The .qsf file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

**Table 10. Synthesis Netlist Optimizations and Associated Settings**

Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Optimization Mode	OPTIMIZATION_MODE	BALANCED HIGH PERFORMANCE EFFORT AGGRESSIVE PERFORMANCE	Global, Instance
Power-Up Don't Care	ALLOW_POWER_UP_DONT_CARE	ON, OFF	Global

### 3.3.2. Physical Synthesis Optimizations

The project .qsf file preserves the settings that you specify in the GUI. Alternatively, you can edit the .qsf directly. The .qsf file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

**Table 11. Physical Synthesis Optimizations and Associated Settings**

Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Advanced Physical Synthesis	ADVANCED_PHYSICAL_SYNTHESIS	ON, OFF	Global

### 3.4. Netlist Optimizations and Physical Synthesis Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2019.04.24	18.1.0	Updated example in "Netlist Optimizations and Physical Synthesis" topic.
2019.04.18	18.1.0	Clarified wording in "Netlist Optimizations and Physical Synthesis" topic.
2018.09.24	18.1.0	Removed reference to unsupported CASCADE buffer from "Optimize IOC Register Placement for Timing Logic Option" topic.
2018.05.07	18.0.0	Removed topic: <i>Isolating a Partition Netlist</i> .
2017.11.06	17.1.0	<ul style="list-style-type: none"> <li>Removed reference to .vqm files</li> <li>Added topic: <i>Isolating a Partition Netlist</i>.</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>Implemented Intel rebranding.</li> <li>Updated physical synthesis options and procedure.</li> </ul>
2016.05.02	16.0.0	<ul style="list-style-type: none"> <li>Removed information about deprecated physical synthesis options.</li> </ul>
2015.11.02	15.1.0	<ul style="list-style-type: none"> <li>Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.</li> <li>Added Physical Synthesis.</li> </ul>
2014.12.15	14.1.0	<ul style="list-style-type: none"> <li>Updated location of Fitter Settings, Analysis &amp; Synthesis Settings, and Physical Synthesis Optimizations Settings to Compiler Settings.</li> <li>Updated DSE II content.</li> </ul>
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Removed HardCopy device information.
<i>continued...</i>		

### 3. Netlist Optimizations and Physical Synthesis

UG-20133 | 2020.09.28



Document Version	Intel Quartus Prime Version	Changes
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	<ul style="list-style-type: none"><li>Added links to Intel Quartus Prime Help in several sections.</li><li>Removed Referenced Documents section.</li><li>Reformatted Document Revision History</li></ul>
November 2009	9.1.0	<ul style="list-style-type: none"><li>Added information to "Physical Synthesis for Registers—Register Retiming"</li><li>Added information to "Applying Netlist Optimization Options"</li><li>Made minor editorial updates</li></ul>
March 2009	9.0.0	<ul style="list-style-type: none"><li>Was chapter 11 in the 8.1.0 release.</li><li>Updated the "Physical Synthesis for Registers—Register Retiming" and "Physical Synthesis Options for Fitting"</li><li>Updated "Performing Physical Synthesis Optimizations"</li><li>Deleted Gate-Level Register Retiming section.</li><li>Updated the referenced documents</li></ul>
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	<ul style="list-style-type: none"><li>Updated "Physical Synthesis Optimizations for Performance on page 11-9"</li><li>Added Physical Synthesis Options for Fitting on page 11-16</li></ul>

## 4. Area Optimization

This chapter describes techniques to reduce resource usage when designing for Intel devices.

### 4.1. Resource Utilization Information

Determining device utilization provides useful information regardless of whether the design achieved a successful fit. If the compilation results in a no-fit error, resource utilization information helps to analyze the fitting problems in the design. If the fitting is successful, this information allows you to determine if design changes introduce fitting difficulties. Additionally, you can determine the impact of the resource utilization in the timing performance.

The Compilation Report provides information about resource usage.

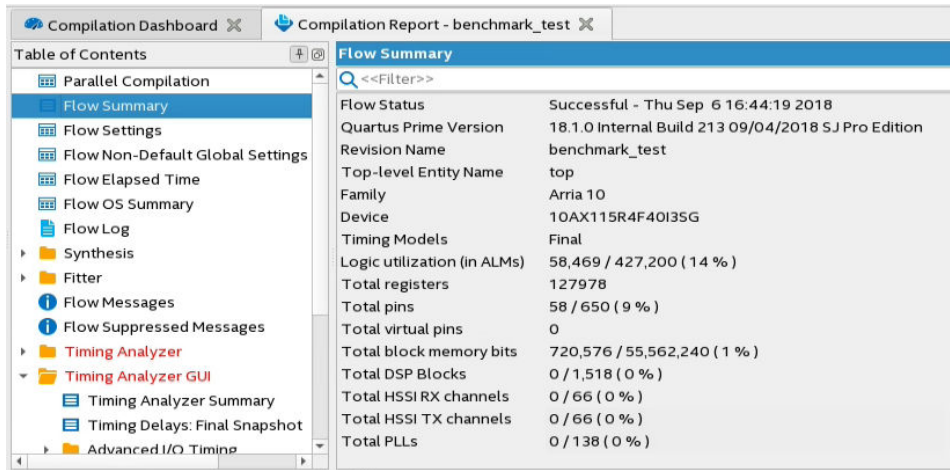
#### Related Information

[Viewing Routing Congestion](#) on page 126

#### 4.1.1. Flow Summary Report

The **Flow Summary** section of the compilation report indicates whether the design exceeds the available device resources, and reports resource utilization, including pins, memory bits, digital signal processing (DSP) blocks, and phase-locked loops (PLLs).

**Figure 11. Flow Summary Report**



Flow Summary	
Flow Status	Successful - Thu Sep 6 16:44:19 2018
Quartus Prime Version	18.1.0 Internal Build 213 09/04/2018 SJ Pro Edition
Revision Name	benchmark_test
Top-level Entity Name	top
Family	Arria 10
Device	10AX115R4F40I35G
Timing Models	Final
Logic utilization (in ALMs)	58,469 / 427,200 ( 14 % )
Total registers	127978
Total pins	58 / 650 ( 9 % )
Total virtual pins	0
Total block memory bits	720,576 / 55,562,240 ( 1 % )
Total DSP Blocks	0 / 1,518 ( 0 % )
Total HSSI RX channels	0 / 66 ( 0 % )
Total HSSI TX channels	0 / 66 ( 0 % )
Total PLLs	0 / 138 ( 0 % )

The Fitter can spread logic throughout the device, which may lead to higher overall utilization.



As the device fills up, the Fitter automatically searches for logic functions with common inputs to place in one ALM. The number of packed registers also increases. Therefore, a design that has high overall utilization might still have space for extra logic if the logic and registers can be packed together more tightly. In those cases, you can benefit by a report that provides more details.

### 4.1.2. Fitter Reports

In the **Fitter** section of the compilation report, reports under **Resource Section** provide detailed resource information.

The **Fitter Resource Usage Summary** report breaks down the logic utilization information and provides additional resource information, including the number of bits in each type of memory block. This panel also contains a summary of the usage of global clocks, PLLs, DSP blocks, and other device-specific resources.

#### Related Information

[Fitter Resources Reports](#)

### 4.1.3. Analysis and Synthesis Reports

For designs synthesized with the Intel Quartus Prime synthesis engine, you can see reports describing optimizations that occurred during compilation.

For example, in the **Analysis & Synthesis** section, **Optimization Results** folder, you can find a list of registers removed during synthesis. With this report you can estimate resource utilization for partial designs so you make sure that registers were not removed due to missing connections with other parts of the design.

#### Related Information

[Synthesis Optimization Results Reports](#)

### 4.1.4. Compilation Messages

If the reports show routing resource usage lower than 100% but the design does not fit, either routing resources are insufficient or the design contains invalid assignments. In either case, the Compiler generates a message in the **Processing** tab of the **Messages** window describing the problem.

If the Fitter finishes unsuccessfully and runs much faster than on similar designs, a resource might be over-utilized or there might be an illegal assignment.

If the Intel Quartus Prime software takes too long to run when compared to similar designs possibly the Compiler is not able to find valid placement or route. In the Compilation Report, look for errors and warnings that indicate these types of problems.

The Chip Planner can help you find areas of the device that have routing congestion for specific types of routing resources. If you find areas with very high congestion, analyze the cause of the congestion. Issues such as high fan-out nets not using global resources, an improperly chosen optimization goal (speed versus area), very restrictive floorplan assignments, or the coding style can cause routing congestion. After you identify the cause, modify the source or settings to reduce routing congestion.

## Related Information

[Viewing Messages](#)

## 4.2. Optimizing Resource Utilization

The following lists the stages after design analysis:

1. Optimize resource utilization—Ensure that you have already set the basic constraints
2. I/O timing optimization—Optimize I/O timing after you optimize resource utilization and your design fits in the desired target device
3. Register-to-register timing optimization

### Related Information

- [Design Optimization Overview](#) on page 6
- [Timing Closure and Optimization](#) on page 56

### 4.2.1. Resource Utilization Issues Overview

Resource utilization issues can be divided into three categories:

- Issues relating to *I/O pin utilization or placement*, including dedicated I/O blocks such as PLLs or LVDS transceivers.
- Issues relating to *logic utilization or placement*, including logic cells containing registers and LUTs as well as dedicated logic, such as memory blocks and DSP blocks.
- Issues relating to *routing*.

### 4.2.2. I/O Pin Utilization or Placement

Resolve I/O resource problems with these guidelines.

#### 4.2.2.1. Guideline: Modify Pin Assignments or Choose a Larger Package

If a design that has pin assignments fails to fit, compile the design without the pin assignments to determine whether a fit is possible for the design in the specified device and package. You can use this approach if an Intel Quartus Prime error message indicates fitting problems due to pin assignments.

If the design fits when all pin assignments are ignored or when several pin assignments are ignored or moved, you might have to modify the pin assignments for the design or select a larger package.

If the design fails to fit because insufficient I/Os pins are available, a larger device package (which can be the same device density) that has more available user I/O pins can result in a successful fit.

### Related Information

[Managing Device I/O Pins](#)

In *Intel Quartus Prime Pro Edition User Guide: Design Constraints*



### 4.2.3. Logic Utilization or Placement

Resolve logic resource problems, including logic cells containing registers and LUTs, as well as dedicated logic such as memory blocks and DSP blocks, with these guidelines.

#### 4.2.3.1. Guideline: Optimize Source Code

If your design does not fit because of logic utilization, then evaluate and modify the design at the source. You can often improve logic significantly by making design-specific changes to your source code. This is typically the most effective technique for improving the quality of your results.

If your design does not fit into available logic elements (LEs) or ALMs, but you have unused memory or DSP blocks, check if you have code blocks in your design that describe memory or DSP functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to allow these functions to be placed into dedicated memory or DSP resources in the target device.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Intel Quartus Prime software, you can check for the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including the state encoding for each state machine that was recognized during compilation. If your state machine is not being recognized, you might have to change your source code to enable it to be recognized.

#### Related Information

- [AN 584: Timing Closure Methodology for Advanced FPGA Designs](#)
- [Recommended HDL Coding Styles](#)  
In *Intel Quartus Prime Pro Edition User Guide: Design Recommendations*

#### 4.2.3.2. Guideline: Optimize Synthesis for Area, Not Speed

If the Fitter cannot resolve a design due to limitations in logic resources, resynthesize the design to improve the area utilization.

First, ensure that the device and timing constraints are set correctly in the synthesis tool. Particularly when area utilization of the design is a concern, ensure that you do not over-constrain the timing requirements for the design. Synthesis tools try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is an important concern, you can optimize for area instead of speed.

- If you are using Intel Quartus Prime synthesis, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)** and select **Balanced** or **Area** for the **Optimization Technique**.
- If you want to reduce area for specific modules in the design using the **Area** or **Speed** setting while leaving the default **Optimization Technique** setting at **Balanced**, use the Assignment Editor.
- You can also turn on the **Speed Optimization Technique for Clock Domains** logic option to optimize for speed all combinational logic in or between the specified clock domains.
- In some synthesis tools, not specifying an  $f_{MAX}$  requirement can result in less resource utilization.

Optimizing for area or speed can affect the register-to-register timing performance.

*Note:* In the Intel Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.

The Intel Quartus Prime software provides additional attributes and options that can help improve the quality of the synthesis results.

#### Related Information

[Optimization Mode](#)

#### 4.2.3.3. Guideline: Restructure Multiplexers

Multiplexers form a large portion of the logic utilization in many FPGA designs. By optimizing your multiplexed logic, you can achieve a more efficient implementation in your Intel device.

#### Related Information

- [Restructure Multiplexers logic option](#)  
For more information about the Restructure Multiplexers option
- [Recommended HDL Coding Styles](#)  
For design guidelines to achieve optimal resource utilization for multiplexer designs

#### 4.2.3.4. Guideline: Perform WYSIWYG Primitive Resynthesis with Balanced or Area Setting

The **Perform WYSIWYG Primitive Resynthesis** logic option specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the **Optimization Technique** logic option. The **Perform WYSIWYG Primitive Resynthesis** logic option is useful for resynthesizing some or all of the WYSIWYG primitives in your design for better area or performance. However, WYSIWYG primitive resynthesis can be done only when you use third-party synthesis tools.





*Note:* The **Balanced** setting typically produces utilization results that are very similar to the **Area** setting with better performance results. The **Area** setting can give better results in some cases. Performing WYSIWYG resynthesis for area in this way typically reduces register-to-register timing performance.

#### Related Information

[Perform WYSIWYG Primitive Resynthesis logic option](#)  
For information about this logic option

### 4.2.3.5. Guideline: Use Register Packing

The **Auto Packed Registers** option implements the functions of two cells into one logic cell by combining the register of one cell in which only the register is used with the LUT of another cell in which only the LUT is used.

#### Related Information

[Auto Packed Registers logic option](#)  
For more information about the Auto Packed Registers logic option

### 4.2.3.6. Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or Logic Lock assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or Logic Lock region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or Logic Lock region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and Logic Lock assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove Logic Lock assignments in the Chip Planner, in the Logic Lock Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

#### Related Information

[Analyzing and Optimizing the Design Floorplan](#) on page 121

### 4.2.3.7. Guideline: Flatten the Hierarchy During Synthesis

Synthesis tools typically provide the option of preserving hierarchical boundaries, which can be useful for verification or other purposes. However, the Intel Quartus Prime software optimizes across hierarchical boundaries so as to perform the most logic minimization, which can reduce area in a design with no design partitions.

### 4.2.3.8. Guideline: Re-target Memory Blocks

If the Fitter cannot resolve a design due to memory resource limitations, the design may require a type of memory that the device does not have.

For memory blocks created with the Parameter Editor, edit the RAM block type to target a new memory block size.

The Compiler can also infer ROM and RAM memory blocks from the HDL code, and the synthesis engine can place large shift registers into memory blocks by inferring the Shift register (RAM-based) IP core. When you turn off this inference in the synthesis tool, the synthesis engine places the memory or shift registers in logic instead of memory blocks. Also, turning off this inference prevents registers from being moved into RAM, improving timing performance,

Depending on the synthesis tool, you can also set the RAM block type for inferred memory blocks. In Intel Quartus Prime synthesis, set the **ramstyle** attribute to the desired memory type for the inferred RAM blocks. Alternatively, set the option to **logic** to implement the memory block in standard logic instead of a memory block.

Consider the Resource Utilization by Entity report in the report file and determine whether there is an unusually high register count in any of the modules. Some coding styles prevent the Intel Quartus Prime software from inferring RAM blocks from the source code because of the blocks' architectural implementation, forcing the software to implement the logic in flip-flops. For example, an asynchronous reset on a register bank might make the register bank incompatible with the RAM blocks in the device architecture, so Compiler implements the register bank in flip-flops. It is often possible to move a large register bank into RAM by slight modification of associated logic.

#### Related Information

- [Inferring Shift Registers in HDL Code](#)
- [Fitter Resource Utilization by Entity Report](#)

#### 4.2.3.9. Guideline: Use Physical Synthesis Options to Reduce Area

The physical synthesis options available at **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)** help you decrease resource usage. When you enable physical synthesis, the Intel Quartus Prime software makes placement-specific changes to the netlist that reduce resource utilization for a specific Intel device.

*Note:* Physical synthesis increases compilation time. To reduce the impact on compilation time, you can apply physical synthesis options to specific instances.

#### Related Information

[Advanced Fitter Settings Dialog Box](#)

#### 4.2.3.10. Guideline: Retarget or Balance DSP Blocks

A design might not fit because it requires too many DSP blocks. You can implement all DSP block functions with logic cells, so you can retarget some of the DSP blocks to logic to obtain a fit.

If the DSP function was created with the parameter editor, open the parameter editor and edit the function so it targets logic cells instead of DSP blocks. The Intel Quartus Prime software uses the `DEDICATED_MULTIPLIER_CIRCUITRY` IP core parameter to control the implementation.



DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. You can turn off this inference in your synthesis tool. When you are using Intel Quartus Prime synthesis, you can disable inference by turning off the **Auto DSP Block Replacement** logic option for your entire project. Click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**. Turn off **Auto DSP Block Replacement**. Alternatively, you can disable the option for a specific block with the Assignment Editor.

The Intel Quartus Prime software also offers the **DSP Block Balancing** logic option, which implements DSP block elements in logic cells or in different DSP block modes. The default **Auto** setting allows DSP block balancing to convert the DSP block slices automatically as appropriate to minimize the area and maximize the speed of the design. You can use other settings for a specific node or entity, or on a project-wide basis, to control how the Intel Quartus Prime software converts DSP functions into logic cells and DSP blocks. Using any value other than **Auto** or **Off** overrides the `DEDICATED_MULTIPLIER_CIRCUITRY` parameter used in IP core variations.

#### 4.2.3.11. Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

### 4.2.4. Routing

Resolve routing resource problems with these guidelines.

#### 4.2.4.1. Guideline: Set Auto Packed Registers to Sparse or Sparse Auto

The **Auto Packed Registers** option reduces LE or ALM count in a design. You can set this option by clicking **Assignment > Settings > Compiler Settings > Advanced Settings (Fitter)**.

##### Related Information

[Auto Packed Registers logic option](#)

#### 4.2.4.2. Guideline: Set Fitter Aggressive Routability Optimizations to Always

The **Fitter Aggressive Routability Optimization** option is useful if your design does not fit due to excessive routing wire utilization.

If there is a significant imbalance between placement and routing time (during the first fitting attempt), it might be because of high wire utilization. Turning on the **Fitter Aggressive Routability Optimizations** option can reduce your compilation time.

On average, this option can save up to 6% wire utilization, but can also reduce performance by up to 4%, depending on the device.

##### Related Information

[Fitter Aggressive Routability Optimizations logic option](#)

#### 4.2.4.3. Guideline: Increase Router Effort Multiplier

The Router Effort Multiplier controls how quickly the router tries to find a valid solution. The default value is 1.0 and legal values must be greater than 0.

- Numbers higher than 1 help designs that are difficult to route by increasing the routing effort.
- Numbers closer to 0 (for example, 0.1) can reduce router runtime, but usually reduce routing quality slightly.

Experimental evidence shows that a multiplier of 3.0 reduces overall wire usage by approximately 2%. Using a Router Effort Multiplier higher than the default value can benefit designs with complex datapaths with more than five levels of logic. However, congestion in a design is primarily due to placement, and increasing the Router Effort Multiplier does not necessarily reduce congestion.

*Note:* Any Router Effort Multiplier value greater than 4 only increases by 10% for every additional 1. For example, a value of 10 is actually 4.6.

#### 4.2.4.4. Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or Logic Lock assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or Logic Lock region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or Logic Lock region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and Logic Lock assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove Logic Lock assignments in the Chip Planner, in the Logic Lock Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

#### Related Information

[Analyzing and Optimizing the Design Floorplan](#) on page 121

#### 4.2.4.5. Guideline: Optimize Synthesis for Area, Not Speed

In some cases, resynthesizing the design to improve the area utilization can also improve the routability of the design. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Ensure that you do not over-constrain the timing requirements for the design, particularly when the area utilization of the design is a concern. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.



If resource utilization is an important concern, you can optimize for area instead of speed.

- If you are using Intel Quartus Prime synthesis, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)** and select **Balanced** or **Area** for the **Optimization Technique**.
- If you want to reduce area for specific modules in your design using the **Area** or **Speed** setting while leaving the default **Optimization Technique** setting at **Balanced**, use the Assignment Editor.
- You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.
- In some synthesis tools, not specifying an  $f_{MAX}$  requirement can result in lower resource utilization.

Optimizing for area or speed can affect the register-to-register timing performance.

*Note:* In the Intel Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.

The Intel Quartus Prime software provides additional attributes and options that can help improve the quality of your synthesis results.

#### Related Information

[Optimization Mode](#)

#### 4.2.4.6. Guideline: Optimize Source Code

If your design does not fit because of routing problems and the methods described in the preceding sections do not sufficiently improve the routability of the design, modify the design at the source to achieve the desired results. You can often improve results significantly by making design-specific changes to your source code, such as duplicating logic or changing the connections between blocks that require significant routing resources.

#### 4.2.4.7. Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

### 4.3. Scripting Support

You can run procedures and assign settings described in this chapter in a Tcl script. You can also run procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime command-line and Tcl API Help browser.

1. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.



- Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <QSF variable name> <value>
```

- Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <QSF variable name> <value> \ -to <instance name>
```

**Note:** If the <value> field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

#### Related Information

- [Intel Quartus Prime Pro Settings File Reference Manual](#)  
For information about all settings and constraints in the Intel Quartus Prime software.
- [Tcl Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*
- [Command Line Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*

### 4.3.1. Initial Compilation Settings

Use the Intel Quartus Prime Settings File (.qsf) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

**Table 12. Advanced Compilation Settings**

Setting Name	.qsf File Variable Name	Values	Type
Placement Effort Multiplier	PLACEMENT_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Effort Multiplier	ROUTER_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Timing Optimization level	ROUTER_TIMING_OPTIMIZATION_LEVEL	NORMAL, MINIMUM, MAXIMUM	Global
Final Placement Optimization	FINAL_PLACEMENT_OPTIMIZATION	ALWAYS, AUTOMATICALLY, NEVER	Global

### 4.3.2. Resource Utilization Optimization Techniques

This table lists QSF assignments and applicable values for Resource Utilization Optimization settings:

**Table 13. Resource Utilization Optimization Settings**

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers	QII_AUTO_PACKED_REGISTERS	AUTO, OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, SPARSE, SPARSE AUTO	Global, Instance
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Optimization Technique	OPTIMIZATION_TECHNIQUE	AREA, SPEED, BALANCED	Global, Instance
<i>continued...</i>			

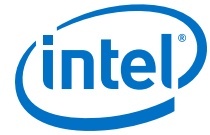


Setting Name	.qsf File Variable Name	Values	Type
Speed Optimization Technique for Clock Domains	SYNTH_CRITICAL_CLOCK	ON, OFF	Instance
State Machine Encoding	STATE_MACHINE_PROCESSING	AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER-ENCODE	Global, Instance
Auto RAM Replacement	AUTO_RAM_RECOGNITION	ON, OFF	Global, Instance
Auto ROM Replacement	AUTO_ROM_RECOGNITION	ON, OFF	Global, Instance
Auto Shift Register Replacement	AUTO_SHIFT_REGISTER_RECOGNITION	ON, OFF	Global, Instance
Auto Block Replacement	AUTO_DSP_RECOGNITION	ON, OFF	Global, Instance
Number of Processors for Parallel Compilation	NUM_PARALLEL_PROCESSORS	Integer between 1 and 16 inclusive, or ALL	Global

## 4.4. Area Optimization Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.10.18	18.1.0	<ul style="list-style-type: none"> <li>Corrected broken link to Optimization Modes Help topic.</li> </ul>
2018.09.24	18.1.0	<ul style="list-style-type: none"> <li>Divided topic: <i>Resource Utilization</i> into topics: <i>Resource Utilization Information</i>, <i>Flow Summary Report</i>, <i>Fitter Reports</i>, <i>Analysis and Synthesis Reports</i>, and <i>Compilation Messages</i>.</li> </ul>
2018.07.03	18.0.0	Fixed typo and added links in topic <i>Guideline: Retarget Memory Blocks</i> .
2017.05.08	17.0.0	<ul style="list-style-type: none"> <li>Removed information about deprecated Integrated Synthesis</li> <li>Revised topics: <i>Resolving Resource Utilization Issues</i>, <i>Guideline: Optimize Synthesis for Area, Not Speed</i></li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>Implemented Intel rebranding.</li> </ul>
2016.05.02	16.0.0	<ul style="list-style-type: none"> <li>Removed information about deprecated physical synthesis options.</li> </ul>
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
2014.12.15	14.1.0	Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.
June 2014	14.0.0	<ul style="list-style-type: none"> <li>Removed Cyclone III and Stratix III devices references.</li> <li>Removed Macrocell-Based CPLDs related information.</li> <li>Updated template.</li> </ul>
May 2013	13.0.0	Initial release.



## 5. Timing Closure and Optimization

---

This chapter describes techniques to improve timing performance when designing for Intel FPGA devices. The application of techniques varies between designs and target FPGA device. Applying each technique does not improve results in all cases.

The default settings and options in the Intel Quartus Prime software provide the most balanced trade-off between compilation time, resource utilization, and timing performance. You can then adjust these settings to determine whether a different mix of settings might provide better results for your design.

### 5.1. Optimize Multi Corner Timing

Process variations and changes in operating conditions can result in path delays that are significantly smaller than those in the slow corner timing model. As a consequence, the design can present hold time violations on those paths, and in rare cases, additional setup time violations.

In addition, designs targeting newer device families (with smaller process geometry) do not always present the slowest circuit performance at the highest operating temperature. The temperature at which the circuit is slowest depends on the selected device, the design, and the compilation results. The Intel Quartus Prime software manages this new dependency by providing newer device families with three different timing corners—Slow 85°C corner, Slow 0°C corner, and Fast 0°C corner. For other device families, two timing corners are available—Fast 0°C and Slow 85°C corner.

The **Optimize multi-corner timing** option directs the Fitter to meet timing requirements at all process corners and operating conditions. The resulting design implementation is more robust across process, temperature, and voltage variations. This option is on by default, and increases compilation time by approximately 10%.

When this option is off, the Fitter optimizes designs considering only slow-corner delays from the slow-corner timing model (slowest manufactured device for a given speed grade, operating in low-voltage conditions).

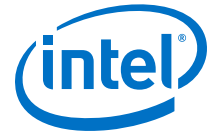
### 5.2. Optimize Critical Paths

Critical paths are timing paths in your design that have a negative slack and may require optimization. These timing paths can span from device I/Os to internal registers, registers to registers, or from registers to device I/Os.

The slack of a path determines its criticality; slack appears in the timing analysis report, which you can generate using the Timing Analyzer.

Design analysis for timing closure is a fundamental requirement for optimal performance in highly complex designs. The analytical capability of the Chip Planner helps you close timing on complex designs.





### Related Information

[Critical Path Delay Reduction Trade-Offs](#) on page 8

## 5.2.1. Viewing Critical Paths

Viewing critical paths in the Chip Planner shows why a specific path is failing. You can see if any modification in the placement can reduce the negative slack. To display paths in the floorplan, perform a timing analysis and display results on the Timing Analyzer.

## 5.3. Optimize Critical Chains

Critical chains are design paths that limit further register retiming optimization. You can use the Hyper-Aware design flow to shorten design cycles and optimize critical chain performance for Intel Stratix 10 and Intel Agilex devices. The Hyper-Aware design flow maximizes use of Hyper-Registers by combining automated register retiming with implementation of targeted timing closure recommendations (Fast Forward compilation). This sum of techniques drive the highest performance for Intel Hyperflex™ architecture designs.

A critical chain reports the design paths that limit further register retiming optimization. The Intel Quartus Prime Pro Edition software provides the Hyper-Retimer critical chain reports to help you improve design performance. You can focus on higher level optimization, because the Hyper-Retimer uses Hyper-Registers to evenly balance slacks on all the registers in a critical chain.

### Related Information

[Compiling Intel Hyperflex Architecture Designs, Intel Hyperflex Architecture High-Performance Design Handbook](#)

## 5.3.1. Viewing Critical Chains

Looking at the critical chain shows the exact logic that limits retiming operations in your design. For example, you can see if the retiming is limited by your RTL code, or by the constraints you applied on the design. Intel Quartus Prime Pro Edition reports one critical chain per clock domain and clock domain crossing.

The critical chain is available at two different stages in the Hyper Aware Design Flow:

- In the Retiming Limit Details Report—this report for the retiming stage in the Hyper Aware Design Flow, and is enabled by default.
- In the Fast Forward Compilation Report—Click **Fast Forward Timing Closure Recommendations** on the Compilation Dashboard to run..
- You can also graphically visualize the critical chains in the Technology Map Viewer.

### Related Information

- [Locate Critical Chains, Intel Hyperflex Architecture High-Performance Design Handbook](#)
- [Intel Stratix 10 HyperFlex Design: Analyzing Critical Chains \(OS10CRCHNS\) Online Course](#)

## 5.4. Design Evaluation for Timing Closure

As you move towards completion of your design, you can begin evaluating your design for timing closure. Follow the techniques in this section to evaluate whether your design is likely to close timing. If your design requires further steps for timing closure, you can use the analysis techniques in this section to determine whether RTL changes can help you to close timing.

### 5.4.1. Review Messages

After compiling your design, review the messages in each section of the compilation report. Most designs that fail timing start out with other problems that the Fitter reports as warning messages during compilation. Determine what causes a warning message, and whether to fix or ignore the warning.

After reviewing the warning messages, review the informational messages. Take note of anything unexpected, for example, unconnected ports, ignored constraints, missing files, or other unexpected conditions.

### 5.4.2. Evaluate Fitter Netlist Optimizations

You can specify options that direct the Fitter to perform optimizations to the design netlist. Specify global options, such as register packing, duplicating or deleting logic cells, or inverting signals by clicking **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**.

Figure 12. Netlist Optimizations Report

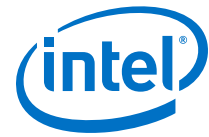
	Node	Action	Operation	Reason
1	inputdspa_reg_d1[0]	Packed Register	Register Packing	Timing optimization
2	inputdspa_r...~_Duplicate	Packed Register	Register Packing	Timing optimization
3	inputdspa_reg_d1[10]	Packed Register	Register Packing	Timing optimization
4	inputdspa_r...~_Duplicate	Packed Register	Register Packing	Timing optimization
5	inputdspa_reg_d1[11]	Packed Register	Register Packing	Timing optimization

### 5.4.3. Evaluate Optimization Results

After checking what optimizations were done and how they improved performance, evaluate the runtime it took to get the extra performance. To reduce compilation time, review the physical synthesis and netlist optimizations over a couple of compilations, and edit the RTL to reflect the changes that physical synthesis performed. If a particular set of registers consistently get retimed, edit the RTL to retime the registers the same way. If the changes are made to match what the physical synthesis algorithms did, the physical synthesis options can be turned off to save compile time while getting the same type of performance improvement.

### 5.4.4. Evaluate Resource Usage

Evaluate the device resources that the design consumes, including global and non-global signal usage, routing utilization, and clustering difficulty. Determine whether you approach capacity for any type of resource that might limit performance.



### 5.4.4.1. Evaluate Global and Non-Global Usage

For designs that contain many clocks, evaluate global and non-global signals to determine whether global resources are used effectively, and if not, consider making changes. You can find these reports in the Resource section under Fitter in the **Compilation Report** panel.

The figure shows an example of inefficient use of a global clock. The

**Figure 13. Inefficient Use of a Global Clock—Single Fan-Out from Global Clock**

Global & Other Fast Signals			
Location	Fan-Out	Global Resource Used	Global Line Name
FRACTIONALPLL_X98_Y2_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y2_N1	29044	Global Clock	GCLK7
PLLOUTPUTCOUNTER_X98_Y13_N1	253103	Global Clock	GCLK6
FF_X185_Y66_N13	280349	Global Clock	GCLK8
PIN_AE17	4887	Global Clock	GCLK4
FRACTIONALPLL_X98_Y11_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y3_N1	1	Global Clock	GCLK5
PLLOUTPUTCOUNTER_X98_Y1_N1	1691	Regional Clock	RCLK29
PLLOUTPUTCOUNTER_X98_Y8_N1	302	Regional Clock	RCLK23
PLLOUTPUTCOUNTER_X98_Y11_N1	141	Regional Clock	RCLK25
PLLOUTPUTCOUNTER_X98_Y10_N1	17	Regional Clock	RCLK22

If you assign these resources to a Regional Clock, the Global Clock becomes available for another signal. You can ignore signals with an empty value in the **Global Line Name** column as the signal uses dedicated routing, and not a clock buffer. The Non-Global High Fan-Out Signals report lists the highest fan-out nodes not routed on global signals. Reset and enable signals appear at the top of the list.

If there is routing congestion in the design, and there are high fan-out non-global nodes in the congested area, consider using global or regional signals to fan-out the nodes, or duplicate the high fan-out registers so that each of the duplicates can have fewer fan-outs. Use the Chip Planner to locate high fan-out nodes, to report routing congestion, and to determine whether the alternatives are viable.

### 5.4.4.2. Evaluate Routing Usage

Review routing usage reported in the **Fitter Resource Usage Summary** report.

**Figure 14. Fitter Resource Usage Summary Report**

Fitter Resource Usage Summary			
Q <<Filter>>			
	Resource	Usage	%
27			
28	IOPLLs	0 / 14	0 %
29	Global signals	1	
30	Impedance control blocks	0 / 24	0 %
31	Average interconnect usage (total/H/V)	0.0% / 0.0% / 0.0%	
32	Peak interconnect usage (total/H/V)	1.6% / 1.6% / 1.6%	
33	Maximum fan-out	1545	
34	Highest non-global fan-out	124	
35	Total fan-out	5650	
36	Average fan-out	3.23	

**Average interconnect usage** reports the average amount of interconnect that is used, out of what is available on the device. **Peak interconnect usage** reports the largest amount of interconnect used in the most congested areas.

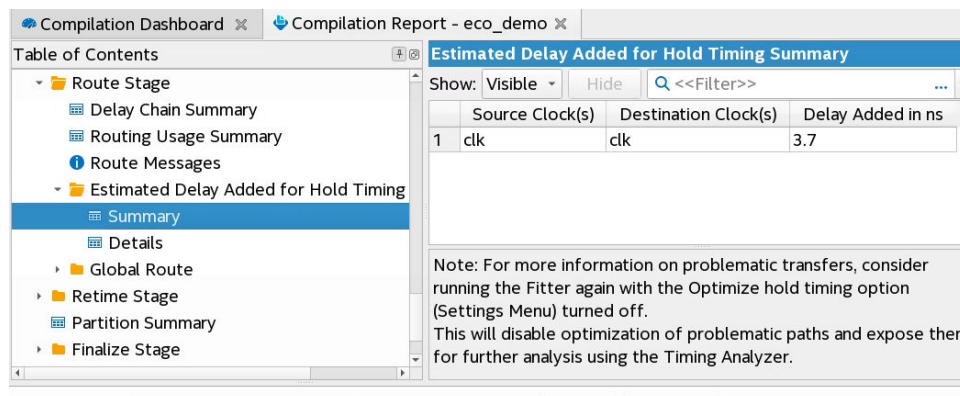
Designs with an average value below 50% typically do not have any problems with routing. Designs with an average between 50-65% may have difficulty routing. Designs with an average over 65% typically have difficulty meeting timing unless the RTL tolerates a highly utilized chip. Peak values at or above 90% are likely to have problems with timing closure; a 100% peak value indicates that all routing in an area of the device has been used, so there is a high possibility of degradation in timing performance.

### 5.4.4.3. Evaluate Wires Added for Hold

During routing the Fitter may add wire between register paths to increase delay to meet hold time requirements. The Fitter reports how much routing delay was added in the **Estimated Delay Added for Hold Timing** report. Excessive additional wire can indicate an error with the constraint. The cause of such errors is typically incorrect multicycle transfers between multi-rate clocks, and between different clock networks.

Review the specific register paths in the **Estimated Delay Added for Hold Timing** report to determine whether the Fitter adds excessive wire to meet hold timing.

Figure 15. Estimated Delay Added for Hold Timing Report

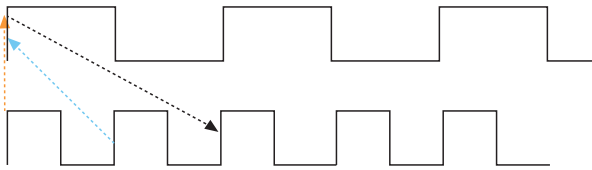


An example of an incorrect constraint which can cause the router to add wire for hold requirements is when there is data transfer from 1x to 2x clocks. Assume the design intent is to allow two cycles per transfer. Data can arrive any time in the two destination clock cycles by adding a multicycle setup constraint as shown in the example:

```
set_multicycle_path -from 1x -to 2x -setup -end 2
```

The timing requirement is relaxed by one 2x clock cycle, as shown in the black line in the waveform in the figure.

**Figure 16. Timing Requirement Relaxed Waveform**



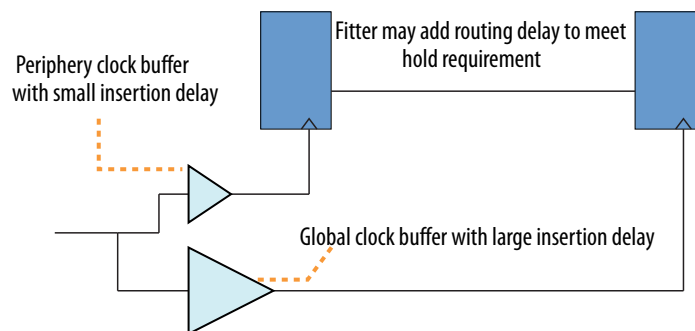
The default hold requirement, shown with the dashed blue line, can force the router to add wire to guarantee that data is delayed by one cycle. To correct the hold requirement, add a multicyle constraint with a hold option.

```
set_multicycle_path -from 1x -to 2x -setup -end 2
set_multicycle_path -from 1x -to 2x -hold -end 1
```

The orange dashed line in the figure above represents the hold relationship, and no extra wire is required to delay the data.

The router can also add wire for hold timing requirements when data transfers in the same clock domain, but between clock branches that use different buffering. Transferring between clock network types happens more often between the periphery and the core. The following figure shows data is coming into a device, a periphery clock drives the source register, and a global clock drives the destination register. A global clock buffer has larger insertion delay than a periphery clock buffer. The clock delay to the destination register is much larger than to the source register, hence extra delay is necessary on the data path to ensure that it meets its hold requirement.

**Figure 17. Clock Delay**



To identify cases where a path has different clock network types, review the path in the Timing Analyzer, and check nodes along the source and destination clock paths. Also, check the source and destination clock frequencies to see whether they are the same, or multiples, and whether there are multicyle exceptions on the paths. Finally, ensure that all cross-domain paths that are false by intent have an associated false path exception.

If you suspect that routing is added to fix real hold problems, you can disable the **Optimize hold timing** advanced Fitter setting (**Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Optimize hold timing**). Recompile the design with **Optimize hold timing** disabled, and then rerun timing analysis to identify and correct any paths that fail hold time requirements.

*Note:* Disable the **Optimize hold timing** option only when debugging your design. Ensure to enable the option (default state) during normal compiles. Wire added for hold is a normal part of timing optimization during routing and is not always a problem.

## 5.4.5. Evaluate Other Reports and Adjust Settings Accordingly

### 5.4.5.1. Difficulty Packing Design

In the Fitter Resource Section, under the **Resource Usage Summary**, review the **Difficulty Packing Design** report. The **Difficulty Packing Design** report details the effort level (low, medium, or high) of the Fitter to fit the design into the device, partition, and Logic Lock region.

As the effort level of **Difficulty Packing Design** increases, timing closure gets harder. Going from medium to high can result in significant drop in performance or increase in compile time. Consider reducing logic to reduce packing difficulty.

### 5.4.5.2. Review Ignored Assignments

The Compilation Report includes details of any assignments that the Fitter ignores. The Fitter may ignore assignments if they refer to nodes names that change, but assignments are not updated accordingly. Make sure that the Fitter is not ignoring any valid assignments.

### 5.4.5.3. Review Non-Default Settings

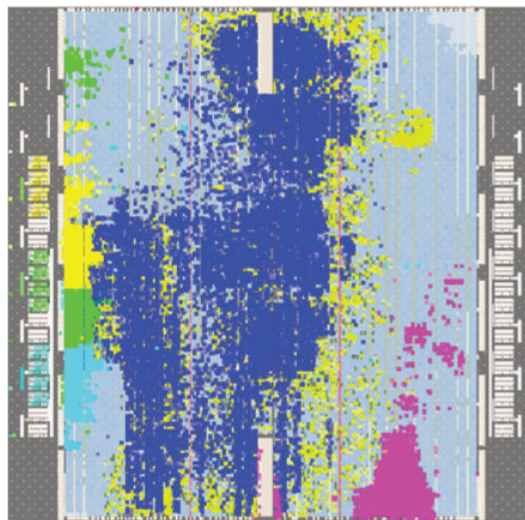
The Synthesis and Fitter reports list all settings set to a non-default value during the compilation. Review the non-default settings to ensure benefit.

### 5.4.5.4. Review the Design Floorplan

Use the Chip Planner for reviewing placement. You can use the Chip Planner to locate hierarchical entities, using colors for each located entity in the floorplan. Look for logic that seems out of place, based on where you expect it to be

For example, logic that interfaces with I/Os should be close to the I/Os, and logic that interfaces with an IP or memory should be close to the IP or memory.

**Figure 18. Floorplan with Color-Coded Entities**







The following notes describe use of the visualization in *Floorplan with Color-Coded Entities*:

- The green block is spread apart. Check to see if those paths are failing timing, and if so, what connects to that module that could affect placement.
- The blue and aqua blocks are spread out and mixed together. Check if connections between the two modules contribute to this.
- The pink logic at the bottom must interface with I/Os at the bottom edge. Check fan-in and fan-out of a highlighted module by using the buttons on the task bar. Look for signals that go a long way across the chip and see if they are contributing to timing failures.
- Check global signal usage for signals that affect logic placement, and verify if the Fitter placed logic feeding a global buffer close to the buffer and away from related logic. Use settings like high fan-out on non-global resource to pull logic together.
- Check for routing congestion. The Fitter spreads out logic in highly congested areas, making the design harder to route.

#### 5.4.5.5. Adjust Placement Effort

You can increase the **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Placement Effort Multiplier** value to spend additional compilation time and effort in Place stage of the Fitter.

Adjust the multiplier after reviewing and optimizing other settings and RTL. Try an increased value, up to 4, and reset to default if performance or compile time does not improve.

#### 5.4.5.6. Adjust Fitter Effort

Fitter **Optimization mode** settings allow you to specify whether the Compiler focuses optimization efforts for performance, resource utilization, power, or compile times.

By default, the Fitter **Optimization mode** is set to **Balanced (Normal flow)** mode, which reduces Fitter effort and compilation time as soon as timing requirements are met. You can optionally select another **Optimization mode** to target performance, area, routability, power, or compile time.

To increase Fitter effort further, you can also enable the **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter) > Fitter Effort** option. The default **Auto Fit** setting reduces Fitter effort once timing requirements are met. **Standard Fit (highest effort)** setting uses maximum effort regardless of the design's requirements, leading to higher compilation time and more timing margin.

#### 5.4.5.7. Review Timing Constraints

Ensure that you constrain all clocks with the correct frequency requirements. For example, check the **Report Ignored Constraints** report to locate any incorrect names in the design, most commonly caused by changes in the design hierarchy. Review the **Report Unconstrained Paths** report to locate unconstrained paths. Add constraints as necessary so that the Compiler can fully optimize the design.

### 5.4.6. Evaluate Clustering Difficulty

You can evaluate clustering difficulty to help reach timing closure. You can monitor clustering difficulty whenever you add logic and recompile. Use the clustering information to gauge how much timing closure difficulty is inherent in your design.

- If your design is full but clustering difficulty is low or medium, your design itself, rather than clustering, is likely the main cause of congestion.
- Conversely, congestion occurring after adding a small amount of logic to the design, can be due to clustering. If clustering difficulty is high, this contributes to congestion regardless of design size.

### 5.4.7. Revise and Recompile

Look for obvious problems that you can fix with minimal effort. To identify where the Compiler had trouble meeting timing, perform seed sweeping with about five compiles. Doing so shows consistently failing paths. Consider recoding or redesigning that part of the design.

To reach timing closure, a well written RTL can be more effective than changing your compilation settings. Seed sweeping can also be useful if the timing failure is very small, and the design has already been optimized for performance improvements and is close to final release. Additionally, seed sweeping can be used for evaluating changes to compilation settings. Compilation results vary due to the random nature of fitter algorithms. If a compilation setting change produces lower average performance, undo the change.

Sometimes, settings or constraints can cause more problems than they fix. When significant changes to the RTL or design architecture have been made, compile periodically with default settings and without Logic Lock regions, and re-evaluate paths that fail timing.

## 5.5. Timing Optimization

You can use the techniques and tools in this section to optimize timing when your design does not meet its timing requirements.

### 5.5.1. Correct Design Assistant Rule Violations

After running any stage of the Compiler, review the Design Assistant reports to analyze any design rule violations, and view specific recommendations to correct failing paths in your design. When enabled, the Intel Quartus Prime Design Assistant automatically runs during compilation and reports any violations against a standard set of Intel FPGA-recommended design guidelines. Design Assistant rule categories include Timing Closure, Clocking, CDC, reset, and floorplanning rules.

You can fully customize the Design Assistant for your individual design characteristics and reporting requirements. You can run Design Assistant in Compilation Flow mode, allowing you to view the violations relevant for that stage at the stage completion. Alternatively, Design Assistant is available in analysis mode in tools such as Timing Analyzer and Chip Planner. Design Assistant can cross-probe from an individual rule violation to the source.

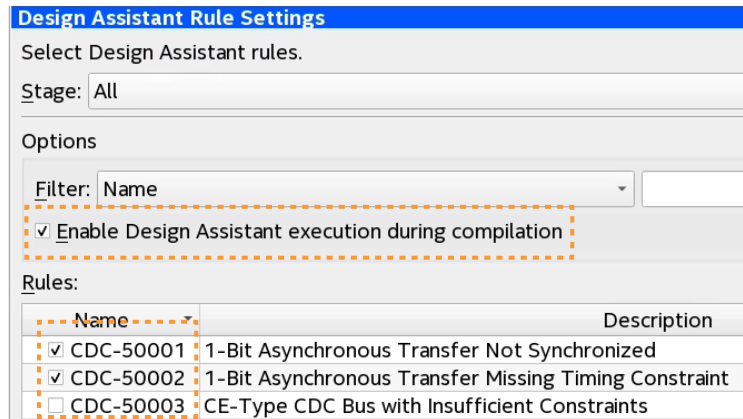
Follow these steps to enable and run Design Assistant and view results following compilation:





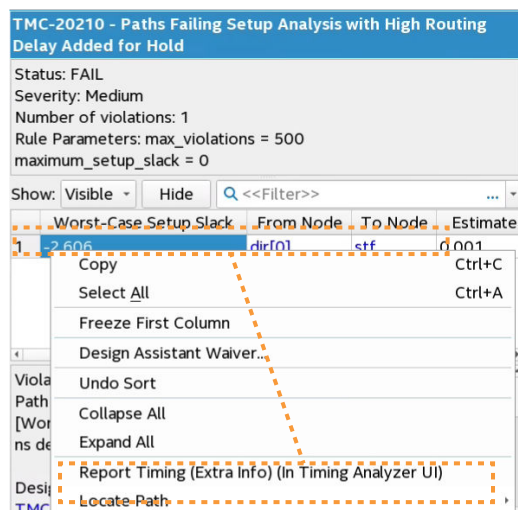
1. Click **Assignments** > **Settings** > **Design Assistant Rules Settings**.

**Figure 19. Design Assistant Rules Settings**



2. To enable Design Assistant checking during compilation, turn on **Enable Design Assistant execution during compilation**.
3. To run Design Assistant during compilation, run one or more modules of the Compiler. Design Assistant reports results for each stage in the Compilation Report.
4. To view the results for each rule, click the rule in the **Rules** list. A description of the rule and design recommendations for correction appear.
5. For timing path-related rule violations, right-click the node or path, and then click **Report Timing (Extra Info)** or **Report Path (Extra Info)**. The Timing Analyzer loads and automatically displays the **Report Timing** or **Report Path** data related to the rule violation, allowing you to probe every aspect of the violation. **Report Path** can report timing even for paths that are cut.

**Figure 20. Cross Probing From Design Assistant Rule Violations to Timing Analyzer**



### Related Information

Design Rule Checking with Design Assistant, Intel Quartus Prime Pro Edition User Guide: Design Recommendations

## 5.5.2. Implement Fast Forward Timing Closure Recommendations

In traditional FPGA timing closure flows, the starting point for most design analysis is the critical path. Due to the nature of Intel Hyperflex architecture and the availability of the Hyper Retimer, it is best to start your timing closure activities from the Retiming Limit Report. Provide the Hyper-Retimer as many optimization opportunities as possible, before having to look into more time intensive and potentially manual timing closure techniques.

### Related Information

Interpreting Critical Chain Reports, Intel Hyperflex Architecture High-Performance Design Handbook

### 5.5.2.1. Retiming Limit Details Report

Use the Retiming Limit Details report to get specific information on what is currently limiting the Hyper Retimer from performing more optimizations.

The Retiming Limit Details report specifies:

- Clock Transfer: Clock domain, or the clock domain transfer for which the critical chain applies
- Limiting Reason: Design conditions which prevent further optimizations from happening.
- Critical Chain Details: Timing paths associated with the timing restrictions.

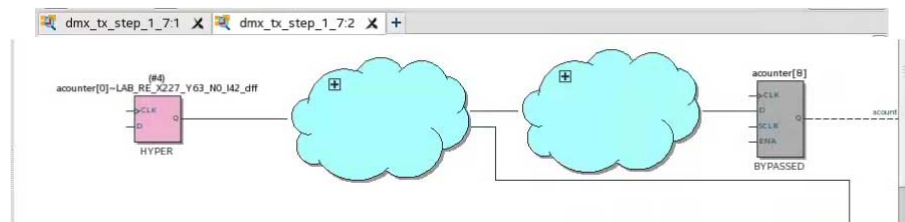
#### 5.5.2.1.1. Using the Retiming Limit Details Report

To access the Retiming Limit Details report:

1. In the **Reports** tab, double-click **Retiming Limit Details** under **Fitter > Retime Stage**.
2. To locate the critical chain in the Technology Map Viewer, right-click any path and click **Locate Critical Chain in Technology Map Viewer**.

The Technology Map Viewer displays a schematic representation of the complete critical chain after place, route and register retiming.

**Figure 21. Critical Chain in Technology Map Viewer**





### 5.5.2.2. Fast Forward Timing Closure Recommendations

When running Fast Forward compilation, the Compiler removes signals from registers to allow mobility within the netlist for subsequent retiming. Fast Forward compilation generates design-specific timing closure recommendations, and predicts maximum performance with removal of all timing restrictions.

After you complete Fast Forward explorations, you can determine which recommendations to implement to provide the most benefit. Implement appropriate recommendations in your RTL, and recompile the design to achieve the performance levels that Fast Forward reports.

The Fast Forward Details Report provides the following information:

**Table 14. Fast Forward Details Report Information**

Name	Description
Step	Displays the various Fast Forward optimization steps, starting from the pre-optimization base compilation. <ul style="list-style-type: none"> <li>Each step comes with its associated critical chain.</li> <li>Each step corresponds to a new optimization cumulative to the previous step.</li> </ul>
Fast Forward Optimization	Analyzed Summary of the optimizations necessary to implement each step.
Estimated $f_{MAX}$	Estimated $f_{MAX}$ performance after you implement the recommendations for this step in your design. This is cumulative, and step $n$ represents the potential $f_{MAX}$ after implementing all previous steps.
Optimization Analyzed	(cumulative) List of all the consecutive optimization steps applied.
Recommendation for Critical Chain	Lists recommended changes to your designs. These recommendations are geared towards removing retiming limitations, and allowing register movement.

#### 5.5.2.2.1. Generating Fast Forward Timing Closure Recommendations

To generate Fast Forward timing closure recommendations:

1. On the Compilation Dashboard, click **Fast Forward Timing Closure Recommendations**.  
The Compiler runs prerequisite synthesis or Fitter stages as needed, and generates timing closure recommendations in the Compilation Report.
2. View timing closure recommendations in the Compilation Report to evaluate design performance, and implement key RTL performance improvements.

The Intel Quartus Prime Pro Edition software allows you to automate or refine Fast Forward analysis:

- To run Fast Forward compilation during each full compilation, click **Assignments > Settings > Compiler Settings > HyperFlex**, and turn on **Run Fast Forward Timing Closure Recommendations during compilation**.
- To modify how Fast Forward compilation interprets specific I/O and block types, click **Assignments > Settings > Compiler Settings > HyperFlex Advanced Settings**.

#### 5.5.2.2.2. Implementing Fast Forward Recommendations

After implementing timing closure recommendations in your design, you can rerun the Retime stage to obtain the predictive performance gains.

You can continue exploring performance and implementing RTL changes to your code until you reach the desired performance target. Once you have completed all the modifications you want to do, continue your timing closure activities with the traditional techniques explained in this document.

For more information about implementing Fast Forward timing closure recommendations in your design, refer to the *Implement Fast Forward Recommendations* section of the *Intel Hyperflex Architecture High Performance Design Handbook*

### 5.5.3. View Timing Optimization Advisor

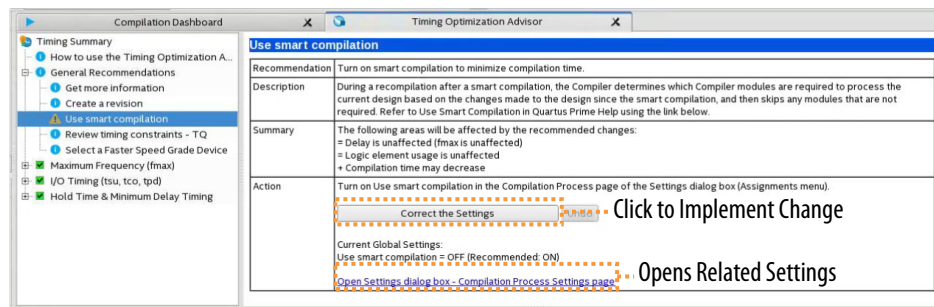
While the Timing Analyzer **Report Timing Closure Recommendations** task gives specific recommendations to fix failing paths, the Timing Optimization Advisor gives more general recommendations to improve timing performance for a design.

The Timing Optimization Advisor guides you in making settings that optimize your design to meet your timing requirements. To run the Timing Optimization Advisor click **Tools > Advisors > Timing Optimization Advisor**. This advisor describes many of the suggestions made in this section.

When you open the Timing Optimization Advisor after compilation, you can find recommendations to improve the timing performance of your design. If suggestions in these advisors contradict each other, evaluate these options and choose the settings that best suit the given requirements.

The example shows the Timing Optimization Advisor after compiling a design that meets its frequency requirements, but requires setting changes to improve the timing.

Figure 22. Timing Optimization Advisor



When you expand one of the categories in the Timing Optimization Advisor, such as **Maximum Frequency (fmax)** or **I/O Timing (tsu, tco, tpd)**, the recommendations appear in stages. These stages show the order in which to apply the recommended settings.

The first stage contains the options that are easiest to change, make the least drastic changes to your design optimization, and have the least effect on compilation time.

Icons indicate whether each recommended setting has been made in the current project. In the figure, the checkmark icons in the list of recommendations for Stage 1 indicates recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icons indicate general suggestions. For these entries, the advisor does not report whether



these recommendations were followed, but instead explains how you can achieve better performance. For a legend that provides more information for each icon, refer to the “How to use” page in the Timing Optimization Advisor.

Each recommendation provides a link to the appropriate location in the Intel Quartus Prime GUI where you can change the settings. For example, consider the **Synthesis Netlist Optimizations** page of the **Settings** dialog box or the **Global Signals category** in the Assignment Editor. This approach provides the most control over which settings are made and helps you learn about the settings in the software. When available, you can also use the **Correct the Settings** button to automatically make the suggested change to global settings.

For some entries in the Timing Optimization Advisor, a button allows you to further analyze your design and see more information. The advisor provides a table with the clocks in the design, indicating whether they have been assigned a timing constraint.

## 5.5.4. Review Timing Path Details

Reporting the timing paths and routing details can help uncover correctable timing and routing delays and other conditions that prevent retiming registers for higher performance.

### 5.5.4.1. Report Timing

The Timing Analyzer's **Reports > Custom Reports > Report Timing** command reports the timing of any path or clock domain in the design. The equivalent scripting command is `report_timing`.

**Figure 23. Report Timing Report**

Report Timing						
Command Info		Summary of Paths				
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship
1	39.103	addr...g[2]	my_...[2]	clk	clk	40.000
2	39.144	addr...g[2]	my_...[2]	clk	clk	40.000
3	39.153	addr...g[2]	my_...[2]	clk	clk	40.000
Path #1: Setup slack is 39.103						
Path Summary		Statistics	Data Path	Waveform		
		Property		Value		
1	From Node			addressr_reg[2]		
2	To Node			my_mlab_inst0 radd_reg_b[2]		
3	Launch Clock			clk		
4	Latch Clock			clk		
5	Data Arrival Time			3.055		
6	Data Required Time			42.158		
7	Slack			39.103		
8	Worst-Case Operating Conditions			Slow 900mV 100C Model		

You can specify various options to customize the reporting. You can specify the **Clocks** and **Targets** that the report displays, the **Analysis Type** to run, whether to display **Extra Info** in the report, and the **Output** options for the report. For example, you can increase the number of paths to report, add a **Target** filter, and add a **From Clock**.

Figure 24. Report Timing Dialog Box (Top Section)

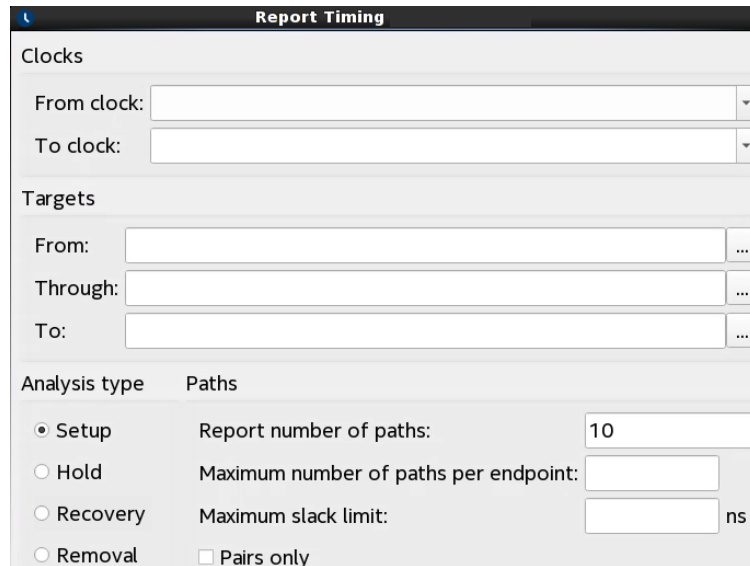


Figure 25. Report Timing Dialog Box (Bottom Section)

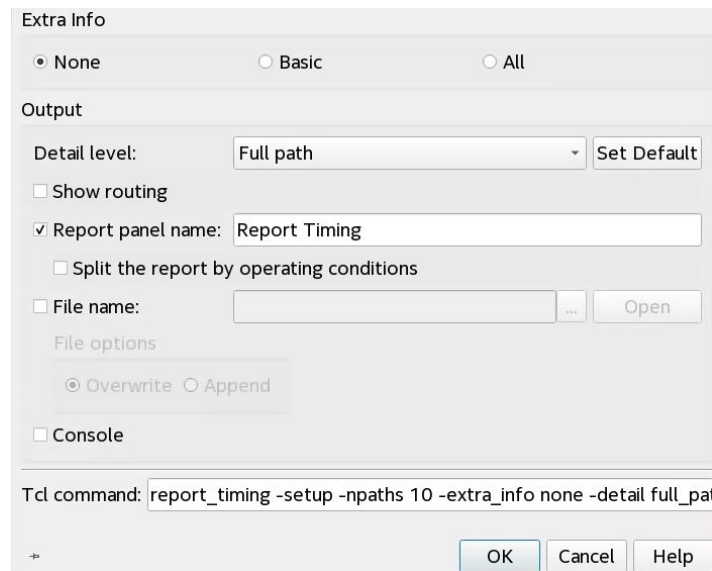


Table 15. Report Timing Settings

Option	Description
<b>Clocks</b>	<b>From Clock</b> and <b>To Clock</b> filter paths in the report to show only the launching or latching clocks you specify.
<b>Targets</b>	Specifies the target node for <b>From Clock</b> and <b>To Clock</b> to report paths with only those endpoints. Specify an I/O or register name or I/O port for this option. The field also supports wildcard characters. For example, to report only paths within a specific hierarchy: <pre>report_timing -from * egress:egress_inst * \ -to * egress:egress_inst * -(other options)</pre>

*continued...*



Option	Description
	When the <b>From</b> , <b>To</b> , or <b>Through</b> boxes are empty, the Timing Analyzer assumes all possible targets in the device. The <b>Through</b> option limits the report to paths that pass through combinatorial logic, or a particular pin on a cell.
<b>Analysis type</b>	The <b>Analysis type</b> options are <b>Setup</b> , <b>Hold</b> , <b>Recovery</b> , or <b>Removal</b> . The Timing Analyzer reports the results for the type of analysis you select.
<b>Paths</b>	Specifies the number of paths to display by endpoint and slack level. The default value for <b>Report number of paths</b> is 10, otherwise, the report can be very long. Enable <b>Pairs only</b> to list only one path for each pair of source and destination. Limit further with <b>Maximum number of paths per endpoints</b> . You can also filter paths by entering a value in the <b>Maximum slack limit</b> field.
<b>Extra Info</b>	Provides additional data that is relevant for diagnosing timing failure root cause, such as intrinsic margin and unexpected routing detours caused by congestion and hold time fix-up. Specify whether to include <b>None</b> , <b>Basic</b> , or <b>All</b> extra information in the report. The <b>Extra Info</b> tab data can help you identify potential, unnecessary routing detours, as well as placement or circuit issues that restrict the path $f_{MAX}$ performance. Refer to <a href="#">Intrinsic Margin and the Extra Info Tab</a> on page 71. <ul style="list-style-type: none"> <li>• <b>All</b>—report includes <b>Extra Info</b> tab that reports extra information for source timing endpoints that pass through the unregistered output of a RAM or DSP block, or for destination timing endpoints that pass through the unregistered input of a DSP block. The <b>Data Path</b> tab includes Estimated Delay Added for Hold and Route Stage Congestion Impact data.</li> <li>• <b>Basic</b>—report includes the <b>Extra Info</b> tab but no extra information on the <b>Data Path</b> tab.</li> <li>• <b>None</b>—report includes no <b>Extra Info</b> tab or other extra information on the <b>Data Path</b> tab.</li> </ul>
<b>Output</b>	Specify the path types the analysis includes in output for <b>Detail level</b> : <ul style="list-style-type: none"> <li>• <b>Summary</b>—level includes basic summary reports. Review the <b>Clock Skew</b> column in the <b>Summary</b> report. If the skew is less than +/-150ps, the clock tree is well balanced between source and destination.</li> <li>• <b>Path only</b>—displays all the detailed information, except the <b>Data Path</b> tab displays the clock tree as one line item.</li> <li>• <b>Path and Clock</b>—displays the same as <b>Path only</b> with respect to the clock.</li> <li>• <b>Full path</b>—when higher clock skew is present, enable the <b>Full path</b> option. This option breaks the clock tree into greater detail, showing every cell, including the input buffer, PLL, global buffer (called <code>CLKCTRL_</code>), and any logic. Review this data to determine the cause of clock skew in your design. Use the <b>Full path</b> option for I/O analysis because only the source clock or destination clock is inside the FPGA, and therefore the delay is a critical factor to meet timing.</li> </ul>
<b>Show routing</b>	Shows routing data in the report.
<b>Split the report by operating conditions</b>	For the operating condition timing corners, subdivides the data by each operating condition.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.

### Intrinsic Margin and the Extra Info Tab

The **Extra Info** tab contains other timing metrics to help you diagnose timing closure issues, including **Intrinsic Margin** for the path. The **Extra Info** tab can help you identify potential significant or unexpected routing detours caused by congestion and hold time fix-up. The **Extra Info** tab can also report extra information for source timing endpoints that pass through the unregistered output of a RAM or DSP block, or for destination timing endpoints that pass through the unregistered input of a DSP block. You can review the **Extra Info** data and **Locate Path** or **Locate Chip Area** in Chip Planner, Technology Map Viewer, or Resource Property Viewer to determine whether to make changes to improve placement and routing.

Figure 26. Extra Info Tab

Path Summary	Statistics	Extra Info	Data Path	Waveform
Property			Value	
1	Intrinsic Margin	-0.690		
2	Max Fanout	1		
3	Source/Destination Bounding Box	(225, 63) - (225, 63)		
4	Cell Bounding Box	(225, 63) - (225, 63)		
5	Interconnect Bounding Box	(225, 63) - (225, 63)		
6	Source/Destination Relative Area	1.000		
7	Cell Relative Area	1.000		
8	Interconnect Relative Area	1.000		
9	Retiming Restriction From Node	Node is in a block that cannot be retimed.		
10	Retiming Restriction To Node	--		
11	From Node is Po...Up "Don't Care"	Yes		
12	.To Node is Power-Up "Don't Care"	Yes		
13	From Node is not fully-registered	RAM with unregistered outputs		
14	Max Borrow Time From Node	0.000		
15	Max Borrow Time To Node	0.000		
16	Route Stage Congestion Impact	--		
17	Estimated Delay Added for Hold	0.000		

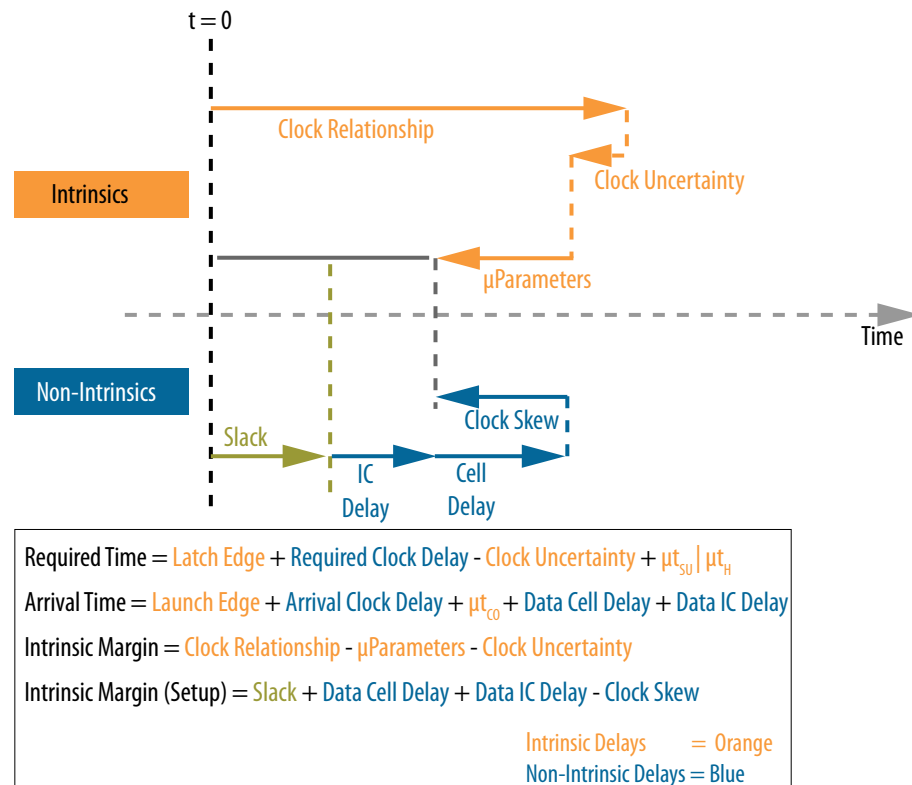
Intrinsic margin is a numeric value that the Timing Analyzer calculates from the timing requirements and path element delays. The Timing Analyzer also derives the slack of the path from the same requirements and delays, but with a different calculation. The slack of a path specifies the margin by which the path meets its timing requirement. A path can fail timing requirements for many different reasons. For example, the clock relationship can be impossibly tight, or there can be excessive routing delays that alone cause failure for the timing path. Calculating the intrinsic margin of a timing path, and then comparing that margin to other delays of the path, can help identify the specific reasons why a path fails its timing requirement.





Some delay elements are more sensitive to a path's placement and routing than others. Intrinsic delays are less sensitive to placement and routing, and are inherent in the RTL and timing requirements. Non-intrinsic delays are the other delays that are sensitive to placement and routing. The Timing Analyzer derives intrinsic margin from the following calculations:

**Figure 27. Intrinsic Margin Calculations**

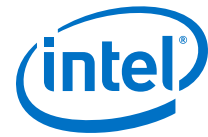


- **When the intrinsic margin is less than 0**—the path has such a tight timing relationship, a large difference in  $\mu$ parameters, or such significant clock source uncertainty that the path fails before any additional delay is added. In this case, review the SDC constraints to verify that the timing relationship is correct. An incorrect relationship can exist between unrelated clocks that lack the proper timing cut. Ensure that parameterizable hard blocks are fully registered. Such blocks include M20K RAM and DSP blocks. Also investigate clock sources to verify that the clocks involved are promoted to use global signals for their routing.
- **When the clock skew exceeds the intrinsic margin**—you must address the clock transfer to meet timing on the path. You may need to create clock region assignments. You might also need to redesign cross-clock transfers to switch from synchronous to asynchronous implementation, such as by using a FIFO or other handshake.

- When the cell delay is greater than its intrinsic margin**—the path would fail timing even if the clocks are perfect and uses no routing wires. In this case, you should take actions to reduce the cell delay. You can rewrite RTL to reduce the logic depth, restructure logic to allow the Compiler to use faster LUT inputs, or unblock retiming optimizations. Many stages of the compilation flow can automatically retime registers to reduce logic depth, but only in ways that maintain functionality and that the device architecture supports. Common steps to unblock the Hyper-Retimer include removing asynchronous resets and removing initial conditions. The **Extra Info** tab reports reasons that block the registers from retiming. Refer to "Retiming Restrictions and Workarounds" in the *Intel Hyperflex Architecture High-Performance Design Handbook*.
- When the interconnect delay is greater than its intrinsic margin**—the path would fail timing even if the clocks are perfect, and there is no logic. This situation can occur if registers are too far apart, or a timing path has to detour around a congested area of the chip. Review the fan-in and fan-out of registers that are far apart to determine why they are far apart. Applying Logic Lock regions can cause the Fitter to place the registers closer together within the same region. However, to avoid introducing other problems, use Logic Lock regions to improve placement only after determining why the placement was initially poor. The Fitter can place registers far apart if you do not appropriately size or position the clock regions.

**Table 16. Extra Info Tab Data**

Extra Info Data	Description
<b>Intrinsic Margin</b>	Reports the intrinsic and non-intrinsic timing elements that comprise the timing path slack value. Intrinsic margin is a numeric value that the Timing Analyzer calculates from the timing requirements and path element delays. The Timing Analyzer also derives the slack of the path from the same requirements and delays, but with a different calculation. Intrinsic delays are less sensitive to placement and routing, and are inherent in the RTL and timing requirements. Non-intrinsic delays are the other delays that are sensitive to placement and routing. Refer to <a href="#">Figure 27</a> on page 73.
<b>Max Fanout</b>	Reports the maximum fan-out of register and combinational nodes in the path.
<b>Source/Destination Bounding Box</b>	Reports the lower-left and upper-right coordinates for the boundary box enclosing the source and destination registers. In an ideal case, the <b>Source/Destination Bounding Box</b> , <b>Cell Bounding Box</b> , and <b>Interconnect Bounding Box</b> values are roughly the same, and the relative areas are approximately 1.0. If the cell bounding box size grows relative to the <b>Source/Destination Bounding Box</b> , that can indicate a potential unnecessary routing detour on the path.
<b>Cell Bounding Box</b>	Reports the lower-left and upper-right coordinates for the boundary box enclosing the source and destination registers, and any cells in the path.
<b>Interconnect Bounding Box</b>	Reports the lower-left and upper-right coordinates for the boundary box enclosing the interconnect. The <b>Interconnect Bounding Box</b> would likely expand beyond the <b>Cell Bounding Box</b> due to hold fix-up.
<b>Source/Destination Relative Area</b>	Reports the area for the source and destination, relative to the <b>Source/Destination Bounding Box</b> . The value is always 1.0, which equals the same size.
<b>Cell Relative Area</b>	Reports the area for the cell, relative to the <b>Source/Destination Bounding Box</b> . A value of 1.0 equals the same size. A value greater than 1.0 can indicate a path has a cell outside of the space between the registers in the path.
<b>Interconnect Relative Area</b>	Reports the area for the interconnect, relative to the <b>Source/Destination Bounding Box</b> . A value of 1.0 equals the same size. A value greater than 1.0 may indicate a path has a cell outside of the space between the registers in the path.
<b>Retiming Restriction From Node</b>	Reports any restrictions (for example, "preserve" assignments) that exist on the From Node that limit retiming ability. Consider removing the retiming restriction to allow retiming and improve performance for timing closure.
<i>continued...</i>	



Extra Info Data	Description
<b>Retiming Restriction To Node</b>	Reports any restrictions (for example, "preserve" assignments) that exist on the <b>To Node</b> that limit retiming ability. Consider removing the retiming restriction to allow retiming and improve performance for timing closure.
<b>From Node is Power-Up "Don't Care"</b>	Reports whether Power-Up "Don't Care" settings exist on the <b>From Node</b> that limit retiming ability. If <b>From Node is Power-Up "Don't Care"</b> value is <b>Yes</b> , this allows retiming. If <b>From Node is Power-Up "Don't Care"</b> value is <b>No</b> ; retiming might be restricted. Consider removing the retiming restriction to allow retiming and improve performance for timing closure.
<b>To Node is Power-Up "Don't Care"</b>	Reports whether Power-Up "Don't Care" settings exists on the <b>To Node</b> that limit retiming ability. If <b>From Node is Power-Up "Don't Care"</b> value is <b>Yes</b> , this allows retiming. If <b>From Node is Power-Up "Don't Care"</b> value is <b>No</b> ; retiming might be restricted. Consider removing the retiming restriction to allow retiming and improve performance for timing closure.
<b>From Node is not fully-registered</b>	Reports RAM or DSP blocks with unregistered outputs. Registering RAM and DSP outputs can significantly improve RAM and DSP timing.
<b>To Node is not fully-registered</b>	Reports RAM or DSP blocks with unregistered inputs. Registering RAM and DSP inputs can significantly improve RAM and DSP timing.
<b>Max Borrow Time From Node</b>	Reports the maximum amount of time borrowing that occurs across all paths that involve the <b>From Node</b> .
<b>Max Borrow Time To Node</b>	Reports the amount of time borrowing that occurs across all paths that involve the <b>To Node</b> .
<b>Route Stage Congestion Impact</b>	Reports whether routing has a <b>Low</b> , <b>Medium</b> , or <b>High</b> impact on congestion. A <b>Low</b> value suggests timing issues are not congestion related. A <b>High</b> value suggests competition for scarce routing resources plays a role in poor timing.
<b>Estimated Delay Added for Hold</b>	Reports the estimated amount of delay added by Hold fix-up. This value can help you determine whether delays are routing congestion or Hold related.

### Related Information

[Retiming Restrictions and Workarounds, Intel Hyperflex Architecture High-Performance Design Handbook](#)

#### 5.5.4.2. Report Logic Depth

The Timing Analyzer's **Reports > Design Metrics > Report Logic Depth** command reports the number of logic levels within a clock domain. This value typically corresponds to the number of look-up tables (LUTs) that a path passes through. The equivalent scripting command is `report_design_metrics -logic_depth`. **Report Logic Depth** shows the distribution of logic depth among the critical paths, allowing you to identify areas where you can reduce logic levels in your RTL.

**Figure 28. Report Logic Depth (Histogram)**

	Clock Name	Relationship	Depth 1	Depth 2	Depth 3
1	clock	10.000	82	45	33

You can specify various options to customize the reporting.

Figure 29. Report Logic Depth Dialog Box

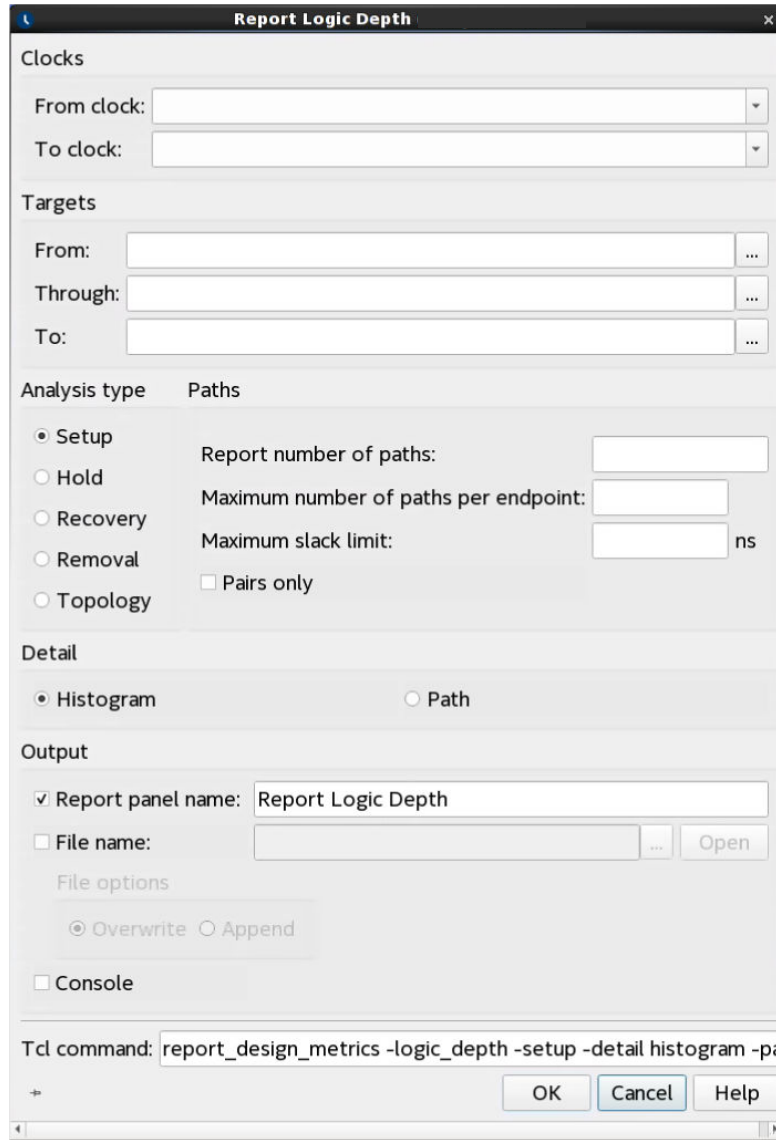
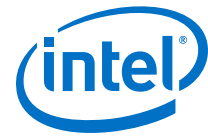


Table 17. Report Logic Depth Settings

Option	Description
<b>Clocks</b>	<b>From Clock</b> and <b>To Clock</b> filter paths in the report to show only the launching or latching clocks you specify.
<b>Targets</b>	Specifies the target node for <b>From Clock</b> and <b>To Clock</b> to report logic depth with only those endpoints. Specify an I/O or register name or I/O port for this option. The field also supports wildcard characters. When the <b>From</b> , <b>To</b> , or <b>Through</b> boxes are empty, the Timing Analyzer assumes all possible targets in the device. The <b>Through</b> option limits the report for paths that pass through combinatorial logic, or a particular pin on a cell.
<b>Analysis type</b>	The <b>Setup</b> , <b>Hold</b> , <b>Recovery</b> , and <b>Removal</b> analyses report the logic depths of the top <i>X</i> paths by slack. <b>Topology</b> analysis reports the logic depths of the top <i>X</i> paths by logic depth.

*continued...*



Option	Description
<b>Paths</b>	Specifies the number of paths to display by endpoint and slack level. The default value for <b>Report number of paths</b> is 10, otherwise, the report can be very long. Enable <b>Pairs only</b> to list only one path for each pair of source and destination. Limit further with <b>Maximum number of paths per endpoints</b> . You can also filter paths by entering a value in the <b>Maximum slack limit</b> field.
<b>Detail</b>	Specify whether to display on <b>Histogram</b> or full <b>Path</b> level of detail.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.

### 5.5.4.3. Report Neighbor Paths

The Timing Analyzer's **Reports > Design Metrics > Report Neighbor Paths** command helps you to determine the root cause of critical paths (for example, high logic level, retiming limitation, sub-optimal placement, I/O column crossing, hold fix-up, time borrowing, or others). The equivalent scripting command is `report_design_metrics -neighbor_paths`.

**Figure 30. Report Neighbor Paths Report**

Path Summary		Path Before	Path	Path After
1	From Node	wraddress_r[0]	ram_in...0~reg0	q_r[0]~...plicate
2	To Node	ram_in...0~reg0	q_r[0]~...plicate	q_r2[0]
3	Launch Clock	clock	clock	clock
4	Latch Clock	clock	clock	clock
5	Relationship	0.500	0.500	0.500
6	Setup Slack	-0.408	-0.873	-0.473
7	Hold Slack	--	0.492	--
8	Setup Operating Conditions	Slow 9... Model	Slow 9... Model	Slow 9... Model
9	Hold Operating Conditions	--	Fast 9... Model	--
10	Borrow Time From Node	0.000	0.000	0.000
11	Borrow Time To Node	0.000	0.000	0.000
12	Max Borrow Time From Node	0.000	0.000	0.000
13	Max Borrow Time To Node	0.000	0.000	0.000
14	From Node Element Type	ALM Register	EC	Hyper-Register
15	To Node Element Type	EC	Hyper-Register	ALM Register
16	Number of Paths	1	1	1
17	Clock Skew	0.052	-0.177	-0.073
18	Data Delay	0.923	1.461	0.969
19	Clock Uncertainty	-0.030	-0.030	-0.030
20	uTsu	-0.007	0.295	0.099
21	uTh	--	--	--
22	Logic Levels	2	6	0

**Report Neighbor Paths** reports the most timing-critical paths in the design, including associated slack, additional path summary information, and path bounding boxes. **Report Neighbor Paths** shows the most timing-critical **Path Before** and **Path After** each critical **Path**. Retiming or logic balancing of the **Path** can simplify timing closure if there is negative slack on the **Path**, but positive slack on the **Path Before** or **Path After**.

**Table 18. Report Neighbor Path Dialog Box Settings**

Option	Description
<b>Clocks</b>	<b>From Clock</b> and <b>To Clock</b> filter paths in the report to show only the launching or latching clocks you specify.
<b>Targets</b>	Specifies the target node for <b>From Clock</b> and <b>To Clock</b> to report neighbor paths with only those endpoints. Specify an I/O or register name or I/O port for this option. The field also supports wildcard characters. When the <b>From</b> , <b>To</b> , or <b>Through</b> boxes are empty, the Timing Analyzer assumes all possible targets in the device. The <b>Through</b> option limits the report for paths that pass through combinatorial logic, or a particular pin on a cell.
<b>Analysis type</b>	The <b>Analysis type</b> options are <b>Setup</b> , <b>Hold</b> , <b>Recovery</b> , or <b>Removal</b> . The Timing Analyzer reports the results for the type of analysis you select.
<b>Paths</b>	Specifies the number of paths to display by endpoint and slack level. The default value for <b>Report number of paths</b> is 10, otherwise, the report can be very long. Enable <b>Pairs only</b> to list only one path for each pair of source and destination. Limit further with <b>Maximum number of paths per endpoints</b> . You can also filter paths by entering a value in the <b>Maximum slack limit</b> field.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.

#### 5.5.4.4. Report Register Spread

The Timing Analyzer's **Reports > Design Metrics > Report Register Spread** command analyzes the final placement to identify registers with sinks pulling them in various directions. These registers are potential candidates for duplication. The equivalent scripting command is `report_register_spread`.

Registers that drive in opposite directions and connect to high fan-out can have placement-warping effects on the floorplan that impact  $f_{MAX}$ . The placement-warping may not cause timing failures. Therefore, you can view this report to identify such registers. Taking steps to address the registers listed in the report can make placement of the design easier and improve  $f_{MAX}$  performance.

You can automate duplication of registers with the `DUPLICATE_REGISTER` and `DUPLICATE_HIERARCHY_DEPTH` .qsf assignments, or you can manually modify RTL to duplicate registers or refactor logic. Refer to "Automatic Register Duplication: Hierarchical Proximity" in *Intel Quartus Prime Pro Edition User Guide: Design Optimization*.

**Figure 31. Report Register Spread Report**

Register Name	Register_Location	Number of Endpoints	Endpoint Centroid	Total Distance of Endpoint
counter_a[0]~ERTM	(69, 67)	78	(69, 67)	117.0
counter_a[1]~ERTM	(68, 67)	39	(68, 66)	21.4
counter_b[1]~ERTM	(70, 67)	39	(70, 67)	21.4
counter_b[2]~ERTM	(70, 67)	38	(70, 67)	20.9
counter_a[2]~ERTM	(68, 67)	38	(68, 66)	20.9
counter_b[3]~ERTM	(70, 67)	37	(70, 66)	20.4

You can specify various options to customize the report.



Figure 32. Report Register Spread Dialog Box

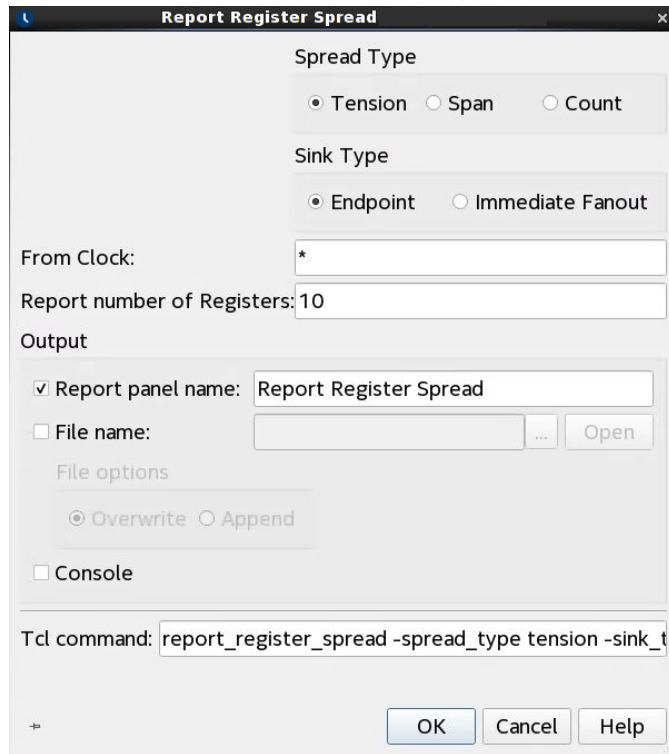


Table 19. Report Register Spread Settings

Option	Available Settings
<b>Spread Type</b>	Specifies the type of spread data in the report: <ul style="list-style-type: none"> <li>• <b>Tension</b>—reports the sum over each sink of the distance from it to the centroid of all the sinks.</li> <li>• <b>Span</b>—reports the maximum 1-dimensional delta between the left bottom-most sink and the right top-most sink.</li> <li>• <b>Count</b>—reports registers with the largest sink counts.</li> </ul>
<b>Sink Type</b>	Specifies the type of sink in the report: <ul style="list-style-type: none"> <li>• <b>Endpoint</b>—the nodes (usually registers) that terminate timing paths from a register.</li> <li>• <b>Immediate Fanout</b>—the immediately connected nodes of the register. For example, lookup tables, other registers, RAM, or DSP blocks.</li> </ul>
<b>From Clock</b>	Filters paths in the report to show only the launching clocks you specify.
<b>Report number of register</b>	Specifies the number of registers to display in the report. The default value for <b>Report number of registers</b> is 10.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.



### 5.5.4.5. Report Route Net of Interest

The Timing Analyzer's **Reports > Design Metrics > Report Route Net of Interest** command reports the nets that require the most effort from the router. The report shows the percentage of total router effort for the nets reported. The equivalent scripting command is `report_route_net_of_interest`.

This report help you to identify nets that should not require significant router effort. For example, you might expect that a low speed management interface nets would not be timing critical, and would therefore not require much router effort. However, if **Report Route Net of Interest** reports that some nets in the low speed management interface require significant effort from the router, you could investigate that further. The investigation can determine whether the timing constraints are correct, whether the fan-out is significant and can reduce through driver duplication, or whether the net passes through congested areas.

Figure 33. Report Route Net of Interest Report

	Net Driver	Route Effort (%)
1	LED~2	3.559
2	reset	3.185
3	my_data_src7 myreg[19]	0.361
4	my_data_src10 myreg[8]	0.359
5	my_data_src6 myreg[17]	0.301
6	my_data_src7 i145~0	0.298
7	my_mlab_inst0 my_mlab_ins...ncram_impl1 rdaddr_reg[0]	0.298
8	my_mlab_inst0 my_mlab_ins...ncram_impl1 rdaddr_reg[1]	0.273
9	my_mlab_inst1 my_mlab_ins...ncram_impl1 rdaddr_reg[1]	0.260
10	my_mlab_inst0 my_mlab_ins...ncram_impl1 rdaddr_reg[1]	0.247

Figure 34. Report Route Net of Interest Dialog Box

**Report Route Net of Interest**

Nets

Maximum number of nets to report: 50

Output

Report panel name: Report Route Net of Interest

File name: [ ] ... Open

File options

Overwrite  Append

Console

Tcl command: `report_route_net_of_interest -num_nets 50 -pane`

OK Cancel Help



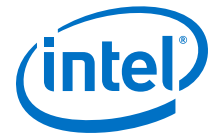


Table 20. Report Route Net of Interest Settings

Option	Available Settings
<b>Nets</b>	Specifies the <b>Maximum number of nets to report</b> . The default value is 50.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.

#### 5.5.4.6. Report Hierarchical Retiming Restrictions

The Timing Analyzer's **Reports > Design Metrics > Report Hierarchical Retiming Restrictions** command reports a summary of the number of Power-up "Care" Restrictions and don't touch information for each port. You can refer to this report to improve the circuit and remove retiming restrictions that limit circuit performance. `report_hierarchical_retiming_restrictions` is the equivalent scripting command.

Figure 35. Report Hierarchical Retiming Restrictions Report

	Compilation Hierarchy Node	Power-up "Care" Restriction	Asynchronous clear port	Drives an asynchronous signal
1		384 (172)	0 (0)	0 (0)
1	LED	5 (5)	0 (0)	0 (0)
2	add_0	40 (40)	0 (0)	0 (0)
3	add_1	39 (39)	0 (0)	0 (0)
4	clk	3 (3)	0 (0)	0 (0)

For table entries with two number values, the number in parentheses indicates the number of retiming restrictions in the specific entity alone. The number listed outside of parentheses indicates the number of retiming restrictions in the specific entity and all of its sub-entities in the hierarchy.

#### 5.5.4.7. Report Pipelining Information

The Timing Analyzer's **Reports > Design Metrics > Report Pipelining Information** command generates a report that can help you to identify potential areas of over-pipelining in your design. Excessive pipelining unnecessarily consumes area. The equivalent scripting command is `report_pipelining_info`.

Figure 36. Report Pipelining Information Report

	Full Hierarchy Name	Average Distance Per Stage	Max Distance Per Stage	Min Distance Per Stage	Bus Width	Bus
1	my_data...a_out_r1	0.3	0.7	0.0	18	6.0
2	my_data...a_out_r1	0.2	0.8	0.0	18	6.0
3	my_data...a_out_r1	0.2	0.6	0.0	20	5.0
4	my_data...a_out_r1	0.2	0.8	0.0	20	5.0
5	my_data...a_out_r1	0.2	0.5	0.0	9	6.0

To help identify potential over-pipelining, **Report Pipelining Information** reports:

- The width of buses in your design
- The number of registers on the bus
- The distance between the registers
- The number of sequential registers

The Average Distance Per Stage, Max Distance Per Stage, and Min Distance Per Stage columns report the Manhattan distance measured in logic array blocks (LABs). The Bus Average Depth, Bus Max Depth, and Bus Min Depth columns report the number of sequential, single fan-out registers.

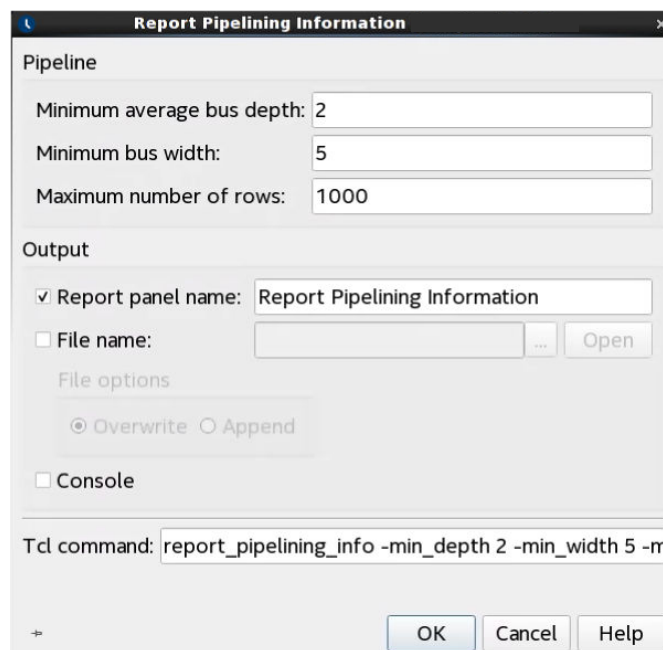
If the report identifies a large register chain with multiple sequential registers, and the distance between registers is low, that condition can suggest over-pipelining. You may be able to remove some registers to recover some of the device area and reduce congestion.

The following options are available for this report:

**Table 21. Report Pipelining Information Settings**

Option	Available Settings
<b>Pipeline</b>	Specifies the thresholds for reporting a register pipeline. You can define the <b>Minimum average bus depth</b> , the <b>Minimum bus width</b> , and the <b>Maximum number of rows</b> that the report includes.
<b>Report panel name</b>	Specifies the name of the report panel. You can optionally enable <b>File name</b> to write the information to a file. If you append <code>.htm</code> or <code>.html</code> as a suffix, the Timing Analyzer produces the report as HTML. If you enable <b>File name</b> , you can <b>Overwrite</b> or <b>Append</b> the file with latest data.
<b>Tcl command</b>	Displays the Tcl syntax that corresponds with the GUI options you select. You can copy the command from the <b>Console</b> into a Tcl file.

**Figure 37. Report Pipelining Information Dialog Box**





### 5.5.4.8. Report CDC Viewer

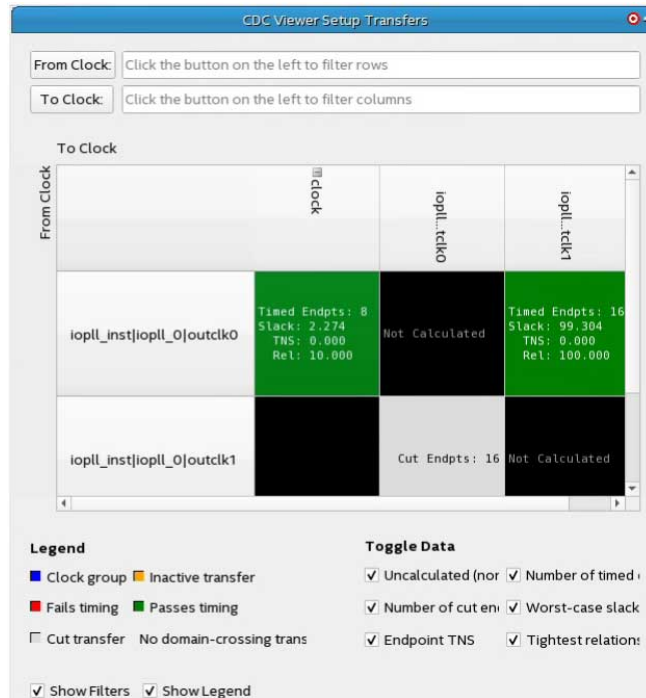
The Timing Analyzer's **Reports** > **Diagnostic** > **Report CDC Viewer** command displays the Clock Domain Crossing (CDC) Viewer. The CDC Viewer graphically displays the setup, hold, recovery, or removal analysis of all clock transfers in your design. The equivalent scripting command is `report_cdc_viewer`.

You can specify the following options to customize CDC Viewer reporting:

**Table 22. Setup Transfers Report Controls**

Control	Description
<b>From Clock:</b> and <b>To Clock:</b>	Filters the display according to the clock names you specify. Click <b>From Clock:</b> or <b>To Clock:</b> to search for specific clock names.
<b>Legend</b>	Defines the status colors. A color coded grid displays the clock transfer status. The clock headers list each clock with transfers in the design. The GUI truncates long clock names, but you can view the full name in a tool tip or by resizing the clock header cell. The GUI represents the generated clocks as children of the parent clock. A '+' icon next to a clock name indicates the presence of generated clocks. Clicking on the clock header displays the generated clocks associated with that clock.
<b>Toggle Data</b>	The text in each transfer cell contains data specific to each transfer. Turn on or off display of the following types of data: <ul style="list-style-type: none"> <li>• <b>Number of timed endpoints</b> between clocks— the number of timed, endpoint-unique paths in the transfer. A path being "timed" means that analysis occurs on that path. Only paths with unique endpoints count towards this total.</li> <li>• <b>Number of cut endpoints</b> between clocks— the number of cut endpoint-unique paths, instead of timed paths. These paths are cut by either a false path or clock group assignment. Timing analysis skips such paths.</li> <li>• <b>Worst-case slack</b> between clocks— the worst-case slack among all endpoint-unique paths in the transfer.</li> <li>• <b>Total negative slack</b> between clocks— the sum of all negative slacks among all endpoint-unique paths in this transfer.</li> <li>• <b>Tightest relationship</b> between clocks— the lowest-value setup, hold, recovery, or removal relationship between the two clocks in this transfer.</li> </ul>
<b>Show Filters</b> and <b>Show Legend</b>	Turns on or off Filters and <b>Legend</b> .

Figure 38. CDC Viewer Setup Transfers Report

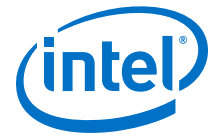


Each block in the grid is a transfer cell. Each transfer cell uses color and text to display important details of the paths in the transfer. The color coding represents the following states:

Table 23. Transfer Cell Content

Cell Color	Color Legend
Black	Indicates no transfers. There are no paths crossing between the source and destination clock of this cell.
Green	Indicates passing timing. All timing paths in this transfer, that have not been cut, meet their timing requirements.
Red	Indicates failing timing. One or more of the timing paths in the transfer do not meet their timing requirements. If the transfer is between unrelated clocks, the paths likely require a synchronizer chain.
Blue	Indicates clock groups. The source and destination clocks of these transfers are cut by means of asynchronous clock groups.
Gray	Indicates a cut transfer. All paths in this transfer are cut by false paths. Therefore, timing analysis does not consider these paths.
Orange	Indicates inactive clocks. One of the clocks in the transfer is an inactive clock (with the <code>set_active_clocks</code> command). The Timing Analyzer ignores such transfers.

Right-click menus allow you to perform operations on transfer cells and clock headers. When the operation is a Timing Analyzer report or SDC command, a dialog box opens containing the contents of the transfer cell.



**Table 24. Transfer Cell Right-Click Menus**

Command	Description
<b>Copy</b>	Copies the contents of the transfer cell or clock header to the clipboard.
<b>Report Timing</b>	Reports timing. Not available for transfer cells with no valid paths (gray or black cells).
<b>Report Endpoints</b>	Reports endpoints. Not available for transfer cells with no cut paths (gray or black cells).
<b>Report False Path</b>	Reports false paths. Not available for transfer cells with no valid paths (black cells).
<b>Report Exceptions</b>	Reports exceptions. Only available for clock group transfers (blue cells).
<b>Report Exceptions (with clock groups)</b>	Reports exceptions with clock groups. Only available for clock group transfers (blue cells).
<b>Set False Path</b>	Sets a false path constraint.
<b>Set Multicycle Path</b>	Sets a multicycle path exception.
<b>Set Min Delay</b>	Sets a min delay constraint.
<b>Set Max Delay</b>	Sets a max delay constraint.
<b>Set Clock Uncertainty</b>	Sets a clock uncertainty constraint.

**Table 25. Clock Header Right-Click Menus**

Command	Description
<b>Copy (include children)</b>	Copies the name of the clock header, and the names of each of its derived clocks. This option only appears for clock headers with generated clocks.
<b>Expand/Collapse All Rows/Columns</b>	Shows or hides all derived clocks in the grid.
<b>Create Slack Histogram</b>	Generates a slack histogram report for the clock you select.
<b>Report Timing From/To Clock</b>	Generates a timing report for the clock you select. If you do not expand the clock to display derived clocks, the timing report includes all clocks that derive from the clock. To prevent this, expand the clock before right-clicking it.
<b>Remove Clock(s)</b>	Removes the clock you select from the design. If you do not expand the clock, timing analysis removes all clocks that derive from the clock.

You can view CDC Viewer output in any of the following formats:

- A report panel in the Timing Analyzer
- Output in the Timing Analyzer Tcl console
- A plain-text file
- An HTML file you can view in a web browser.

**Related Information**

[Tips for Analyzing Failing Clock Paths that Cross Clock Domains](#) on page 109

**5.5.4.9. Timing Closure Recommendations**

The **Report Timing Closure Recommendations** command in the Timing Analyzer **Task** pane analyzes paths and provides specific recommendations based on path characteristics. Since Design Assistant now provides more targeted timing closure recommendations, **Report Timing Closure Recommendations** is marked for deprecation.

### 5.5.4.10. Global Network Buffers

Routing paths allow you to identify global network buffers that fail timing. Buffer locations names reflect the network they drive.

- CLK\_CTRL\_Gn—for Global driver
- CLK\_CTRL\_Rn—for Regional driver

Buffers that access the global networks are in the center of each side of the device. Buffering to route a core logic signal on a global signal network causes insertion delay. Trade-offs to consider for global and non-global routing are source location, insertion delay, fan-out, distance a signal travels, and possible congestion if the signal demotes to local routing.

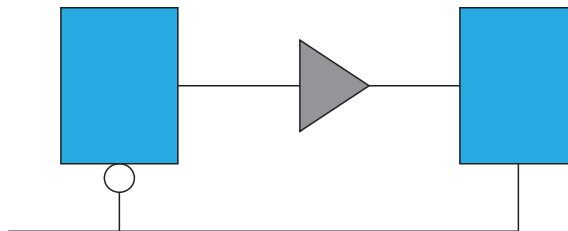
#### 5.5.4.10.1. Source Location

If you cannot move the register feeding the global buffer closer, then consider changing either the design logic or the routing type.

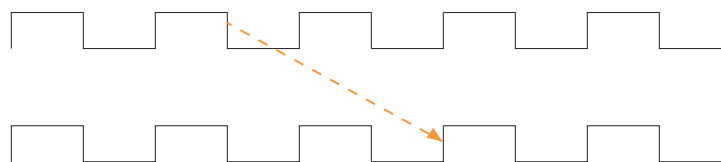
#### 5.5.4.10.2. Insertion Delay

If the design requires a global signal, consider adding half a cycle to timing by using a negative-edge triggered register to generate the signal, and use a multicycle setup constraint.

**Figure 39. Negative-Edge Triggered Register**



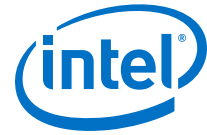
**Figure 40. Multicycle Setup Constraint**



```
set_multicycle_path -from <generating_register> -setup -end 2
```

#### 5.5.4.10.3. Fan-Out

Nodes with very high fan-out that use local routing tend to pull logic that they drive close to the source node. This can make other paths fail timing. Duplicating registers can help reduce the impact of high fan-out paths. Consider manually duplicating and preserving these registers. Using a MAX\_FANOUT assignment may make arbitrary groups of fan-out nodes, whereas a designer can make more intelligent fan-out groups.



#### 5.5.4.10.4. Global Signal Assignment

You can use the Global Signal assignment to control the global signal usage on a per-signal basis. For example, if a signal needs local routing, you set the Global Signal assignment to **Off**.

#### 5.5.4.11. Resets and Global Networks

The Compiler often routes reset signals on global networks. Sometimes, the use of a global network causes recovery failures. Consider reviewing the placement of the register that generates the reset and the routing path of the signal.

#### 5.5.4.12. Suspicious Setup

Suspicious setup failures include paths with very small or very large requirements.

One typical cause is math precision error. For example,  $10\text{MHz}/3 = 33.33$  ns per period. In three cycles, the time is 99.999 ns vs 100.000 ns. Setting a maximum delay can provide an appropriate setup relationship.

Another cause of failure are paths that must be false by design intent, such as:

- Asynchronous paths handled through FIFOs, or
- Slow asynchronous paths that rely on handshaking for data that remain available for multiple clock cycles.

To prevent the Fitter from having to meet unnecessarily restrictive timing requirements, consider adding false or multicycle path statements.

#### 5.5.4.13. Auto Shift Register Replacement

During synthesis, the Compiler can convert shift registers or register chains into RAMs to save area. However, conversion to RAM often reduces speed. The Compiler names the converted registers with the prefix "altshift\_taps".

- If paths that fail timing begin or end in shift registers, consider disabling the **Auto Shift Register Replacement** option. Do not convert registers that are intended for pipelining.
- For shift registers that are converted to a chain, evaluate area/speed trade off of implementing in RAM or logic cells.
- If a design uses nearly the full device capacity, you can save area by shifting register conversion to RAM, benefiting non-critical clock domains. You can change the settings from the default **AUTO** to **OFF** globally, or on a register or hierarchy basis.

#### 5.5.4.14. Clocking Architecture

For better timing results, place all registers driven by a regional clock in one quadrant of the chip. You can review the clock region boundaries in the Chip Planner.

Timing failure can occur when the I/O interface at the top of the device connects to logic driven by a regional clock which is in one quadrant of the device, and placement restrictions force long paths to and from I/Os to logic across quadrants.

Use a different type of clock source to drive the logic, such as global, which covers the whole device, or dual-regional which covers half the device. Alternatively, you can reduce the frequency of the I/O interface to accommodate the long path delays. You can also redesign the pinout of the device to place all the specified I/Os adjacent to the regional clock quadrant. This issue can happen when register locations are restricted, such as with Logic Lock regions, clocking resources, or hard blocks (memories, DSPs, IPs).

The **Extra Fitter Information** tab in the Timing Analyzer timing report informs you when placement is restricted for nodes in a path.

#### Related Information

[Viewing Available Clock Networks in the Device](#) on page 124

### 5.5.5. Try Optional Fitter Settings

This section focuses only on the optional timing-optimization Fitter settings, which are the **Optimize Hold Timing**, **Optimize Multi-Corner Timing**, and **Fitter Aggressive Routability Optimization**.

**Caution:** The settings that best optimize different designs might vary. The group of settings that work best for one design does not necessarily produce the best result for another design.

#### Related Information

[Advanced Fitter Setting Dialog Box Help Topic](#)  
In *Intel Quartus Prime Help*

#### 5.5.5.1. Optimize Hold Timing

The **Optimize Hold Timing** option directs the Intel Quartus Prime software to optimize minimum delay timing constraints.

When you turn on **Optimize Hold Timing** in the **Advanced Fitter Settings** dialog box, the Intel Quartus Prime software adds delay to paths to ensure that your design meets the minimum delay requirements. If you select **I/O Paths and Minimum TPD Paths**, the Fitter works to meet the following criteria:

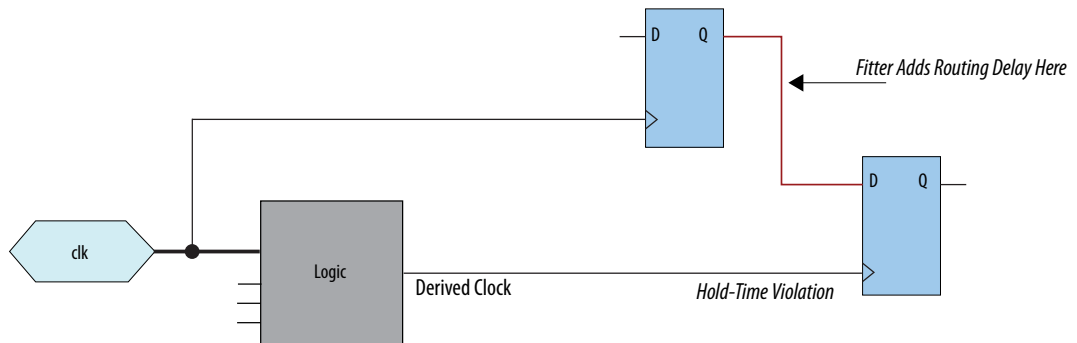
- Hold times ( $t_H$ ) from the device input pins to the registers
- Minimum delays from I/O pins to I/O registers or from I/O registers to I/O pins
- Minimum clock-to-out time ( $t_{CO}$ ) from registers to output pins

If you select **All Paths**, the Fitter also works to meet hold requirements from registers to registers, as highlighted in blue in the figure, in which a derived clock generated with logic causes a hold time problem on another register.





**Figure 41. Optimize Hold Timing Option Fixing an Internal Hold Time Violation**



However, if your design still has internal hold time violations between registers, you can manually add delays by instantiating LCELL primitives, or by making changes to your design, such as using a clock enable signal instead of a derived or gated clock.

**Related Information**

**Recommended Design Practices**

In *Intel Quartus Prime Pro Edition User Guide: Design Recommendations*

**5.5.5.2. Fitter Aggressive Routability Optimization**

The **Fitter Aggressive Routability Optimizations** logic option allows you to specify whether the Fitter aggressively optimizes for routability. Performing aggressive routability optimizations may decrease design speed, but may also reduce routing wire usage and routing time.

This option is useful if routing resources are resulting in no-fit errors, and you want to reduce routing wire use.

The table lists the settings for the **Fitter Aggressive Routability Optimizations** logic option.

**Table 26. Fitter Aggressive Routability Optimizations Logic Option Settings**

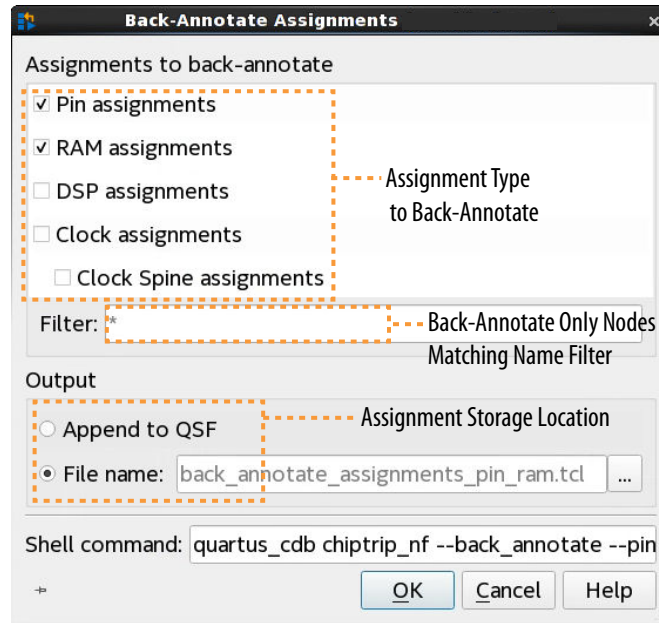
Settings	Description
Always	The Fitter always performs aggressive routability optimizations. If you set the <b>Fitter Aggressive Routability Optimizations</b> logic option to <b>Always</b> , reducing wire utilization may affect the performance of your design.
Never	The Fitter never performs aggressive routability optimizations. If improving timing is more important than reducing wire usage, then set this option to <b>Automatically</b> or <b>Never</b> .
Automatically	The Fitter performs aggressive routability optimizations automatically, based on the routability and timing requirements of the design. If improving timing is more important than reducing wire usage, then set this option to <b>Automatically</b> or <b>Never</b> .

**5.5.6. Back-Annotate Optimized Assignments**

The Compiler maps the elements of your design to specific device resources during fitting. After compilation, you can back-annotate (copy) the Compiler's resource assignments to preserve that same implementation in subsequent compilations. Back-annotation can simplify timing closure by allowing you to lock down placement of your optimized results.

Locking down placement of large blocks related to Clocks, RAMs, and DSPs can produce higher  $f_{MAX}$  with less noise. Large blocks like RAMs and DSPs have heavier connectivity than regular LABs, complicating movement during placement. When a seed produces good results from suitable RAM and DSP placement, you can capture that placement with back-annotation. Subsequent compiles can then benefit from the high quality RAM and DSP placement from the good seed.

**Figure 42. Back-Annotate Assignments Dialog Box**



To back-annotate (copy) the device resource assignments from the last compilation to the project `.qsf` (or to a Tcl file) for use in the next compilation:

1. Run a full compilation, or run the Fitter through at least the **Place** stage.
2. Click **Assignments** ► **Back-Annotate Assignments**.
3. Under **Assignments to back-annotate**, specify whether you want to preserve **Pin assignments**, **RAM assignments**, **DSP assignments**, **Clock assignments**, and **Clock Spine assignments** in the back-annotation.
4. In **Filter**, specify a text string (including wildcards) if you want to filter back-annotated assignments by entity name.
5. Under **Output**, specify whether to save the back-annotated assignments to the `.qsf` or to a Tcl file. A default Tcl file name displays.

Alternatively, you can run back-annotation with the following `quartus_cdb` executable. The **Shell command** field displays the shell command constructed by the options that you specifying the GUI.

```
quartus_cdb chiptrip_nf --back_annotate --pin --ram --dsp --clocks \
--spines --file "<file>.tcl"
```

**Note:** Check available arguments by running `quartus_cdb <project> --back_annotate --help`.



### 5.5.7. Optimize Settings with Design Space Explorer II

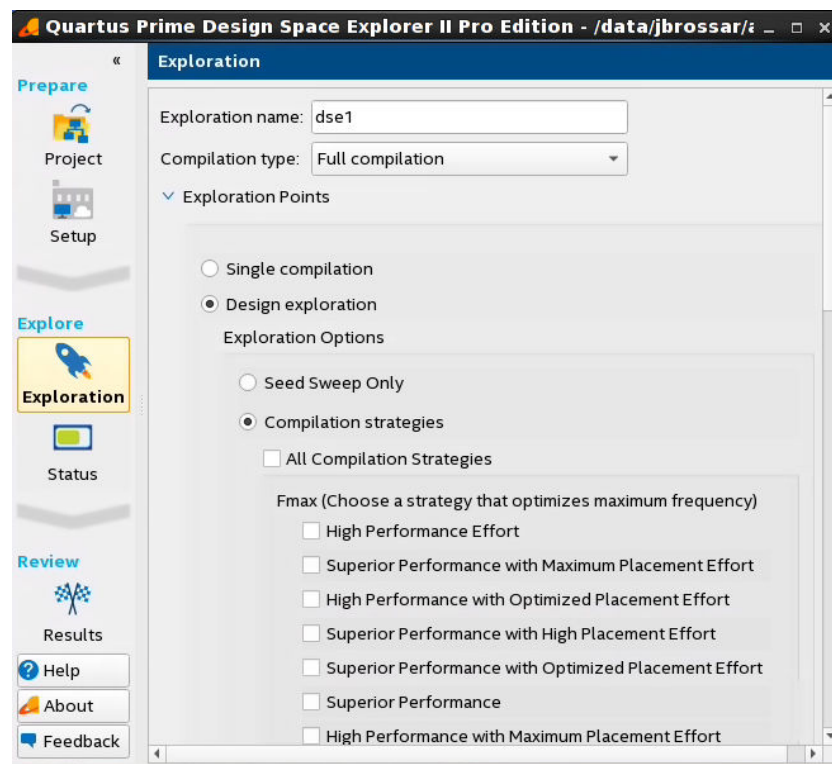
The Design Space Explorer II tool (**Tools > Launch Design Space Explorer II**) allows you to find optimal project settings for resource, performance, or power optimization goals. Design Space Explorer II (DSE II) processes a design using combinations of settings and constraints, and reports the best settings for the design. You can take advantage of the DSE II parallelization abilities to compile on multiple computers.

In DSE II, an *exploration point* is a collection of Analysis & Synthesis, Fitter, and placement settings, and a group of exploration points is a *design exploration*. A design exploration can also include different fitter seeds.

DSE II compiles the design using the settings corresponding to each exploration point. When the compilation finishes, DSE II evaluates the performance data against an optimization goal that you specify. You can direct the DSE II to optimize for timing, area, or power.

If a design is close to meeting timing or area requirements, you can try different seeds with the DSE II, and find one seed that meets timing or area requirements.

Figure 43. Design Space Explorer II



You can run DSE II at any step in the design process; however, because large changes in a design can neutralize gains achieved from optimizing settings, Intel FPGA recommends that you run DSE II late in the design cycle.



### Related Information

[Using Design Space Explorer](#)  
21 Minute Online Course

#### 5.5.7.1. DSE II Computing Resources

You can configure DSE II to take advantage of your computing resources to run the design explorations. In the DSE II GUI, the **Setup** page contains the job launch options, and the **Status** page allows you to monitor and control jobs.

DSE II supports running compilations on your local computer or a remote host through LSF, SSH or Torque. For SSH, you can also define a comma-separated list of remote hosts.

If you have a laptop or standard computer, you can use the single compilation feature to compile your design on a workstation with higher computing performance and memory capacity.

When running on a compute farm, you can direct the DSE II to safely exit after submitting all the jobs while the compilations continue to run until completion. Optionally, you can receive an e-mail when the compilations are complete.

If you launch jobs using SSH, the remote host must enable public and private key authentication. For private keys encrypted with a pass phrase, the remote host must run the ssh key agent to decrypt the private key, so the `quartus_dse` executable can access the key.

*Note:* Windows remote hosts require Cygwin's sshd server and PuTTY.

#### 5.5.7.2. DSE II Optimization Parameters

DSE II provides a collection of predefined exploration spaces that focus on what you want to optimize. Additionally, you can define a set of compilation seeds. The number of exploration points is the number of seeds multiplied by the number of exploration modes.

*Note:* The availability of predefined spaces depends on the device family that the design targets.

In the DSE GUI, you specify these settings in the **Exploration** page.

### Related Information

[Exploration Page \(Design Space Explorer II\)](#)  
In *Intel Quartus Prime Help*

#### 5.5.7.3. DSE II Result Management

DSE II compares the compilation results to determine the best Intel Quartus Prime software settings for the design. The **Report** page displays a summary of results.

In an exploration, DSE II selects the best worst-case slack value from among all timing corners across all exploration points. If you want to optimize for worst-case setup slack or hold slack, specify timing constraints in the Intel Quartus Prime software.



### Disk Space

By default, DSE II saves all the compilation data. You can save disk space by limiting the type of files that you want to save after a compilation finishes. These settings are in the **Exploration** page, **Results** section.

### Reports

DSE II has reporting tools that help you quickly determine important design metrics, such as worse-case slack, across all exploration points.

DSE II provides a performance data report for all points it explores and saves the information in a `project-name.dse.rpt` file in the project directory. DSE II archives the settings of the exploration points in Intel Quartus Prime Archive Files (`.qar`).

### Related Information

[Report Page \(Design Space Explorer II\)](#)

In *Intel Quartus Prime Help*

#### 5.5.7.4. Running DSE II

**Note:** Before running DSE II, specify the timing constraints for the design.

This description covers the type of settings that you need to define when you want to run a design exploration. For details about all the options available in the GUI, refer to the Intel Quartus Prime Help.

To perform a design exploration with the DSE II tool:

1. Start the DSE II tool.  
If you have an open project in the Intel Quartus Prime software and launch DSE II, a dialog box appears asking if you want to close the Intel Quartus Prime software. Click **Yes**.
2. In the **Project** page, specify the project and revision that you want to explore.
3. In the **Setup** page, specify whether you want to perform a local or a remote exploration, and set up the job launch.
4. In the **Exploration** page, specify optimization settings and goals.
5. When the configuration is complete, click **Start**.

#### 5.5.8. I/O Timing Optimization Techniques

This stage of design optimization focuses on I/O timing, including setup delay ( $t_{SU}$ ), hold time ( $t_H$ ), and clock-to-output ( $t_{CO}$ ) parameters.

Before proceeding with I/O timing optimization, ensure that:

- The design's assignments follow the suggestions in the *Initial Compilation: Required Settings* section of the *Design Optimization Overview* chapter.
- Resource utilization is satisfactory.

**Note:** Complete this stage before proceeding to the register-to-register timing optimization stage. Changes to the I/O paths affect the internal register-to-register timing.

### Summary of Techniques for Improving Setup and Clock-to-Output Times

The table lists the recommended order of techniques to reduce  $t_{SU}$  and  $t_{CO}$  times. Reducing  $t_{SU}$  times increases hold ( $t_H$ ) times.

*Note:* Verify which options are available to each device family

**Table 27. Improving Setup and Clock-to-Output Times**

Order	Technique	Affects $t_{SU}$	Affects $t_{CO}$
1	Verify of that the appropriate constraints are set for the failing I/Os (refer to <i>Initial Compilation: Required Settings</i> )	Yes	Yes
2	Use timing-driven compilation for I/O (refer to <i>Fast Input, Output, and Output Enable Registers</i> )	Yes	Yes
3	Use fast input register (refer to <i>Programmable Delays</i> )	Yes	N/A
4	Use fast output register, fast output enable register, and fast OCT register (refer to <i>Programmable Delays</i> )	N/A	Yes
5	Decrease the value of <b>Input Delay from Pin to Input Register</b> or set <b>Decrease Input Delay to Input Register = ON</b>	Yes	N/A
6	Decrease the value of <b>Input Delay from Pin to Internal Cells</b> or set <b>Decrease Input Delay to Internal Cells = ON</b>	Yes	N/A
7	Decrease the value of <b>Delay from Output Register to Output Pin</b> or set <b>Increase Delay to Output Pin = OFF</b> (refer to <i>Fast Input, Output, and Output Enable Registers</i> )	N/A	Yes
8	Increase the value of <b>Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations</b> (refer to <i>Fast Input, Output, and Output Enable Registers</i> )	Yes	N/A
9	<b>Use PLLs to shift clock edges</b>	Yes	Yes
10	Increase the value of <b>Delay to output enable pin</b> or set <b>Increase delay to output enable pin</b> (refer to <i>Use PLLs to Shift Clock Edges</i> )	N/A	Yes

[I/O Timing Constraints](#) on page 94

[Optimize IOC Register Placement for Timing Logic Option](#) on page 95

[Fast Input, Output, and Output Enable Registers](#) on page 95

[Programmable Delays](#) on page 96

[Use PLLs to Shift Clock Edges](#) on page 97

[Use Fast Regional Clock Networks and Regional Clocks Networks](#) on page 97

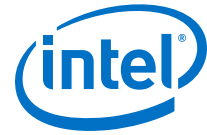
[Spine Clock Limitations](#) on page 97

#### Related Information

[Required Settings for Initial Compilation](#) on page 7

### 5.5.8.1. I/O Timing Constraints

Timing Analyzer supports the Synopsys\* Design Constraints (SDC) format for constraining your design. When using the Timing Analyzer for timing analysis, use the `set_input_delay` constraint to specify the data arrival time at an input port with respect to a given clock. For output ports, use the `set_output_delay` command to specify the data arrival time at an output port's receiver with respect to a given clock. You can use the `report_timing` Tcl command to generate the I/O timing reports.



The I/O paths that do not meet the required timing performance are reported as having negative slack and are highlighted in red in the Timing Analyzer **Report** pane. In cases where you do not apply an explicit I/O timing constraint to an I/O pin, the Intel Quartus Prime timing analysis software still reports the **Actual** number, which is the timing number that must be met for that timing parameter when the device runs in your system.

#### Related Information

##### [Creating I/O Requirements](#)

In *Intel Quartus Prime Pro Edition User Guide: Timing Analyzer*

### 5.5.8.2. Optimize IOC Register Placement for Timing Logic Option

This option moves registers into I/O elements to meet  $t_{SU}$  or  $t_{CO}$  assignments, duplicating the register if necessary (as in the case in which a register fans out to multiple output locations). This option is turned on by default and is a global setting.

The **Optimize IOC Register Placement for Timing** logic option affects only pins that have a  $t_{SU}$  or  $t_{CO}$  requirement. Using the I/O register is possible only if the register directly feeds a pin or is fed directly by a pin. Therefore, this logic option does not affect registers with any of the following characteristics:

*Note:* To optimize registers with these characteristics, use other Intel Quartus Prime Fitter optimizations.

- Have combinational logic between the register and the pin
- Are part of a carry chain
- Have an overriding location assignment
- Use the asynchronous load port and the value is not 1 (in device families where the port is available)

#### Related Information

##### [Optimize IOC Register Placement for Timing Logic Option Help Topic](#)

In *Intel Quartus Prime Help*

### 5.5.8.3. Fast Input, Output, and Output Enable Registers

You can place individual registers in I/O cells manually by making fast I/O assignments with the Assignment Editor. By default, with correct timing assignments, the Fitter places the I/O registers in the correct I/O cell or in the core, to meet the performance requirement.

If the fast I/O setting is on, the register is always placed in the I/O element. If the fast I/O setting is off, the register is never placed in the I/O element. This is true even if the **Optimize IOC Register Placement for Timing** option is turned on. If there is no fast I/O assignment, the Intel Quartus Prime software determines whether to place registers in I/O elements if the **Optimize IOC Register Placement for Timing** option is turned on.

You can also use the four fast I/O options (**Fast Input Register**, **Fast Output Register**, **Fast Output Enable Register**, and **Fast OCT Register**) to override the location of a register that is in a Logic Lock region and force it into an I/O cell. If you apply this assignment to a register that feeds multiple pins, the Fitter duplicates the register and places it in all relevant I/O elements.

For more information about the **Fast Input Register** option, **Fast Output Register** option, **Fast Output Enable Register** option, and **Fast OCT (on-chip termination) Register** option, refer to Intel Quartus Prime Help.

#### Related Information

- [Fast Input Register logic option Help Topic](#)
- [Fast Output Register logic option](#)
- [Fast Output Enable Register logic option](#)
- [Fast OCT Register logic option](#)

#### 5.5.8.4. Programmable Delays

You can use various programmable delay options to minimize the  $t_{SU}$  and  $t_{CO}$  times. Programmable delays are advanced options that you use only after you compile a project, check the I/O timing, and determine that the timing is unsatisfactory.

The Intel Quartus Prime software automatically adjusts the applicable programmable delays to help meet timing requirements. For detailed information about the effect of these options, refer to the device family handbook or data sheet.

After you have made a programmable delay assignment and compiled the design, you can view the implemented delay values for every delay chain and every I/O pin in the **Delay Chain Summary** section of the Compilation Report.

You can assign programmable delay options to supported nodes with the Assignment Editor. You can also view and modify the delay chain setting for the target device with the Chip Planner and Resource Property Editor. When you use the Resource Property Editor to make changes after performing a full compilation, recompiling the entire design is not necessary; you can save changes directly to the netlist. Because these changes are made directly to the netlist, the changes are not made again automatically when you recompile the design. The change management features allow you to reapply the changes on subsequent compilations.

Although the programmable delays in newer devices are user-controllable, Intel recommends their use for advanced users only. However, the Intel Quartus Prime software might use the programmable delays internally during the Fitter phase.

For details about the programmable delay logic options available for Intel devices, refer to the following Intel Quartus Prime Help topics:

#### Related Information

- [Input Delay from Pin to Input Register logic option Help Topic](#)
- [Input Delay from Pin to Internal Cells logic option Help Topic](#)
- [Output Enable Pin Delay logic option Help Topic](#)
- [Delay from Output Register to Output Pin logic option Help Topic](#)
- [Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations logic option Help Topic](#)

In *Intel Quartus Prime Help*

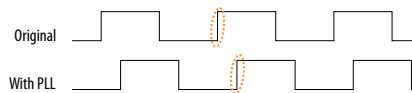




### 5.5.8.5. Use PLLs to Shift Clock Edges

Using a PLL typically improves I/O timing automatically. If the timing requirements are still not met, most devices allow the PLL output to be phase shifted to change the I/O timing. Shifting the clock backwards gives a better  $t_H$  at the expense of  $t_{SU}$ , while shifting it forward gives a better  $t_{SU}$  at the expense of  $t_H$ . You can use this technique only in devices that offer PLLs with the phase shift option.

**Figure 44. Shift Clock Edges Forward to Improve  $t_{SU}$  at the Expense of  $t_H$**



You can achieve the same type of effect in certain devices by using the programmable delay called **Input Delay from Dual Purpose Clock Pin to Fan-Out Destinations**.

### 5.5.8.6. Use Fast Regional Clock Networks and Regional Clocks Networks

Regional clocks provide the lowest clock delay and skew for logic contained in a single quadrant. In general, fast regional clocks have less delay to I/O elements than regional and global clocks, and are used for high fan-out control signals. Placing clocks on these low-skew and low-delay clock nets provides better  $t_{CO}$  performance.

Intel devices have a variety of hierarchical clock structures. These include dedicated global clock networks, regional clock networks, fast regional clock networks, and periphery clock networks. The available resources differ between the various Intel device families.

For the number of clocking resources available in your target device, refer to the appropriate device handbook.

### 5.5.8.7. Spine Clock Limitations

In projects with high clock routing demands, limitations in the Intel Quartus Prime software can cause spine clock errors. These errors are often seen with designs using multiple memory interfaces and high-speed serial interface (HSSI) channels, especially PMA Direct mode.

Global clock networks, regional clock networks, and periphery clock networks have an additional level of clock hierarchy known as spine clocks. Spine clocks drive the final row and column clocks to their registers; thus, the clock to every register in the chip is reached through spine clocks. Spine clocks are not directly user controllable.

To reduce these spine clock errors, constrain your design to use your regional clock resources better:

- If your design does not use Logic Lock regions, or if the Logic Lock regions are not aligned to your clock region boundaries, create additional Logic Lock regions and further constrain your logic.
- If Periphery features ignore Logic Lock region assignment, possibly because the global promotion process is not functioning properly. To ensure that the global promotion process uses the correct locations, assign specific pins to the I/Os using these periphery features.
- By default, some Intel FPGA IP functions apply a global signal assignment with a value of dual-regional clock. If you constrain your logic to a regional clock region and set the global signal assignment to **Regional** instead of **Dual-Regional**, you can reduce clock resource contention.

#### Related Information

- [Viewing Available Clock Networks in the Device](#) on page 124
- [Layers Settings](#) on page 122
- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)  
In *Intel Quartus Prime Help*

### 5.5.9. Register-to-Register Timing Optimization Techniques

The next stage of design optimization seeks to improve register-to-register ( $f_{MAX}$ ) timing. The following sections provide available options if the design does not meet timing requirements after compilation.

Coding style affects the performance of a design to a greater extent than other changes in settings. Always evaluate the code and make sure to use synchronous design practices.

*Note:* In the context of the Timing Analyzer, register-to-register timing optimization is the same as maximizing the slack on the clock domains in a design. The techniques in this section can improve the slack on different timing paths in the design.

Before performing design optimizations, understand the structure of the design as well as the effects of techniques in different types of logic. Techniques that do not benefit the logic structure can decrease performance.

#### Related Information

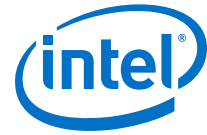
##### [Recommended Design Practices](#)

In *Intel Quartus Prime Pro Edition User Guide: Design Recommendations*

#### 5.5.9.1. Optimize Source Code

In many cases, optimizing the design's source code can have a very significant effect on your design performance. In fact, optimizing your source code is typically the most effective technique for improving the quality of your results and is often a better choice than using Logic Lock or location assignments.

Be aware of the number of logic levels needed to implement your logic while you are coding. Too many levels of logic between registers might result in critical paths failing timing. Try restructuring the design to use pipelining or more efficient coding



techniques. Also, try limiting high fan-out signals in the source code. When possible, duplicate and pipeline control signals. Make sure the duplicate registers are protected by a preserve attribute, to avoid merging during synthesis.

If the critical path in your design involves memory or DSP functions, check whether you have code blocks in your design that describe memory or functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to cause these functions to be placed into high-performance dedicated memory or resources in the target device. When using RAM/DSP blocks, enable the optional input and output registers.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Intel Quartus Prime software, you can check the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including state encoding for each state machine that was recognized during compilation. If your state machine is not recognized, you might have to change your source code to enable it to be recognized.

#### Related Information

[AN 584: Timing Closure Methodology for Advanced FPGA Designs](#)

### 5.5.9.2. Improving Register-to-Register Timing

The choice of options and settings to improve the timing margin (slack) or to improve register-to-register timing depends on the failing paths in the design. To achieve the results that best approximate your performance requirements, apply the following techniques and compile the design after each step:

1. Ensure that your timing assignments are complete and correct. For details, refer to the *Initial Compilation: Required Settings* section in the *Design Optimization Overview* chapter.
2. Review all warning messages from your initial compilation and check for ignored timing assignments.
3. Apply netlist synthesis optimization options.
4. To optimize for speed, apply the following synthesis options:
  - Optimize Synthesis for Speed, Not Area
  - Flatten the Hierarchy During Synthesis
  - Set the Synthesis Effort to High
  - Prevent Shift Register Inference
  - Use Other Synthesis Options Available in Your Synthesis Tool
5. To optimize for performance, turn on Advanced Physical Optimization
6. Try different Fitter seeds. If only a small number of paths are failing by small negative slack, then you can try with a different seed to find a fit that meets constraints in the Fitter seed noise.

*Note:* Omit this step if a large number of critical paths are failing, or if the paths are failing by a long margin.

7. To control placement, make Logic Lock assignments.
8. Modify your design source code to fix areas of the design that are still failing timing requirements by significant amounts.
9. Make location assignments, or as a last resort, perform manual placement by back-annotating the design.

You can use Design Space Explorer II (DSE) to automate the process of running different compilations with different settings.

If these techniques do not achieve performance requirements, additional design source code modifications might be required.

#### Related Information

- [Optimize Settings with Design Space Explorer II](#) on page 91
- [Required Settings for Initial Compilation](#) on page 7

### 5.5.9.3. Physical Synthesis Optimizations

The Intel Quartus Prime software offers physical synthesis optimizations that can help improve design performance regardless of the synthesis tool. You can apply physical synthesis optimizations both during synthesis and during fitting.

During the synthesis stage of the Intel Quartus Prime compilation, physical synthesis optimizations operate either on the output from another EDA synthesis tool, or as an intermediate step in synthesis. These optimizations modify the synthesis netlist to improve either area or speed, depending on the technique and effort level you select.

To view and modify the synthesis netlist optimization options, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**.

If you use a third-party EDA synthesis tool and want to determine if the Intel Quartus Prime software can remap the circuit to improve performance, use the **Perform WYSIWYG Primitive Resynthesis** option. This option directs the Intel Quartus Prime software to un-map the LEs in an atom netlist to logic gates, and then map the gates back to Intel-specific primitives. Intel-specific primitives enable the Fitter to remap the circuits using architecture-specific techniques.

The Intel Quartus Prime technology mapper optimizes the design to achieve maximum speed performance, minimum area usage, or balances high performance and minimal logic usage, according to the setting of the **Optimization Technique** option. Set this option to **Speed** or **Balanced**.

During the Fitter stage of the Intel Quartus Prime compilation, physical synthesis optimizations make placement-specific changes to the netlist that improve speed performance results for the specific Intel device.

#### Related Information

- [Perform WYSIWYG Primitive Resynthesis Logic Option Help Topic](#)
- [Optimization Technique Logic Option Help Topic](#)  
In *Intel Quartus Prime Help*



#### 5.5.9.4. Set Power Optimization During Synthesis to Normal Compilation

The default value for the Compiler's **Power Optimization During Synthesis** setting is **Normal Compilation**. However, if **Power Optimization During Synthesis** is set to **Extra Effort**, design performance can be affected. To avoid any possible effect, click **Assignments > Settings > Compiler Settings > Power Optimization During Synthesis** to confirm the **Normal Compilation** setting value.

##### Related Information

- [Power Optimization](#)
- [Power Optimization Logic Option Help Topic](#)  
In *Intel Quartus Prime Help*

#### 5.5.9.5. Optimize Synthesis for Speed, Not Area

Design performance varies depending on coding style, synthesis tool used, and options you specify when synthesizing. Change your synthesis options if a large number of paths are failing, or if specific paths fail by a great margin and have many levels of logic.

Identify the default optimization targets of your Synthesis tool, and set your device and timing constraints accordingly. For example, if you do not specify a target frequency, some synthesis tools optimize for area.

You can specify logic options for specific modules in your design with the Assignment Editor while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Area** (if area is an important concern). You can also use the **Speed Optimization Technique for Clock Domains** option in the Assignment Editor to specify that all combinational logic in or between the specified clock domains are optimized for speed.

##### Related Information

- [Optimization Technique Logic Option Help Topic](#)  
In *Intel Quartus Prime Help*

#### 5.5.9.6. Flatten the Hierarchy During Synthesis

Synthesis tools typically let you preserve hierarchical boundaries, which can be useful for verification or other purposes. However, the best optimization results generally occur when the synthesis tool optimizes across hierarchical boundaries, because doing so often allows the synthesis tool to perform the most logic minimization, which can improve performance. Whenever possible, flatten your design hierarchy to achieve the best results.

#### 5.5.9.7. Set the Synthesis Effort to High

Synthesis tools offer varying synthesis effort levels to trade off compilation time with synthesis results. Set the synthesis effort to **high** to achieve best results when applicable.

### 5.5.9.8. Duplicate Registers for Fan-Out Control

Often, timing failures can occur due to the influence of signals that are not directly involved in the failing transfers. This condition tends to manifest when off-critical nets, most commonly with a high fan-out, span a large distance and consequentially, warp the optimization of other paths around them.

Duplicating the sources of these types of globally-influential signals can help to disperse them across many hops, or even across many clock cycles, and focus more on local transfers.

For example, by duplicating a high fan-out signal in the form of a tree of registers, you can disperse the signal over several clock cycles. As the signal progresses down the tree, it progressively feeds more into local copies of the original registers, such that any individual register's destinations are well-localized and its influence on register optimization is minimal. The key to this optimization is to determine how to assign the original signal's fan-outs among the duplicates. If any individual register requires driving a large distance, the benefit of the tree can be removed.

You can manually create a register tree and group the endpoints in the RTL by leveraging your system-level knowledge about how best to disperse the signal throughout your design, but it can be time consuming and have a widespread impact. For more information about manually creating a register tree, refer to [Manual Register Duplication](#) on page 102.

You can create register trees automatically in one of the following ways.

- [Estimated Physical Proximity](#)
- [Hierarchical Proximity](#)

Each method has its own methodology to determine the number of duplicates to create and how to assign the fan-outs between the duplicates.

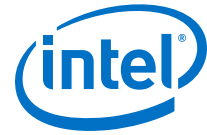
#### 5.5.9.8.1. Manual Register Duplication

Synthesis tools support options or attributes that specify the maximum fan-out of a register. When using Intel Quartus Prime synthesis, you can set the **Maximum Fan-Out** logic option in the Assignment Editor to control the number of destinations for a node so that the fan-out count does not exceed a specified value. You can also use the `maxfan` attribute in your HDL code. The software duplicates the node as required to achieve the specified maximum fan-out.

Logic duplication using **Maximum Fan-Out** assignments normally increases resource utilization, and can potentially increase compilation time, depending on the placement and the total resource usage within the selected device.

The improvement in timing performance that results from **Maximum Fan-Out** assignments is design-specific. This is because when you use the **Maximum Fan-Out** assignment, the Fitter duplicates the source logic to limit the fan-out, but does not control the destinations that each of the duplicated sources drive. Therefore, it is possible for duplicated source logic to be driving logic located all around the device. To avoid this situation, you can use the **Manual Logic Duplication** logic option.

If you are using **Maximum Fan-Out** assignments, benchmark your design with and without these assignments to evaluate whether they give the expected improvement in timing performance. Use the assignments only when you get improved results.



You can manually duplicate registers in the Intel Quartus Prime software regardless of the synthesis tool used. To duplicate a register, apply the **Manual Logic Duplication** logic option to the register with the Assignment Editor.

*Note:* Some Fitter optimizations may cause a small violation to the **Maximum Fan-Out** assignments to improve timing.

#### 5.5.9.8.2. Automatic Register Duplication: Estimated Physical Proximity

The `DUPLICATE_REGISTER` assignment helps in leveraging estimated physical proximity information to guide the creation of duplicates and their fan-out assignments.

```
set_instance_assignment -name DUPLICATE_REGISTER -to <register_name>
<num_duplicates>
```

where,

- `register_name` is the register to duplicate. To create a register tree from a chain, create a unique assignment for each register in the chain. `DUPLICATE_REGISTER` assignments are processed in the appropriate order if they apply to registers that drive each other in a chain.
- `num_duplicates` is the number of duplicates of the register to create (including the original). If the original signal has  $M$  fan-out, the average fan-outs of the duplicates are  $M/N$  but any individual duplicate may have more or fewer, at the discretion of the algorithm.

The `DUPLICATE_REGISTER` assignment is processed during the Fitter stage. It is necessary to create the duplicates and assign fan-outs between the duplicates based on early estimates of physical proximity to maximize the amount of time spent optimizing the design post-duplication. However, this renders fine-grained assignment decisions imprecise. The `DUPLICATE_REGISTER` assignment is best used when the number of duplicates is small (under 100) and the groups created are coarse-grained enough to allow for flexibility during optimization after the duplicates are created.

The **Fitter Duplication Summary** panel of the Fit report details the `DUPLICATE_REGISTER` assignments picked up by Intel Quartus Prime Pro Edition. It also summarizes any registered signal with greater than 1000 fan-outs, as they could be reasonable candidates for `DUPLICATE_REGISTER` assignments in future.

- Important:*
- Setting `PHYSICAL_SYNTHESIS` to `OFF` disables `DUPLICATE_REGISTER`.
  - Unlike other physical synthesis optimizations, the `DUPLICATE_REGISTER` assignment does allow duplication of registers that feed asynchronous clears and registers having location assignments.
  - The `DUPLICATE_REGISTER` assignment does not process registers that have any of the following conditions:
    - Registers drive global signals or clock signals.
    - Registers have timing assignments or exceptions applied to them.
    - Registers have a `preserve` attribute or a `PRESERVE_REGISTER` assignment.
    - Registers are marked as `don't touch`.
    - Registers drive or are driven by other partitions.

### 5.5.9.8.3. Automatic Register Duplication: Hierarchical Proximity

Leveraging design hierarchy information to guide the creation of duplicates and their fan-out assignments is enabled by the `DUPLICATE_HIERARCHY_DEPTH` assignment.

```
set_instance_assignment -name DUPLICATE_HIERARCHY_DEPTH -to <register_name>  
<num_levels>
```

where,

- `register_name` is the last register in a chain that fans out to multiple hierarchies. To create a register tree, ensure that there are sufficient simple registers behind the node and those simple registers are automatically pulled into the tree.
- `num_levels` corresponds to the upper bound of the number of registers that exist in the chain to use for duplicating down the hierarchies.

The `DUPLICATE_HIERARCHY_DEPTH` assignment is processed during the Synthesis stage. It is common for high-fanout signals to go through a pipeline of registers and drive into a sub-hierarchy of modules. For example, a system-wide reset can be propagated over several clock cycles and driven into many modules across the design. In several scenarios, it is useful to take advantage of the structure of this sub-hierarchy to infer the structure of the register tree to be created, such that endpoints within similar hierarchies are assigned the same copy of the signal, and branches in the design hierarchy dictates where to place branches in the register tree.

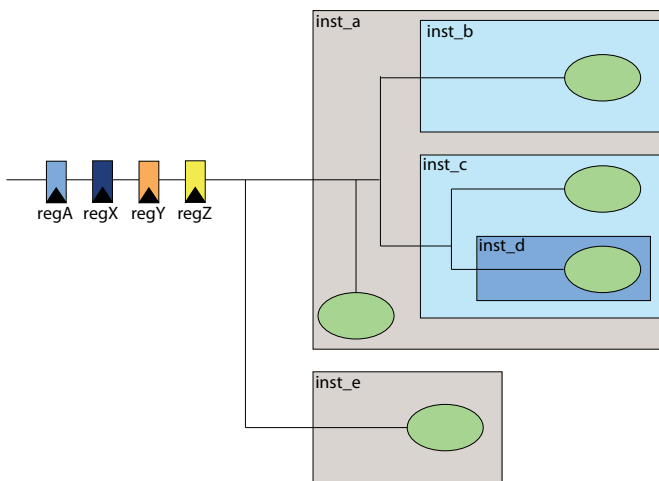
*Important:* The registers in the chain must satisfy all of the following conditions to be included in duplication:

- Registers must be fed only by another register.
- Registers must not be fed by a combinational logic.
- Registers must not be part of a synchronizer chain.
- Registers must not have any secondary signals.
- Registers must not have a `preserve` attribute or a `PRESERVE_REGISTER` assignment.
- All registers in the chain except the last one must have only one fan-out.

Consider the following example illustration of a netlist with a register chain and hierarchical organization of the endpoints it drives. The `DUPLICATE_HIERARCHY_DEPTH` assignment duplicates the pipeline registers across hierarchies, as shown in [Figure 48](#) on page 106.



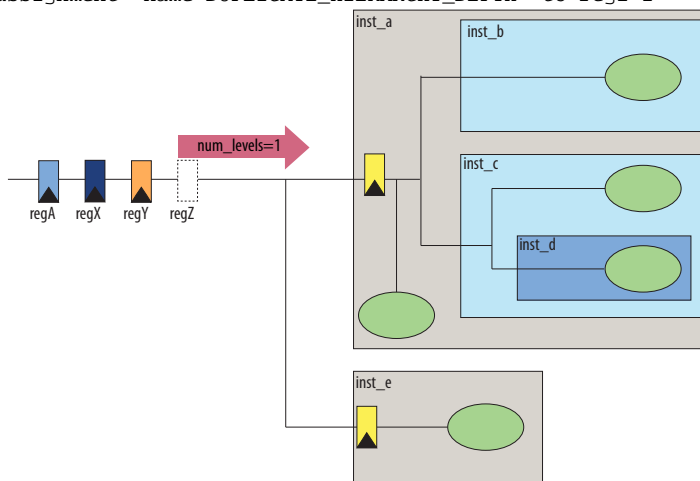
**Figure 45. Original Diagram Showing Four Pipeline Registers Connected to Multiple Hierarchies**



In this case, `regZ` is the appropriate assignment target as it is the endpoint in a chain of four registers. There is a maximum of three duplication candidates in this example (`regZ`, `regY`, and `regX`), so the assignment value can be anywhere between 1 and 3. `regA` is not pulled into the hierarchy to preserve the timing and optimization of paths that precede it. The `DUPLICATE_HIERARCHY_DEPTH` assignment is best used when a signal must be duplicated to more than 100 duplicates and the sub-hierarchy below the chain is deep and meaningful enough to guide the structure of the tree required.

**Figure 46. Netlist After Duplicating `regZ` to Hierarchy Level One**

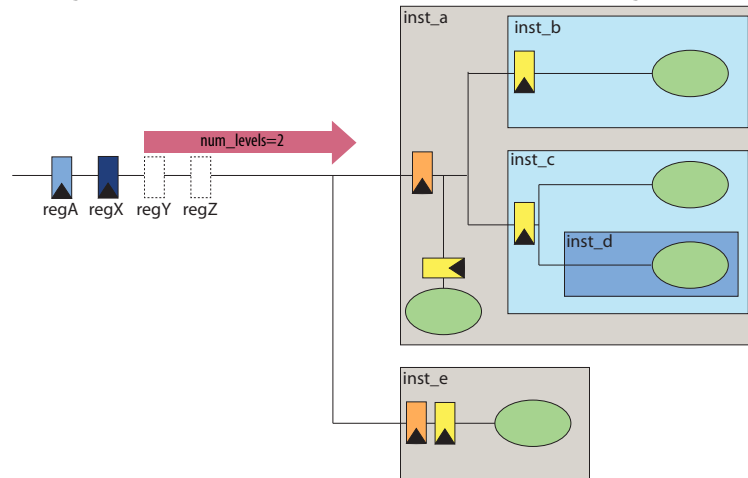
`set_instance_assignment -name DUPLICATE_HIERARCHY_DEPTH -to regZ 1`



When `num_levels` is set to 1, only `regZ` is pulled out of the chain and pushed down one hierarchy level into its fan-out tree.

**Figure 47. Netlist After Duplicating regZ to Hierarchy Level Two**

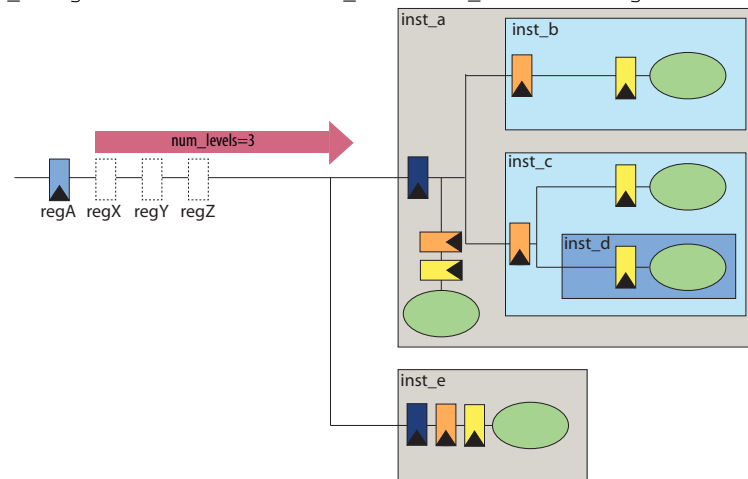
```
set_instance_assignment -name DUPLICATE_HIERARCHY_DEPTH -to regZ 2
```



When `num_levels` is set to 2, both `regY` and `regZ` are pulled out of the chain. `regZ` ends up at a maximum hierarchy depth two and `regY` ends up at hierarchy depth one.

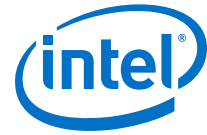
**Figure 48. Registers Duplicated Across Hierarchies**

```
set_instance_assignment -name DUPLICATE_HIERARCHY_DEPTH -to regZ 3
```



When `num_levels` is set to 3, all three registers (`regZ`, `regY` and `regZ`) are pulled out of the chain and pushed to a maximum hierarchy depth of three, two, and one levels, respectively.

The **Hierarchical Tree Duplication Summary** panel in the Synthesis report provides information on the registers specified by the `DUPLICATE_HIERARCHY_DEPTH` assignment. It also includes a reason for the chain length that can be used as a starting point for further improvements with the assignment. The Synthesis report also provides a panel named **Hierarchical Tree Duplication Details**, which provides information about the individual registers in the chain that can be used to better understand the structure of the implemented duplicates.



### 5.5.9.9. Prevent Shift Register Inference

Turning off the inference of shift registers can increase performance. This setting forces the software to use logic cells to implement the shift register, instead of using the ALTSHIFT\_TAPS IP core to implement the registers in memory block. If you implement shift registers in logic cells instead of memory, logic utilization increases.

### 5.5.9.10. Use Other Synthesis Options Available in Your Synthesis Tool

With your synthesis tool, experiment with the following options if they are available:

- Turn on register balancing or retiming
- Turn on register pipelining
- Turn off resource sharing

These options can increase performance, but typically increase the resource utilization of your design.

### 5.5.9.11. Fitter Seed

The Fitter seed affects the initial placement configuration of the design. Any change in the initial conditions changes the Fitter results; accordingly, each seed value results in a somewhat different fit. You can experiment with different seeds to attempt to obtain better fitting results and timing performance.

Changes in the design impact performance between compilations. This random variation is inherent in placement and routing algorithms—it is impossible to try all seeds and get the absolute best result.

*Note:*

Any design change that directly or indirectly affects the Fitter has the same type of random effect as changing the seed value. This includes any change in source files, **Compiler Settings** or **Timing Analyzer Settings**. The same effect can appear if you use a different computer processor type or different operating system, because different systems can change the way floating point numbers are calculated in the Fitter.

If a change in optimization settings marginally affects the register-to-register timing or number of failing paths, you cannot always be certain that your change caused the improvement or degradation, or whether it is due to random effects in the Fitter. If your design is still changing, running a seed sweep (compiling your design with multiple seeds) determines whether the average result improved after an optimization change, and whether a setting that increases compilation time has benefits worth the increased time, such as with physical synthesis settings. The sweep also shows the amount of random variation to expect for your design.

If your design is finalized you can compile your design with different seeds to obtain one optimal result. However, if you subsequently make any changes to your design, you might need to perform seed sweep again.

Click **Assignments** ► **Compiler Settings** to control the initial placement with the seed. You can use the DSE II to perform a seed sweep easily.

To specify a Fitter seed use the following Tcl command :

```
set_global_assignment -name SEED <value>
```

### Related Information

[Optimize Settings with Design Space Explorer II](#) on page 91

#### 5.5.9.12. Set Maximum Router Timing Optimization Level

To improve routability in designs where the router did not pick up the optimal routing lines, set the **Router Timing Optimization Level** to **Maximum**. This setting determines how aggressively the router tries to meet the timing requirements. Setting this option to **Maximum** can marginally increase design speed at the cost of increased compilation time. Setting this option to **Minimum** can reduce compilation time at the cost of marginally reduced design speed. The default value is **Normal**.

### Related Information

[Router Timing Optimization Level Logic Option](#)  
In *Intel Quartus Prime Help*

#### 5.5.9.13. Register-to-Register Timing Analysis

Your design meets timing requirements when you do not have negative slack on any register-to-register path on any of the clock domains. When timing requirements are not met, a report on the failed paths can uncover more detail.

##### 5.5.9.13.1. Tips for Analyzing Failing Paths

When you are analyzing failing paths, examine the reports and waveforms to determine if the correct constraints are being applied, and add timing exceptions as appropriate. A multicycle constraint relaxes setup or hold relationships by the specified number of clock cycles. A false path constraint specifies paths that can be ignored during timing analysis. Both constraints allow the Fitter to work harder on affected paths.

- Focus on improving the paths that show the worst slack. The Fitter works hardest on paths with the worst slack. If you fix these paths, the Fitter might be able to improve the other failing timing paths in the design.
- Check for nodes that appear in many failing paths. These nodes are at the top of the list in a timing report panel, along with their minimum slacks. Look for paths that have common source registers, destination registers, or common intermediate combinational nodes. In some cases, the registers are not identical, but are part of the same bus.
- In the timing analysis report panels, click the **From** or **To** column headings to sort the paths by source or destination registers. If you see common nodes, these nodes indicate areas of your design that might be improved through source code changes or Intel Quartus Prime optimization settings. Constraining the placement for just one of the paths might decrease the timing performance for other paths by moving the common node further away in the device.

### Related Information

- [Exploring Paths in the Chip Planner](#) on page 131
- [Design Evaluation for Timing Closure](#) on page 58
- [Review Timing Path Details](#) on page 69



### 5.5.9.13.2. Tips for Analyzing Failing Clock Paths that Cross Clock Domains

When analyzing clock path failures:

- Check whether these paths cross two clock domains. In paths that cross two clock domains, the **From Clock** and **To Clock** in the timing analysis report are different.

**Figure 49. Different Value in From Clock and To Clock Field**

Setup Transfers						
	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	clkin	clkin	21	0	0	0
2	clkin	clkout	false path	0	0	0
3	clkout	clkout	31	0	0	0

- Check if the design contains paths that involve a different clock in the middle of the path, even if the source and destination register clock are the same.
- Check whether failing paths between these clock domains need to be analyzed synchronously. Set failing paths that are not to be analyzed synchronously as false paths.
- When you run `report_timing` on a design, the report shows the launch clock and latch clock for each failing path. Check whether the relationship between the launch clock and latch clock is realistic and what you expect from your knowledge of the design. For example, the path can start at a rising edge and end at a falling edge, which reduces the setup relationship by one half clock cycle.
- Review the clock skew that appears in the Timing Report. A large skew may indicate a problem in the design, such as a gated clock, or a problem in the physical layout (for example, a clock using local routing instead of dedicated clock routing). When you have made sure the paths are analyzed synchronously and that there is no large skew on the path, and that the constraints are correct, you can analyze the data path. These steps help you fine tune your constraints for paths across clock domains to ensure you get an accurate timing report.
- Check if the PLL phase shift is reducing the setup requirement. You might adjust this by using PLL parameters and settings.
- Ignore paths that cross clock domains for logic protected with synchronization logic (for example, FIFOs or double-data synchronization registers), even if the clocks are related.
- Set false path constraints on all unnecessary paths. Attempting to optimize unnecessary paths can prevent the Fitter from meeting the timing requirements on timing paths that are critical to the design.

#### Related Information

[Report CDC Viewer](#) on page 83

### 5.5.9.13.3. Tips for Critical Path Analysis

When analyzing the failing paths in a design, it is helpful to understand the interactions around the critical paths.

To understand what may be pulling on a critical path, the following `report_timing` command can be useful.

1. In the project directory, run the `report_timing` command to find the nodes in a critical path.
2. Copy the code below in a `.tcl` file, and replace the first two variable with the node names from the **From Node** and **To Node** columns of the worst path. The script analyzes the path between the worst source and destination registers.

```

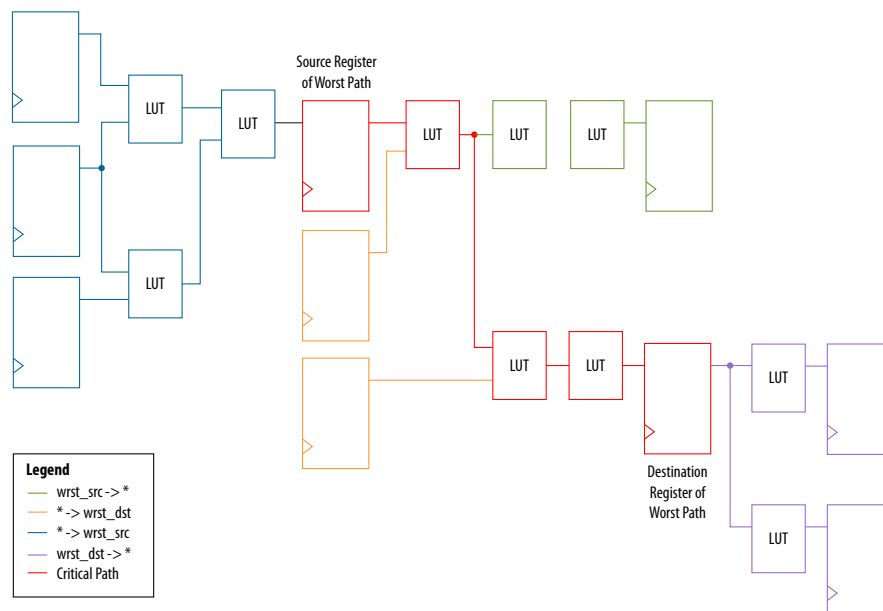
set wrst_src <insert_source_of_worst_path_here>
set wrst_dst <insert_destination_of_worst_path_here>
report_timing -setup -npaths 50 -detail path_only -from $wrst_src \
-panel_name "Worst Path||wrst_src -> *"
report_timing -setup -npaths 50 -detail path_only -to $wrst_dst \
-panel_name "Worst Path||* -> wrst_dst"
report_timing -setup -npaths 50 -detail path_only -to $wrst_src \
-panel_name "Worst Path||* -> wrst_src"
report_timing -setup -npaths 50 -detail path_only -from $wrst_dst \
-panel_name "Worst Path||wrst_dst -> *"

```

3. From the **Script** menu, source the `.tcl` file.
4. In the resulting timing panel, locate timing failed paths (highlighted in red) in the Chip Planner, and view information such as distance between the nodes and large fan-outs.

The figure shows a simplified example of what these reports analyzed.

**Figure 50. Timing Report**





The critical path of the design is in red. The relation between the .tcl script and the figure is:

- The first two lines show everything inside the two endpoints of the critical path that are pulling them in different directions.
  - The first `report_timing` command analyzes all paths the source is driving, shown in green.
  - The second `report_timing` command analyzes all paths going to the destination, including the critical path, shown in orange.
- The last two `report_timing` commands show everything outside of the endpoints pulling them in other directions.

If any of these neighboring paths have slacks near the critical path, the Fitter is balancing these paths with the critical path, trying to achieve the best slack.

### Related Information

[Review Timing Path Details](#) on page 69

#### 5.5.9.13.4. Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles

Many designs have the same critical paths show up after each compile. In other designs, critical paths bounce around between different hierarchies, changing with each compile.

This behavior happens in high speed designs where many register-to-register paths have very little slack. Different placements can then result in timing failures in the marginal paths.

1. In the project directory, create a script named `TQ_critical_paths.tcl`.
2. After compilation, review the critical paths and then write a generic `report_timing` command to capture those paths.

For example, if several paths fail in a low-level hierarchy, add a command such as:

```
report_timing -setup -npaths 50 -detail path_only \
  -to "main_system: main_system_inst|app_cpu:cpu|*" \
  -panel_name "Critical Paths||s: * -> app_cpu"
```

3. If there is a specific path, such as a bit of a state-machine going to other `*count_sync*` registers, you can add a command similar to:

```
report_timing -setup -npaths 50 -detail path_only \
  -from "main_system: main_system_inst|egress_count_sm:egress_inst|
update" \
  -to "*count_sync*" -panel_name "Critical Paths||s: egress_sm|update
-> count_sync"
```

4. Execute this script in the Timing Analyzer after every compilation, and add new `report_timing` commands as new critical paths appear.

This helps you monitor paths that consistently fail and paths that are only marginal, so you can prioritize effectively

#### 5.5.9.13.5. Global Routing Resources

Global routing resources are designed to distribute high fan-out, low-skew signals (such as clocks) without consuming regular routing resources. Depending on the device, these resources can span the entire chip or a smaller portion, such as a

quadrant. The Intel Quartus Prime software attempts to assign signals to global routing resources automatically, but you might be able to make more suitable assignments manually.

For details about the number and types of global routing resources available, refer to the relevant device handbook.

Check the global signal utilization in your design to ensure that the appropriate signals have been placed on the global routing resources. In the Compilation Report, open the Fitter report and click **Resource Section**. Analyze the Global & Other Fast Signals and Non-Global High Fan-out Signals reports to determine whether any changes are required.

You might be able to reduce skew for high fan-out signals by placing them on global routing resources. Conversely, you can reduce the insertion delay of low fan-out signals by removing them from global routing resources. Doing so can improve clock enable timing and control signal recovery/removal timing, but increases clock skew. Use the **Global Signal** setting in the Assignment Editor to control global routing resources.

### 5.5.10. Metastability Analysis and Optimization Techniques

Metastability problems can occur when a signal is transferred between circuitry in unrelated or asynchronous clock domains, because the designer cannot guarantee that the signal meets its setup and hold time requirements. The mean time between failures (MTBF) is an estimate of the average time between instances when metastability could cause a design failure.

You can use the Intel Quartus Prime software to analyze the average MTBF due to metastability when a design synchronizes asynchronous signals and to optimize the design to improve the MTBF. These metastability features are supported only for designs constrained with the Timing Analyzer, and for select device families.

If the MTBF of your design is low, refer to the Metastability Optimization section in the Timing Optimization Advisor, which suggests various settings that can help optimize your design in terms of metastability.

#### Related Information

- [Understanding Metastability in FPGAs](#)
- [Managing Metastability with the Intel Quartus Prime Software](#)  
In *Intel Quartus Prime Pro Edition User Guide: Design Recommendations*

## 5.6. Periphery to Core Register Placement and Routing Optimization

The Periphery to Core Register Placement and Routing Optimization (P2C) option specifies whether the Fitter performs targeted placement and routing optimization on direct connections between periphery logic and registers in the FPGA core. P2C is an optional pre-routing-aware placement optimization stage that enables you to more reliably achieve timing closure.

*Note:* The **Periphery to Core Register Placement and Routing Optimization** option applies in both directions, periphery to core and core to periphery.



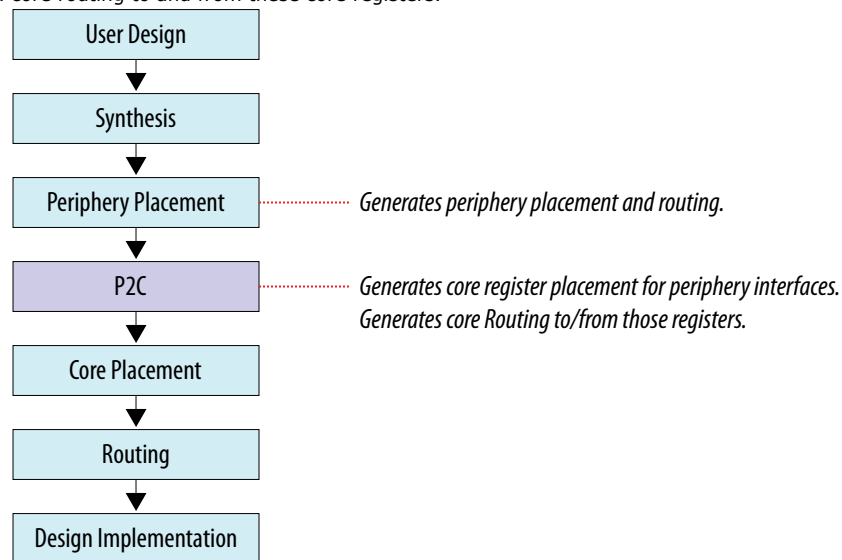


Transfers between external interfaces (for example, high-speed I/O or serial interfaces) and the FPGA often require routing many connections with tight setup and hold timing requirements. When this option is turned on, the Fitter performs P2C placement and routing decisions before those for core placement and routing. This reserves the necessary resources to ensure that your design achieves its timing requirements and avoids routing congestion for transfers with external interfaces.

This option is available as a global assignment, or can be applied to specific instances within your design.

### Figure 51. Periphery to Core Register Placement and Routing Optimization (P2C) Flow

P2C runs after periphery placement, and generates placement for core registers on corresponding P2C/C2P paths, and core routing to and from these core registers.



[Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box](#) on page 113

[Setting Periphery to Core Optimizations in the Assignment Editor](#) on page 114

[Viewing Periphery to Core Optimizations in the Fitter Report](#) on page 114

## 5.6.1. Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box

The **Periphery to Core Placement and Routing Optimization** setting specifies whether the Fitter optimizes targeted placement and routing on direct connections between periphery logic and registers in the FPGA core.

You can optionally perform periphery to core optimizations by instance with settings in the Assignment Editor.

1. In the Intel Quartus Prime software, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.
2. In the **Advanced Fitter Settings** dialog box, for the **Periphery to Core Placement and Routing Optimization** option, select one of the following options depending on how you want to direct periphery to core optimizations in your design:

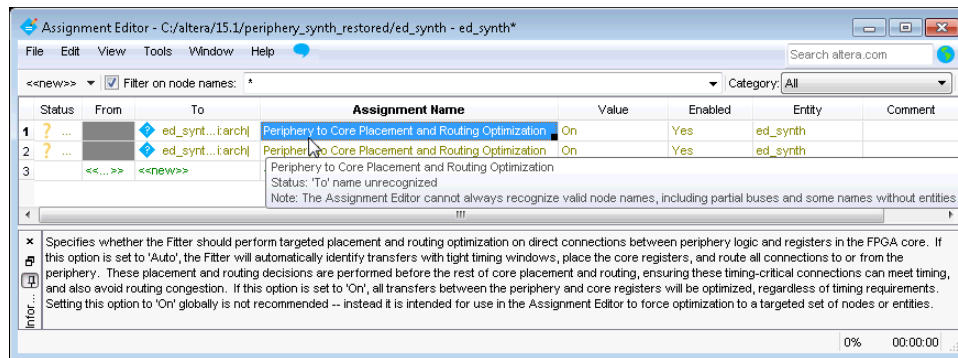
- a. Select **Auto** to direct the software to automatically identify transfers with tight timing windows, place the core registers, and route all connections to or from the periphery.
- b. Select **On** to direct the software to globally optimize all transfers between the periphery and core registers, regardless of timing requirements.  
*Note:* Setting this option to **On** in the **Advanced Fitter Settings** is not recommended. The intended use for this setting is in the Assignment Editor to force optimization for a targeted set of nodes or instance.
- c. Select **Off** to disable periphery to core path optimization in your design.

### 5.6.2. Setting Periphery to Core Optimizations in the Assignment Editor

When you turn on the **Periphery to Core Placement and Routing Optimization (P2C/C2P)** setting in the Assignment Editor, the Intel Quartus Prime software performs periphery to core, or core to periphery optimizations on selected instances in your design.

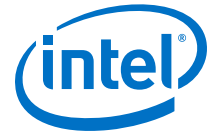
You can optionally perform periphery to core optimizations by instance with settings in the **Advanced Fitter Settings** dialog box.

1. In the Intel Quartus Prime software, click **Assignments > Assignment Editor**.
2. For the selected path, double-click the **Assignment Name** column, and then click the **Periphery to core register placement and routing optimization** option in the drop-down list.
3. In the **To** column, choose either a periphery node or core register node on a P2C/C2P path you want to optimize. Leave the **From** column empty. For paths to appear in the Assignments Editor, you must first run Analysis & Synthesis on your design.



### 5.6.3. Viewing Periphery to Core Optimizations in the Fitter Report

The Intel Quartus Prime software generates a periphery to core placement and routing optimization summary in the **Fitter (Place & Route)** report after compilation.



## 5. Timing Closure and Optimization

UG-20133 | 2020.09.28

1. Compile your Intel Quartus Prime project.
2. In the **Tasks** pane, select **Compilation**.
3. Under **Fitter (Place & Route)**, double-click **View Report**.
4. In the **Fitter** folder, expand the **Place Stage** folder.
5. Double-click **Periphery to Core Transfer Optimization Summary**.

**Table 28. Fitter Report - Periphery to Core Transfer Optimization (P2C) Summary**

From Path	To Path	Status
Node 1	Node 2	<b>Placed and Routed</b> —Core register is locked. Periphery to core/core to periphery routing is committed.
Node 3	Node 4	<b>Placed but not Routed</b> —Core register is locked. Routing is not committed. This occurs when P2C is not able to optimize all targeted paths within a single group, for example, the same delay/wire requirement, or the same control signals. Partial P2C routing commitments may cause unresolvable routing congestion.
Node 5	Node 6	<b>Not Optimized</b> —This occurs when P2C is set to <b>Auto</b> and the path is not optimized due to one of the following issues: <ol style="list-style-type: none"> <li>a. The delay requirement is impossible to achieve.</li> <li>b. The minimum delay requirement (for hold timing) is too large. The P2C algorithm cannot efficiently handle cases when many wires need to be added to meet hold timing.</li> <li>c. P2C encountered unresolvable routing congestion for this particular path.</li> </ol>

Periphery to Core Transfer Optimization Summary			
	From	To	Status
1	Periphery to Core Transfer		
2	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[1]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instsingle_port_af1_rdata_valid_regs[0]	Placed but Not Routed
3	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[2]lane_gen[2]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[63]	Placed but Not Routed
4	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[1]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[11]	Placed but Not Routed
5	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[7]	Placed but Not Routed
6	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[3]	Placed but Not Routed
7	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[0]	Placed but Not Routed
8	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[79]	Placed but Not Routed
9	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[75]	Placed but Not Routed
10	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[0]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[73]	Placed but Not Routed
11	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[2]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[71]	Placed but Not Routed
12	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[1]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[69]	Placed but Not Routed
13	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[1]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[35]	Placed but Not Routed
14	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[31]	Placed but Not Routed
15	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[3]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[27]	Placed but Not Routed
16	dutjarchjarch_instjio_tiles_wrap_instjio_tiles_insttile_gen[0]lane_gen[2]lane_inst	dutjarchjarch_instjafi_if_instjmem_sp_bidir_data.mem_dq.af1_regs_isr_out[23]	Placed but Not Routed

## 5.7. Scripting Support

You can run procedures and make settings described in this manual in a Tcl script. You can also run procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> -to <instance name>
```

**Note:** If the <value> field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

#### Related Information

- [Intel Quartus Prime Pro Edition Settings File Reference Manual](#)  
For information about all settings and constraints in the Intel Quartus Prime software.
- [Tcl Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*
- [Command Line Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*

### 5.7.1. Initial Compilation Settings

Use the Intel Quartus Prime Settings File (.qsf) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

The top table lists the .qsf variable name and applicable values for the settings described in the *Initial Compilation: Required Settings* section in the *Design Optimization Overview* chapter. The bottom table lists the advanced compilation settings.

**Table 29. Initial Compilation Settings**

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING	ON, OFF	Global
Optimize Hold Timing	OPTIMIZE_HOLD_TIMING	OFF, IO PATHS AND MINIMUM TPD PATHS, ALL PATHS	Global

**Table 30. Advanced Compilation Settings**

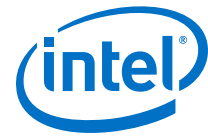
Setting Name	.qsf File Variable Name	Values	Type
Router Timing Optimization level	ROUTER_TIMING_OPTIMIZATION_LEVEL	NORMAL, MINIMUM, MAXIMUM	Global

#### Related Information

[Design Optimization Overview](#) on page 6

### 5.7.2. I/O Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the I/O timing optimization settings.

**Table 31. I/O Timing Optimization Settings**

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING	ON, OFF	Global
Fast Input Register	FAST_INPUT_REGISTER	ON, OFF	Instance
Fast Output Register	FAST_OUTPUT_REGISTER	ON, OFF	Instance
Fast Output Enable Register	FAST_OUTPUT_ENABLE_REGISTER	ON, OFF	Instance
Fast OCT Register	FAST_OCT_REGISTER	ON, OFF	Instance

### 5.7.3. Register-to-Register Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the settings described in *Register-to-Register Timing Optimization Techniques*.

**Table 32. Register-to-Register Timing Optimization Settings**

Setting Name	.qsf File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Fitter Seed	SEED	<integer>	Global
Maximum Fan-Out	MAX_FANOUT	<integer>	Instance
Manual Logic Duplication	DUPLICATE_ATOM	<node name>	Instance
Optimize Power during Synthesis	OPTIMIZE_POWER_DURING_SYNTHESIS	NORMAL, OFF EXTRA_EFFORT	Global
Optimize Power during Fitting	OPTIMIZE_POWER_DURING_FITTING	NORMAL, OFF EXTRA_EFFORT	Global

## 5.8. Timing Closure and Optimization Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2020.09.28	20.3	<ul style="list-style-type: none"> <li>Added "Back-Annotate Optimized Assignments" topic to describe new GUI support for back-annotation of pin, RAM, DSP, and clock assignments.</li> <li>Added "Correct Design Assistant Rule Violations" topic.</li> <li>Updated "Report Timing" topic for Extra Info tab data.</li> <li>Added new "Report Logic Depth," "Report Neighbor Paths," "Report Register Spread," "Report Route Net of Interest," "Report Hierarchical Retiming Restrictions," and "Report Pipelining Information," topics to "Review Details of Timing Paths" section.</li> <li>Moved "Optimize Settings with Design Space Explorer II" to "Design Evaluation for Timing Closure" section and updated links to Help.</li> <li>Retitled "Intel Stratix 10 Timing Closure Recommendations" topic to "Implement Fast Forward Timing Closure Recommendations".</li> </ul>
2019.07.01	19.1	Added important notes to <i>Automatic Register Duplication: Estimated Physical Proximity</i> and <i>Automatic Register Duplication: Hierarchical Proximity</i> topics.
		<b>continued...</b>

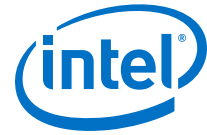


Document Version	Intel Quartus Prime Version	Changes
2019.04.01	19.1	<ul style="list-style-type: none"> <li>Added more information about register duplication methods in <i>Duplicate Logic for Fan-out Control</i> topic.</li> <li>Moved content related to manual register duplication from <i>Duplicate Logic for Fan-out Control</i> topic to a newly created sub-topic <i>Manually Adding Duplicate Registers</i>.</li> <li>Added <i>Automatic Register Duplication: Estimated Physical Proximity</i> and <i>Automatic Register Duplication: Hierarchical Proximity</i> as new sub-topics under <i>Duplicate Logic for Fan-out Control</i> to describe automatic register duplication process.</li> </ul>
2018.11.12	18.1.0	<ul style="list-style-type: none"> <li>Updated "Placement Effort Multiplier" figure and text descriptions in "Adjust Placement Effort" topic.</li> <li>Updated "Fitter Effort" figure and text descriptions in "Adjust Fitter Effort" topic.</li> <li>Updated "Optimize Hold Timing Option" screenshot in "Wires Added for Hold" topic.</li> </ul>
2018.09.24	18.1.0	<ul style="list-style-type: none"> <li>Removed duplicated topic: <i>Resource Utilization Optimization Techniques</i>. The topic is now in the <i>Area Optimization</i> chapter.</li> <li>Removed reference to unsupported CARRY and CASCADE buffers from "Optimize IOC Register Placement for Timing Logic Option" topic.</li> </ul>
2017.11.06	17.1.0	<ul style="list-style-type: none"> <li>Added support for Intel Stratix 10 Hyper-Retiming, Fast Forward compilation, and Fast Forward Viewer.               <ul style="list-style-type: none"> <li>Added topics: Critical Chains, Viewing Critical Chains, Intel Stratix 10 Timing Closure Recommendations, Retiming Limit Details Report, Using the Retiming Limit Details Report, Fast Forward Timing Closure Recommendations, Generating Fast Forward Timing Closure Recommendations, Implementing Fast Forward Recommendations.</li> </ul> </li> <li>Added topic about using partitions to achieve timing closure.</li> <li>Moved Topic: Design Evaluation for Timing Closure after Initial Compilation: Optional Fitter Settings.</li> <li>Removed statement about applying physical synthesis optimizations in a portion of a design.</li> <li>Removed references to optimizing hold timing for selected paths.</li> <li>Updated logic options about resource utilization optimization settings.</li> </ul>
2017.05.08	17.0.0	<ul style="list-style-type: none"> <li>Added topic: <i>Critical Paths</i>.</li> <li>Updated <i>Register-to-Register Timing</i> and renamed to <i>Register-to-Register Timing Analysis</i>.</li> <li>Renamed topic: <i>Timing Analysis with the Timing Analyzer</i> to <i>Displaying Path Reports with the Timing Analyzer</i>.</li> <li>Removed (LUT-Based Devices) remark from topic titles.</li> <li>Renamed topic: <i>Optimizing Timing (LUT-Based Devices)</i> to <i>Timing Optimization</i>.</li> <li>Renamed topic: <i>Debugging Timing Failures in the Timing Analyzer</i> to <i>Displaying Timing Closure Recommendations for Failing Paths</i>.</li> <li>Renamed topic: <i>Improving Register-to-Register Timing Summary</i> to <i>Improving Register-to-Register Timing</i>.</li> <li>Removed topics: <i>Tips for Locating Multiple Paths to the Chip Planner</i>, <i>LogicLock Assignments</i> and <i>Hierarchy Assignments</i>.</li> <li>Removed reference to deprecated Fitter Effort Logic Option.</li> <li>Removed information about Pin Advisor and Resource Optimization Advisor.</li> <li>Removed figure: Clock Regions</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>Implemented Intel rebranding.</li> </ul>
2016.05.02	16.0.0	<ul style="list-style-type: none"> <li>Removed information about deprecated physical synthesis options.</li> <li>Added information about monitoring clustering difficulty.</li> </ul>

**continued...**

## 5. Timing Closure and Optimization

UG-20133 | 2020.09.28



Document Version	Intel Quartus Prime Version	Changes
2015.11.02	15.1.0	<ul style="list-style-type: none"> <li>Added: <i>Periphery to Core Register Placement and Routing Optimization</i>.</li> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
2014.12.15	14.1.0	<ul style="list-style-type: none"> <li>Updated location of Fitter Settings, Analysis &amp; Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.</li> <li>Updated DSE II content.</li> </ul>
June 2014	14.0.0	<ul style="list-style-type: none"> <li>Dita conversion.</li> <li>Removed content about obsolete devices that are no longer supported in QII software v14.0: Arria GX, Arria II, Cyclone III, Stratix II, Stratix III.</li> <li>Replaced Megafunction content with IP core content.</li> </ul>
November 2013	13.1.0	<ul style="list-style-type: none"> <li>Added Design Evaluation for Timing Closure section.</li> <li>Removed Optimizing Timing (Macrocell-Based CPLDs) section.</li> <li>Updated Optimize Multi-Corner Timing and Fitter Aggressive Routability Optimization.</li> <li>Updated Timing Analysis with the Timing Analyzer to show how to access the <b>Report All Summaries</b> command.</li> <li>Updated Ignored Timing Constraints to include a help link to <i>Fitter Summary Reports</i> with the <b>Ignored Assignment Report</b> information.</li> </ul>
May 2013	13.0.0	<ul style="list-style-type: none"> <li>Renamed chapter title from Area and Timing Optimization to Timing Closure and Optimization.</li> <li>Removed design and area/resources optimization information.</li> <li>Added the following sections: <ul style="list-style-type: none"> <li>Fitter Aggressive Routability Optimization.</li> <li>Tips for Analyzing Paths from/to the Source and Destination of Critical Path.</li> <li>Tips for Locating Multiple Paths to the Chip Planner.</li> <li>Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles.</li> </ul> </li> </ul>
November 2012	12.1.0	<ul style="list-style-type: none"> <li>Updated "Initial Compilation: Optional Fitter Settings" on page 13-2, "I/O Assignments" on page 13-2, "Initial Compilation: Optional Fitter Settings" on page 13-2, "Resource Utilization" on page 13-9, "Routing" on page 13-21, and "Resolving Resource Utilization Problems" on page 13-43.</li> </ul>
June 2012	12.0.0	<ul style="list-style-type: none"> <li>Updated "Optimize Multi-Corner Timing" on page 13-6, "Resource Utilization" on page 13-10, "Timing Analysis with the Timing Analyzer" on page 13-12, "Using the Resource Optimization Advisor" on page 13-15, "Increase Placement Effort Multiplier" on page 13-22, "Increase Router Effort Multiplier" on page 13-22 and "Debugging Timing Failures in the Timing Analyzer" on page 13-24.</li> <li>Minor text edits throughout the chapter.</li> </ul>
November 2011	11.1.0	<ul style="list-style-type: none"> <li>Updated the "Timing Requirement Settings", "Standard Fit", "Fast Fit", "Optimize Multi-Corner Timing", "Timing Analysis with the Timing Analyzer", "Debugging Timing Failures in the Timing Analyzer", "LogicLock Assignments", "Tips for Analyzing Failing Clock Paths that Cross Clock Domains", "Flatten the Hierarchy During Synthesis", "Fast Input, Output, and Output Enable Registers", and "Hierarchy Assignments" sections</li> <li>Updated Table 13-6</li> <li>Added the "Spine Clock Limitations" section</li> <li>Removed the Change State Machine Encoding section from page 19</li> <li>Removed Figure 13-5</li> <li>Minor text edits throughout the chapter</li> </ul>

*continued...*



Document Version	Intel Quartus Prime Version	Changes
May 2011	11.0.0	<ul style="list-style-type: none"><li>• Reorganized sections in "Initial Compilation: Optional Fitter Settings" section</li><li>• Added new information to "Resource Utilization" section</li><li>• Added new information to "Duplicate Logic for Fan-Out Control" section</li><li>• Added links to Help</li><li>• Additional edits and updates throughout chapter</li></ul>
December 2010	10.1.0	<ul style="list-style-type: none"><li>• Added links to Help</li><li>• Updated device support</li><li>• Added "Debugging Timing Failures in the Timing Analyzer" section</li><li>• Removed Classic Timing Analyzer references</li><li>• Other updates throughout chapter</li></ul>
August 2010	10.0.1	Corrected link
July 2010	10.0.0	<ul style="list-style-type: none"><li>• Moved Compilation Time Optimization Techniques section to new <i>Reducing Compilation Time</i> chapter</li><li>• Removed references to Timing Closure Floorplan</li><li>• Moved Smart Compilation Setting and Early Timing Estimation sections to new <i>Reducing Compilation Time</i> chapter</li><li>• Added Other Optimization Resources section</li><li>• Removed outdated information</li><li>• Changed references to DSE chapter to Help links</li><li>• Linked to Help where appropriate</li><li>• Removed Referenced Documents section</li></ul>



## 6. Analyzing and Optimizing the Design Floorplan

---

As FPGA designs grow larger in density, the ability to analyze the design for performance, routing congestion, and logic placement is critical to meet the design requirements. This chapter discusses how the Chip Planner and Logic Lock regions help you improve your design's floorplan.

Design floorplan analysis helps to close timing, and ensures optimal performance in highly complex designs. With analysis capability, the Intel Quartus Prime Chip Planner helps you close timing quickly on your designs. You can use the Chip Planner together with Logic Lock regions to compile your designs hierarchically and assist with floorplanning. Additionally, use partitions to preserve placement and routing results from individual compilation runs.

You can perform design analysis, as well as create and optimize the design floorplan with the Chip Planner. To make I/O assignments, use the Pin Planner.

**Note:** As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.

For information about the Early Place Flow, refer to the *Intel Quartus Prime Pro Edition User Guide: Compiler*.

For information about floorplanning a Partial Reconfiguration design, refer to the *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*.

### Related Information

- [Early Place Flow](#)  
In *Intel Quartus Prime Pro Edition User Guide: Compiler*
- [Floorplanning a Partial Reconfiguration Design](#)  
In *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*
- [Managing Device I/O Pins](#)

### 6.1. Design Floorplan Analysis in the Chip Planner


The Chip Planner simplifies floorplan analysis by providing visual display of chip resources. With the Chip Planner, you can view post-compilation placement, connections, and routing paths. You can also make assignment changes, such as creating and deleting resource assignments.

The Chip Planner showcases:

- Logic Lock regions
- Relative resource usage
- Detailed routing information
- Fan-in and fan-out connections between nodes
- Timing paths between registers
- Delay estimates for paths
- Routing congestion information

### 6.1.1. Starting the Chip Planner

To start the Chip Planner, select **Tools** ► **Chip Planner**. You can also start the Chip Planner by the following methods:

- Click the Chip Planner icon  on the Intel Quartus Prime software toolbar.
- In the following tools, right-click any chip resource and select **Locate** ► **Locate in Chip Planner**:
  - Compilation Report
  - **Logic Lock Regions Window**
  - Technology Map Viewer
  - **Project Navigator** window
  - Node Finder
  - Simulation Report
  - Report Timing panel of the Timing Analyzer

### 6.1.2. Chip Planner GUI Components

#### 6.1.2.1. Chip Planner Toolbar

The Chip Planner toolbar provides powerful tools for visual design analysis. You can access Chip Planner commands either from the **View** menu, or by clicking the icons in the toolbars.

#### 6.1.2.2. Layers Settings

The Chip Planner allows you to control the display of resources.

##### Layers Settings Pane

With the **Layers Settings** pane, you can manage the graphic elements that the Chip Planner displays.

You open the **Layers Settings** pane by clicking **View** ► **Layers Settings**. The **Layers Settings** pane offers layer presets, which group resources that are often used together. The **Basic**, **Detailed**, and **Floorplan Editing** default presets are useful for general assignment-related activities. You can also create custom presets tailored to your needs.

##### Related Information

- [Viewing Architecture-Specific Design Information](#) on page 124



- [Layers Settings Dialog Box](#)  
In *Intel Quartus Prime Help*

### 6.1.2.3. Locate History Window

As you optimize your design floorplan, you might have to locate a path or node in the Chip Planner more than once. The **Locate History** window lists all the nodes and paths you have displayed using a **Locate in Chip Planner** command, providing easy access to the nodes and paths of interest to you.

If you locate a required path from the **Timing Analyzer Report Timing** pane, the **Locate History** window displays the required clock path. If you locate an arrival path from the **Timing Analyzer Report Timing** pane, the **Locate History** window displays the path from the arrival clock to the arrival data. Double-clicking a node or path in the **Locate History** window displays the selected node or path in the Chip Planner.

### 6.1.2.4. Chip Planner Floorplan Views

The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the targeted Intel device. As you zoom in, the level of abstraction decreases, revealing more details about your design.

#### Bird's Eye View

The Bird's Eye View displays a high-level picture of resource usage for the entire chip and provides a fast and efficient way to navigate between areas of interest in the Chip Planner.

The Bird's Eye View is particularly useful when the parts of your design that you want to view are at opposite ends of the chip, and you want to quickly navigate between resource elements without losing your frame of reference.

#### Properties Window

The **Properties** window displays detailed properties of the objects (such as atoms, paths, Logic Lock regions, or routing elements) currently selected in the Chip Planner. To display the **Properties** window, right-click the object and select **View > Properties**.

#### Related Information

[Bird's Eye View Window](#)  
In *Intel Quartus Prime Help*

### 6.1.3. Viewing Design Elements in the Chip Planner

The Chip Planner allows you to locate and report details on various elements of your design, such as viewing available clock networks, routing congestion, I/O banks, and high-speed serial interfaces in the floorplan.

The following section described how to view various design elements in the Chip Planner.

### 6.1.3.1. Viewing Architecture-Specific Design Information

The Chip Planner allows you to view architecture-specific information related to your design. By enabling the options in the **Layers Settings** pane, you can view:

- **Device routing resources used by your design**—View how blocks are connected, as well as the signal routing that connects the blocks.
- **LE configuration**—View logic element (LE) configuration in your design. For example, you can view which LE inputs are used; whether the LE utilizes the register, the look-up table (LUT), or both; as well as the signal flow through the LE.
- **ALM configuration**—View ALM configuration in your design. For example, you can view which ALM inputs are used; whether the ALM utilizes the registers, the upper LUT, the lower LUT, or all of them. You can also view the signal flow through the ALM.
- **I/O configuration**—View device I/O resource usage. For example, you can view which components of the I/O resources are used, whether the delay chain settings are enabled, which I/O standards are set, and the signal flow through the I/O.
- **PLL configuration**—View phase-locked loop (PLL) configuration in your design. For example, you can view which control signals of the PLL are used with the settings for your PLL.
- **Timing**—View the delay between the inputs and outputs of FPGA elements. For example, you can analyze the timing of the `DATAB` input to the `COMBOUT` output.

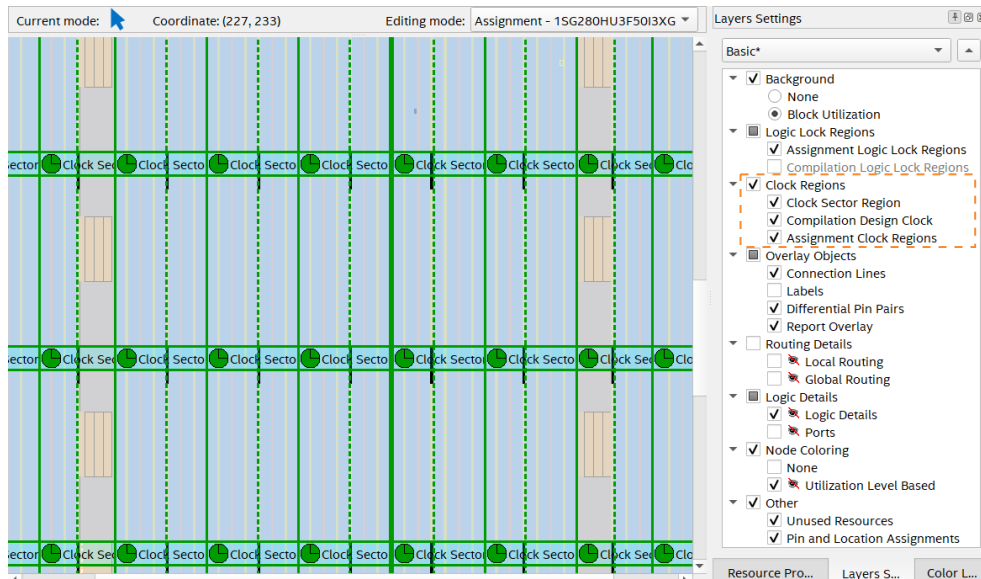
#### Related Information

- [Layers Settings](#) on page 122
- [Layers Settings Dialog Box](#)  
In *Intel Quartus Prime Help*

### 6.1.3.2. Viewing Available Clock Networks in the Device

When you enable a clock region layer in the **Layers Settings** pane, you display the areas of the chip that are driven by global and regional clock networks. When the selected device does not contain a given clock region, the option for that category is unavailable in the dialog box.

Figure 52. Clock Regions



- Depending on the clock layers that you activate in the **Layers Settings** pane, the Chip Planner displays regional and global clock regions in the device, and the connectivity between clock regions, pins, and PLLs.
- Clock regions appear as rectangular overlay boxes with labels indicating the clock type and index. Select a clock network region by clicking the clock region. The clock-shaped icon at the top-left corner indicates that the region represents a clock network region.
- Spine/sector clock regions have a dotted vertical line in the middle. This dotted line indicates where two columns of row clocks meet in a sector clock.
- To change the color in which the Chip Planner displays clock regions, select **Tools** > **Options** > **Colors** > **Clock Regions**.

#### Related Information

- [Spine Clock Limitations](#) on page 97
- [Layers Settings](#) on page 122
- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)  
In *Intel Quartus Prime Help*

#### 6.1.3.3. Viewing Clock Sector Utilization

The Chip Planner provides a visual representation of a design's clock sector utilization.

To generate the report in the Chip Planner:

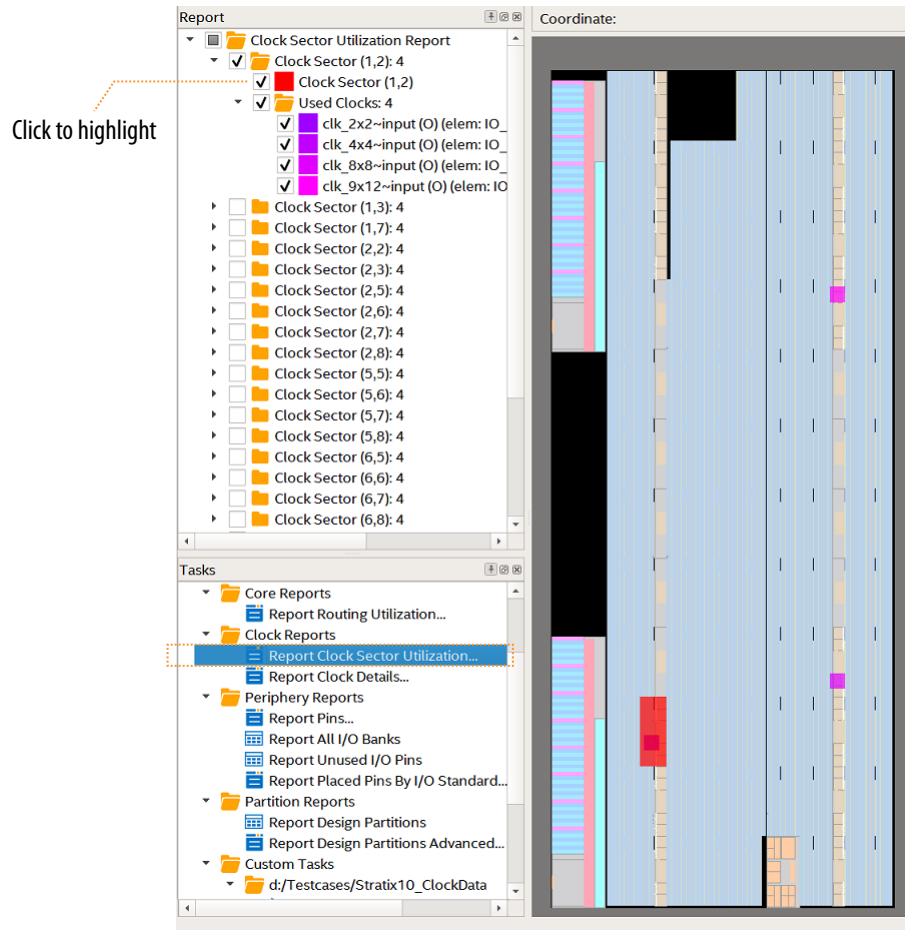
1. In the **Tasks** pane, double-click **Report Clock Sector Utilization** to open the **Report Clock Sector Utilization** dialog box.
2. If you want the report to include the source nodes, turn on **Report source nodes**.  
The equivalent TCL command appears at the bottom of the Dialog Box.
3. Click **OK**.

The report output shows the most used clock sectors.

The **Report** pane displays a list of clock sectors, with colors according to utilization. The clock sector with the highest utilization appears in red, and the sector with least utilization appears in blue.

You can turn on or off the sector visibility from the **Report** pane. You can also highlight nodes, if applicable.

**Figure 53. Clock Sector Utilization Report**



### Related Information

[Report Clock Sector Utilization dialog box \(Chip Planner\)](#)

In *Intel Quartus Prime Help*

### 6.1.3.4. Viewing Routing Congestion

The **Report Routing Utilization** task allows you to determine the percentage of routing resources in use following a compilation. This feature can identify zones with lack of routing resources, helping you to make design changes to meet routing congestion design requirements.



To view the routing congestion in the Chip Planner:

1. In the **Tasks** pane, double-click the **Report Routing Utilization** command to launch the **Report Routing Utilization** dialog box.
2. Click **Preview** in the **Report Routing Utilization** dialog box to preview the default congestion display.
3. Change the **Routing Utilization Type** to display congestion for specific resources.  
The default display uses dark blue for 0% congestion (blue indicates zero utilization) and red for 100%. You can adjust the slider for **Threshold percentage** to change the congestion threshold level.

The congestion map helps you determine whether you can modify the floorplan, or modify the RTL to reduce routing congestion. Consider:

- The routing congestion map uses the color and shading of logic resources to indicate relative resource utilization; darker shading represents a greater utilization of routing resources. Areas where routing utilization exceeds the threshold value that you specify in the **Report Routing Utilization** dialog box appear in red.
- To identify a lack of routing resources, you must investigate each routing interconnect type separately by selecting each interconnect type in turn in the **Routing Utilization Settings** dialog box.
- The Compiler's messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, can indicate difficulties fitting your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, show increased chances of not getting a valid fit.

#### Related Information

[Viewing Routing Resources](#) on page 133

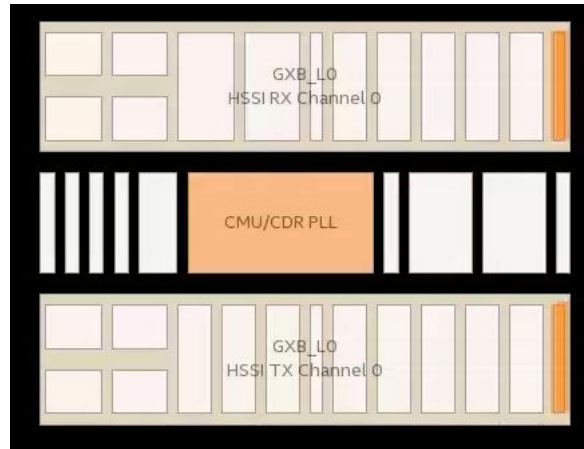
#### 6.1.3.5. Viewing I/O Banks

To view the I/O bank map of the device in the Chip Planner, double-click **Report All I/O Banks** in the **Tasks** pane.

#### 6.1.3.6. Viewing High-Speed Serial Interfaces (HSSI)

The Chip Planner displays a detailed block view of the receiver and transmitter channels of the high-speed serial interfaces. To display the HSSI block view, double-click **Report HSSI Block Connectivity** in the **Tasks** pane.

Figure 54. Intel Arria® 10 HSSI Channel Blocks



### 6.1.3.7. Viewing the Source and Destination of Placed Nodes

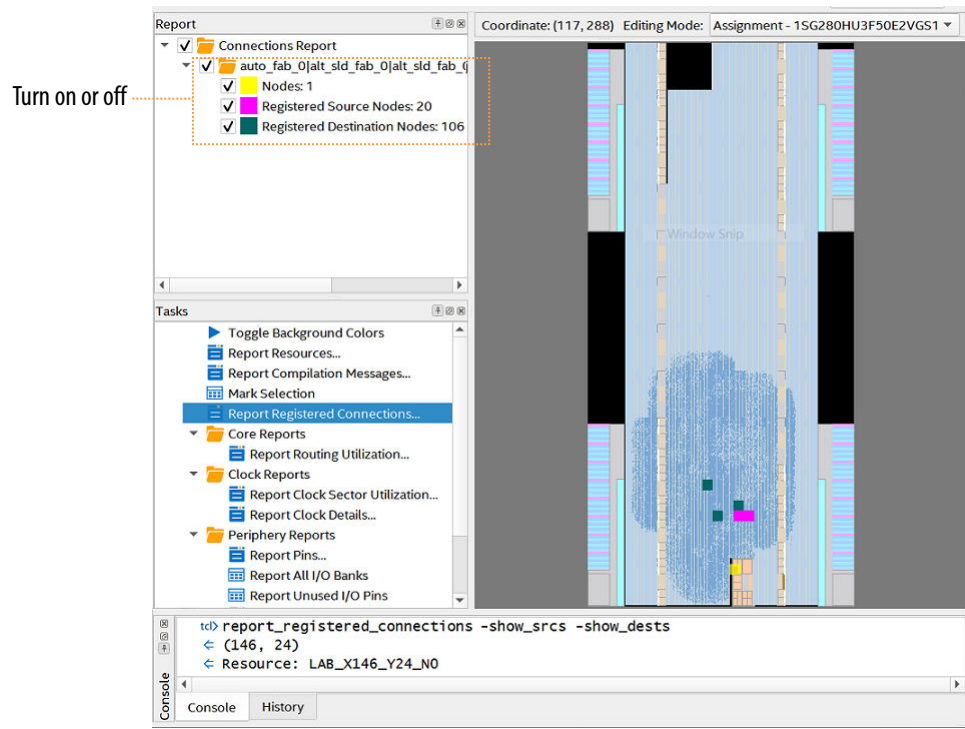
The Chip Planner allows you to view the registered fan-in or fan-outs of nodes in compiled designs with the **Report Registered Connections** task. This report is different from the **Generate Fanin/Fanout connections** report in that the source and destination nodes appear without connection lines, which may obscure the view.

1. In the Chip Planner, select one or more nodes.
2. In the **Task** pane, double-click **Report Registered Connections**.
3. Select the options from the dialog box, and click **OK**.

The **Reports** pane displays the registered source and destination nodes. Turn on or off to switch visibility in the graphic view.



Figure 55. Report Registered Connections






### Related Information

- [Viewing Fan-In and Fan-Out Connections of Placed Resources](#) on page 129
- [Expand Connections Command \(View Menu\)](#)  
In *Intel Quartus Prime Help*

### 6.1.3.8. Viewing Fan-In and Fan-Out Connections of Placed Resources

Displays the atoms that fan-in to or fan-out from a resource, including connectivity lines.

To display the fan-in or fan-out connections from a resource you selected,

1. In the Chip Planner toolbar, click the **Generate Fan-In Connections**  icon or the **Generate Fan-Out Connections**  icon.
2. To remove other connections that appear on the Chip Planner view, click the **Clear Unselected Connections**  icon.

You can also perform this actions from the Chip Planner **View** menu.


### Related Information

- [Viewing the Source and Destination of Placed Nodes](#) on page 128
- [Expand Connections Command \(View Menu\)](#)  
In *Intel Quartus Prime Help*

### 6.1.3.9. Viewing Immediate Fan-In and Fan-Out Connections

Displays the immediate fan-in or fan-out connection for the selected atom.

For example, when you view the immediate fan-in for a logic resource, you see the routing resource that drives the logic resource. You can generate immediate fan-ins and fan-outs for all logic resources and routing resources.

- To display the immediate fan-in or fan-out connections, click **View > Generate Immediate Fan-In Connections** or **View > Generate Immediate Fan-Out Connections**.
- To remove the connections displayed, use the **Clear Unselected Connections** icon  in the Chip Planner toolbar.

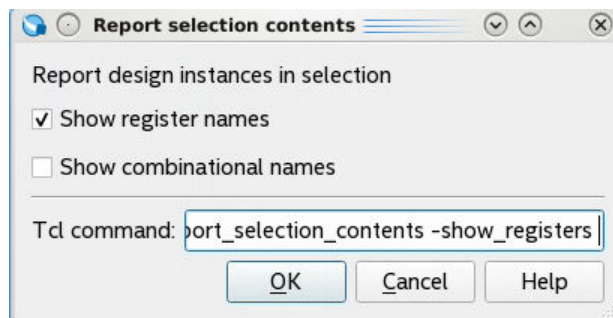
### 6.1.3.10. Viewing Selected Contents

You can view a detailed report of the contents of any area that you select in the Chip Planner. When you view the contents of a selected area, Chip Planner generates a hierarchical, color coded list of the design elements in the selection. This functionality allows you to quickly determine where the Compiler places specific modules of the design.

Follow these steps to view selected contents in the Chip Planner:

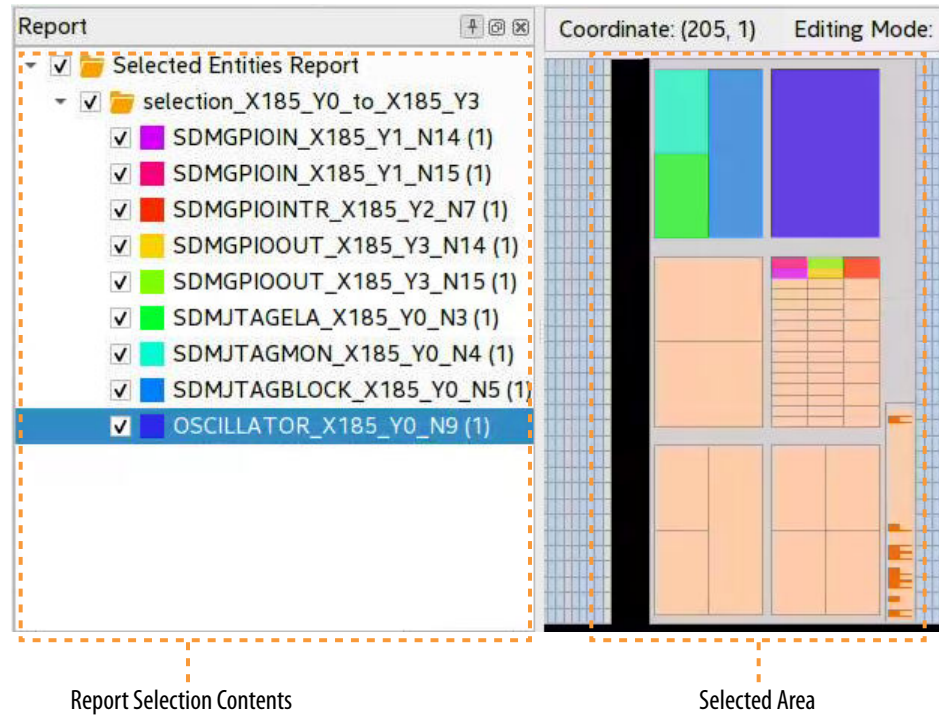
1. In the **Tasks** pane, double-click **Report Selection Contents**. The **Report Selection Contents** dialog box appears.
2. Under **Report design instances in selection**, turn on or off **Show register names** and **Show combinational names** to display names of those type in the report.

**Figure 56. Report Selection Contents Dialog Box**



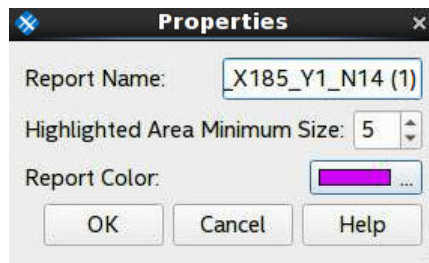
3. Click **OK**. The report generates and displays the list of selected elements in the **Reports** pane.

Figure 57. Viewing Selected Contents



- To customize the color coding of report folders, right-click any report, and then click **Properties**. You can customize the **Report Name**, **Report Color**, and the **Highlighted Area Minimum Size** for the report.


Figure 58. Selected Entities Report Properties



### 6.1.4. Exploring Paths in the Chip Planner

Use the Chip Planner to explore paths between logic elements. The following examples use the Chip Planner to traverse paths from the Timing Analysis report.

#### 6.1.4.1. Analyzing Connections for a Path

To determine the elements forming a selected path or connection in the Chip Planner, click the **Expand Connections** icon  in the Chip Planner toolbar.

### Related Information

Expand Connections Command (View Menu)  
In *Intel Quartus Prime Help*

#### 6.1.4.2. Locate Path from the Timing Analysis Report to the Chip Planner

To locate a path from the Timing Analysis report to the Chip Planner, perform the following steps:

1. Select the path you want to locate in the Timing Analysis report.
2. Right-click the path and point to **Locate Path** ► **Locate in Chip Planner**. The path appears in the **Locate History** window of the Chip Planner.

**Figure 59. Path List in the Locate History Window**

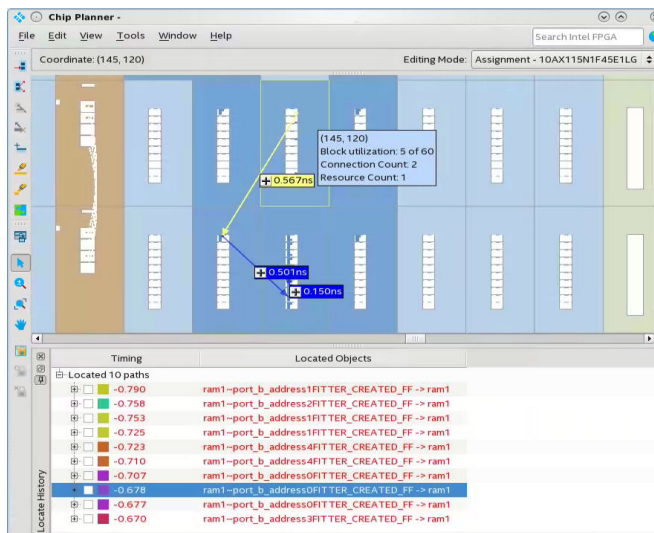
Timing	Located Objects
Located 10 paths	
-0.790	ram1~port_b_address1FITTER_CREATED_FF -> ram1
-0.758	ram1~port_b_address2FITTER_CREATED_FF -> ram1
-0.753	ram1~port_b_address1FITTER_CREATED_FF -> ram1
-0.725	ram1~port_b_address1FITTER_CREATED_FF -> ram1
-0.723	ram1~port_b_address4FITTER_CREATED_FF -> ram1
-0.710	ram1~port_b_address4FITTER_CREATED_FF -> ram1
-0.707	ram1~port_b_address0FITTER_CREATED_FF -> ram1
-0.678	ram1~port_b_address0FITTER_CREATED_FF -> ram1
-0.677	ram1~port_b_address0FITTER_CREATED_FF -> ram1
-0.670	ram1~port_b_address3FITTER_CREATED_FF -> ram1

#### 6.1.4.3. Show Delays

With the **Show Delays** feature, you can view timing delays for paths appearing in Timing Analyzer reports. To access this feature, click **View** ► **Show Delays** in the main menu. Alternatively click the Show Delays icon in the Chip Planner toolbar. To see the partial delays on the selected path, click the "+" sign next to the path delay displayed in the **Locate History** window.

For example, you can view the delay between two logic resources or between a logic resource and a routing resource.

Figure 60. Show Delays Associated in a Timing Analyzer Path

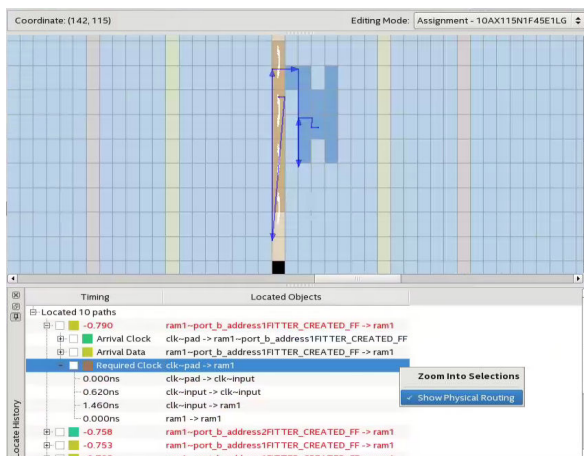


#### 6.1.4.4. Viewing Routing Resources

With the Chip Planner and the **Locate History** window, you can view the routing resources that a path or connection uses. You can also select and display the Arrival Data path and the Arrival Clock path.

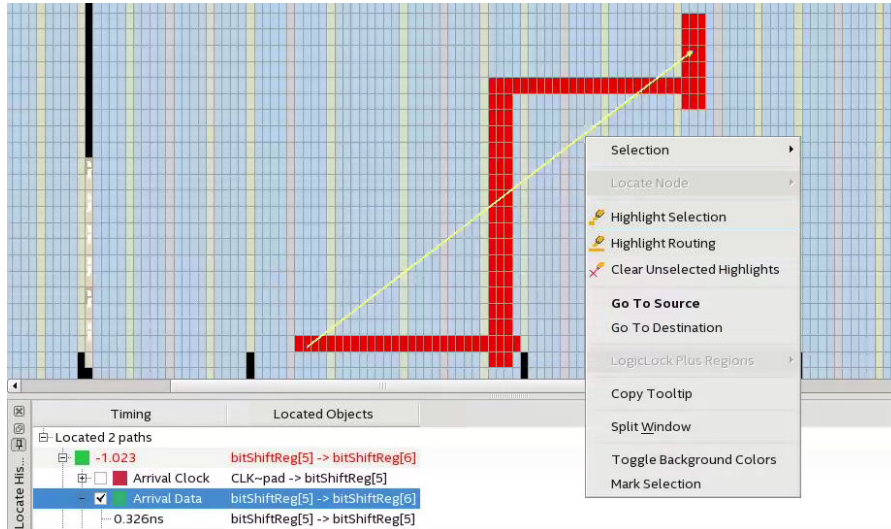
Figure 61. Show Physical Routing

In the **Locate History** window, right-click a path and select **Show Physical Routing** to display the physical path. To adjust the display, right-click and select **Zoom to Selection**.



**Figure 62. Highlight Routing**

To see the rows and columns where the Fitter routed the path, right-click a path and select **Highlight Routing**.



**Related Information**

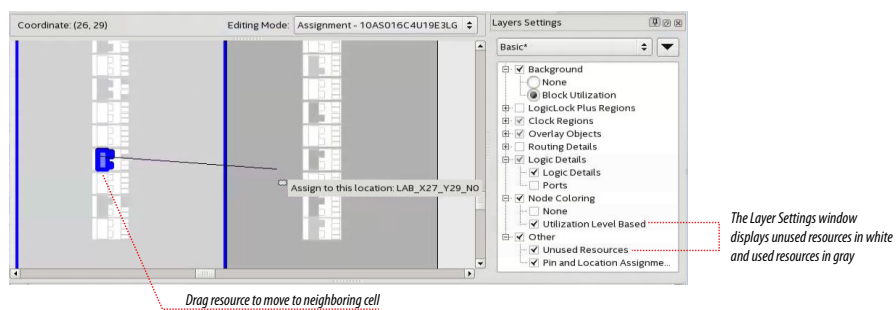
Viewing Routing Congestion on page 126

**6.1.5. Viewing Assignments in the Chip Planner**

You can view location assignments in the Chip Planner by selecting the appropriate layer, or any custom preset that displays block utilization in the **Layers Settings** pane.

The Chip Planner displays assigned resources in a predefined color (gray, by default).

**Figure 63. Viewing Assignments in the Chip Planner**



To create or move an assignment, or to make node and pin location assignments to Logic Lock regions, drag the selected resource to a new location. The Fitter applies the assignments that you create during the next place-and-route operation.

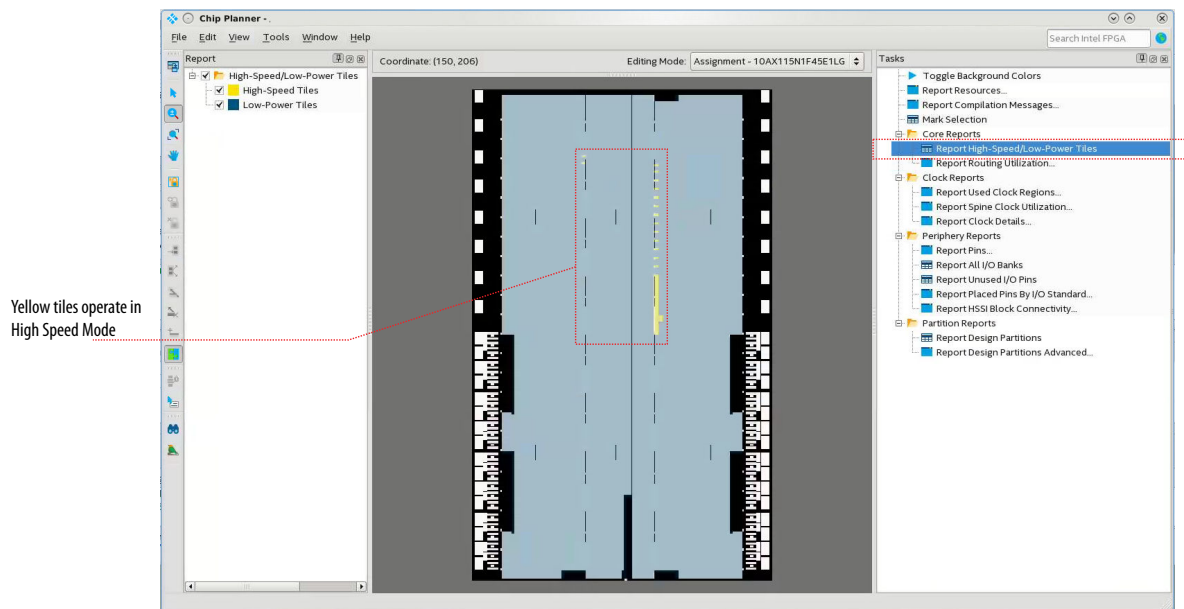


### 6.1.6. Viewing High-Speed and Low-Power Tiles in the Chip Planner

Some Intel devices have ALMs that can operate in either high-speed mode or low-power mode. The power mode is set during the fitting process in the Intel Quartus Prime software. These ALMs are grouped together to form larger blocks, called “tiles”.

To view a power map, double-click **Tasks** > **Core Reports** > **Report High-Speed/Low-Power Tiles** after running the Fitter. The Chip Planner displays low-power and high-speed tiles in contrasting colors; yellow tiles operate in a high-speed mode, while blue tiles operate in a low-power mode.

Figure 64. High-Speed and Low Power Tiles in an Intel Arria® 10 Device



## 6.2. Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner

Using Logic Lock regions with design partitions allows you to preserve the location of a block while the Fitter works in other portions of the design. When you use the Design Partition Planner with the Chip Planner, you can create partitions and Logic Lock regions in a way that benefits both the connectivity and physical locations of entities.

To use this technique in an Intel Quartus Prime Pro Edition design:

1. Compile the design.
2. Open the Chip Planner and the Design Partition Planner.
  - Click **Tools** > **Chip Planner**
  - Click **Tools** > **Design Partition Planner**
3. In the **Chip Planner** window, go to the **Tasks** pane, and double-click **Report Design Partitions**.

The Report Design Partitions task causes the Chip Planner to display the physical locations of design entities using the same colors that the entities displayed in the Design Partition Planner.

4. In the Chip Planner, click **View > Bird's Eye View**

The **Bird's Eye** View opens.

5. In the Design Partition Planner, drag all the larger entities out from their parents. Alternatively, you can right-click the entity and click **Extract from Parent**.

The Chip Planner displays the physical placement of the entities shown in the Design Partition Planner, with consistent colors between the two tools. You can view physical placement in the Chip Planner and connectivity in the Design Partition Planner.

6. Identify entities that are unsuitable to place in Logic Lock regions:
  - The Chip Planner shows an entity to be physically dispersed over noncontiguous areas of the device
  - The Design Partition Planner shows an entity to have a large number of connections to other entities.

7. Return entities unsuitable to place in Logic Lock regions to their parent, by dragging into the parent's entities.

Alternatively, right-click the entity and click **Collapse to Parent**

8. Create a partition for each remaining entity by right-clicking the entity, and then clicking **Create Design Partition**.

9. Create a Logic Lock region for each partition by right-clicking the partition, and then clicking **Create Logic Lock Region**.

### Related Information

#### [Planning Design Partitions](#)

In *Intel Quartus Prime Pro Edition User Guide: Block-Based Design*

## 6.2.1. Viewing Design Connectivity and Hierarchy

By default, when you open a compiled design, the Design Partition Planner displays the design as a single top-level entity, containing lower-level entities. If the Design Partition Planner has opened the design previously, the design appears in its last state.



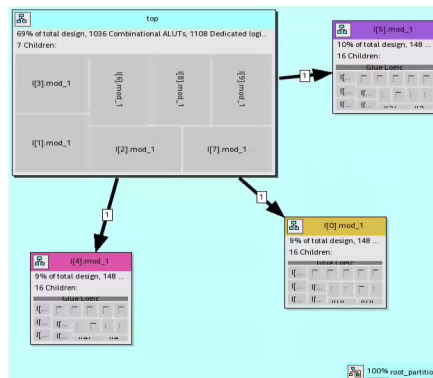



Figure 65. Top-Level Entity in the Design Partition Planner



- To show connectivity between entities, extract entities from the top-level entity by dragging them into the surrounding white space, or by right-clicking an entity and clicking **Extract from Parent** on the shortcut menu. When you extract entities, Design Partition Planner draws the connection bundles between entities, showing the number of connections between pairs of entities.

Figure 66. Partitioned Design with Connection Bundles



- To customize the appearance of connection bundles or to set thresholds for connection counts, click **View > Bundle Configuration**, and set the necessary options in the **Bundle Configuration** dialog box.
- To see bundles containing failing paths, open the Timing Analyzer, and then click **View > Show Timing Data** in the Design Partition Planner. Bundles containing failing paths are displayed in red, as are entities having nodes that reside on failing paths.
- To see detailed information about the connections in a bundle, right-click the bundle, and then click **Bundle Properties** to open the **Bundle Properties** dialog box.
- To switch between connectivity display mode and hierarchical display mode, click **View > Hierarchy Display**. Alternatively, click and hold the hierarchy icon  in the top-left corner of any entity to switch temporarily to a hierarchy display.

## 6.3. Using Logic Lock Regions in the Chip Planner

You can easily create Logic Lock regions in the Chip Planner and assign resources to them.

### 6.3.1. Viewing Connections Between Logic Lock Regions in the Chip Planner

You can view and edit Logic Lock regions using the Chip Planner. To view and edit Logic Lock regions, use **Floorplan Editing** in the **Layers Settings** window, or any layers setting mode that has the **User-assigned Logic Lock regions** setting enabled.

The Chip Planner shows the connections between Logic Lock regions. By default, you can view each connection as an individual line. You can choose to display connections between two Logic Lock regions as a single bundled connection rather than as individual connection lines. To use this option, open the Chip Planner and on the View menu, click **Inter-region Bundles**.

#### Related Information

##### [Inter-region Bundles Dialog Box](#)

For more information about the Inter-region Bundles dialog box, refer to Intel Quartus Prime Help.

### 6.3.3. Logic Lock Regions

Logic Lock regions are floorplan location constraints. When you assign instances or nodes to a Logic Lock region, you direct the Fitter to place those instances or nodes within the region. A floorplan can contain multiple Logic Lock regions.

*Note:* As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.

Logic Lock regions do not have preservation attributes, just boundaries and reservation of logic resources. You can use Intel Quartus Prime Pro Edition software to implement fully hierarchical Logic Lock region assignments.

A Logic Lock region is composed of two elements:

- **Placement Region:** Constrains logic to a specific area of the device; the Fitter places the logic in the region you specify. If you designate a region as Reserved, the Fitter cannot place other logic in the region.
- **Routing Region:** Constrains routing to a specific area. By default, routing regions are unconstrained. The routing region must encompass the placement region. A routing region cannot be reserved. For more details, refer to *Defining Routing Regions*.

#### Related Information

- [Early Place Flow](#)  
In *Intel Quartus Prime Pro Edition User Guide: Compiler*
- [Creating Logic Lock Regions](#) on page 139



### 6.3.4. Attributes of a Logic Lock Region

The following table lists the attributes of a Logic Lock region. In the Intel Quartus Prime software, the Logic Lock Regions window displays the attributes of all the Logic Lock regions in the design.

**Table 33. Attributes of Logic Lock Regions**

Name	Value	Behavior
Width	Number of columns	Specifies the width of the Logic Lock region. If <b>Size/State</b> is set to <b>Auto/Floating</b> , the attribute is set to <b>Undetermined</b> .
Height	Number of rows	Specifies the height of the Logic Lock region. If <b>Size/State</b> is set to <b>Auto/Floating</b> , the attribute is set to <b>Undetermined</b> .
Origin	Any Floorplan Location	Specifies the location of the Logic Lock region on the floorplan. The origin is at the lower left corner of the Logic Lock region.
Reserved	Off   On	Prevents the Fitter from placing other logic in the region. You cannot apply the <b>Reserved</b> assignment to routing regions.
Core-Only	Off   On	Excludes periphery resources from a region. Unlike the Intel Quartus Prime Standard Edition software, Intel Quartus Prime Pro Edition region assignments apply to periphery resources by default. If the region is designated as <b>Reserved</b> and <b>Core Only</b> , periphery resources are not reserved from the region.
Size/State	Fixed/Locked   Auto/Floating	Specifies whether you or the Fitter determine the size and placement of the Logic Lock region. <ul style="list-style-type: none"> <li>If set to <b>Fixed/Locked</b>, the default value, you define the Logic Lock region's size and placement.</li> <li>If set to <b>Auto/Floating</b>, the Fitter determines the size and placement of the Logic Lock region.</li> </ul>
Routing Region	Unconstrained   Whole Chip   Fixed with Expansion   Custom	Type of routing region. For more details, refer to <i>Defining Routing Regions</i> .

#### Related Information

[Logic Lock Regions Window](#) on page 148

### 6.3.5. Migrating Assignments between Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition

The Intel Quartus Prime Pro Edition software does not support the Intel Quartus Prime Standard Edition Logic Lock (Standard) assignments. Therefore, if you are migrating a design from Intel Quartus Prime Standard Edition to Intel Quartus Prime Pro Edition, you must convert the Logic Lock (Standard) assignments into Logic Lock assignments.

#### Related Information

[Replace Logic Lock Regions](#)

In *Intel Quartus Prime Pro Edition User Guide: Getting Started*

### 6.3.6. Creating Logic Lock Regions

### 6.3.6.1. Creating Logic Lock Regions with the Chip Planner

1. Click **View** ► **Logic Lock Regions** ► **Create Logic Lock Region**
2. Click and drag on the Chip Planner floorplan to create a region of your preferred location and size

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape does not matter.

### 6.3.6.2. Creating Logic Lock Regions with the Project Navigator

1. Perform either a full compilation or analysis and elaboration on the design.
2. If the Project Navigator is not already open, click **View** ► **Utility Windows** ► **Project Navigator**. The Project Navigator displays the hierarchy of the design.
3. With the design hierarchy fully expanded, right-click any design entity, and click **Create New Logic Lock Region**.
4. Assign the entity to the new region.

The new region has the same name as the entity.

### 6.3.6.3. Creating Logic Lock Regions with the Logic Lock Regions Window

1. Click **Assignments** ► **Logic Lock Regions Window**.
2. In the **Logic Lock Regions** window, click <<new>>.

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape does not matter.

#### Related Information

[Logic Lock Regions Window](#) on page 148

### 6.3.6.4. Defining Routing Regions

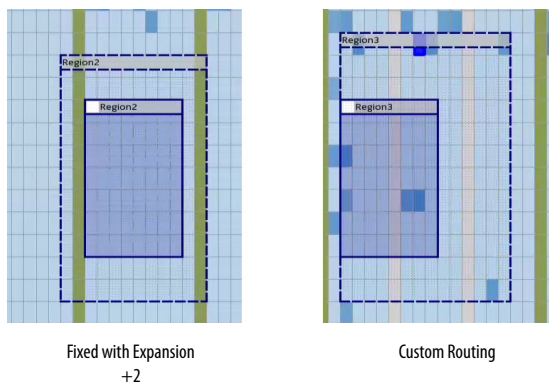
A routing region is an element of a Logic Lock region that specifies the routing area. A routing region must encompass the existing Logic Lock placement region. Routing regions cannot be set as reserved. To define the routing region, double-click the **Routing Region** cell in the **Logic Lock Regions** window, and select an option from the drop-down menu.

Valid routing region options are:

**Table 34. Routing Region Options**

Option	Description
Unconstrained (default)	Allows the fitter to use any available routes on the device.
Whole Chip	Same as Unconstrained, but writes the constraint in the Intel Quartus Prime settings file (.qsf).
Fixed with Expansion	Follows the outline of the placement region. The routing region scales by a number of rows/cols larger than the placement region.
Custom	Allows you to make a custom shape routing region around the Logic Lock region. When you select the <b>Custom</b> option, the placement and routing regions move independently in the Chip Planner. In this case, move the placement and routing regions by selecting both using the Shift key.

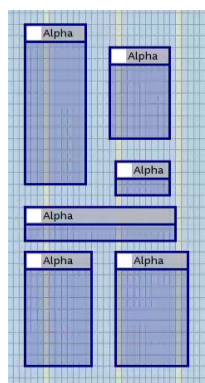
**Figure 67. Routing Regions**



### 6.3.6.5. Noncontiguous Logic Lock Regions

You can create disjointed regions by using the Logic Lock region manipulation tools. Noncontiguous regions act as a single Logic Lock region for all Logic Lock region attributes.

**Figure 68. Noncontiguous Logic Lock Region**



#### Related Information

[Merging Logic Lock Regions](#) on page 143

### 6.3.6.6. Considerations on Using Auto Sized Regions

If you use **Auto/Floating** Size/State Logic Lock regions, take into account:

- **Auto/Floating** regions cannot be reserved.
- Verify that your Logic Lock region is not empty. If you do not assign any instance to the region, the Fitter reduces the size to 0 by 0, making the region invalid.
- The region may or may not be associated with a partition. When you combine partitions with **Auto/Floating** Size/State Logic Lock regions, you get flexibility to solve your particular fitting challenges. However, every constraint that you add reduces the solutions available, and too many constraints can result in the Fitter not finding a solution. Some cases are:
  - If a partition is preserved at synthesis or not preserved, the Logic Lock region confines the logic to a specific area, allowing the Fitter to optimize the logic within the partition, and optimize the placement within the Logic Lock region.
  - If a partition is preserved at placement, routed, or final; a Logic Lock region is not an effective placement boundary, because the location of the partition's logic is fixed.
  - However, if the Logic Lock region is reserved, the Fitter avoids placing other logic in the area, which can help you reduce resource congestion.
- Once the outcome of the Logic Lock region meets your specification, you can:
  - Convert the Logic Lock region to **Fixed/Locked** Size/State.
  - Leave the Logic Lock region with **Auto/Floating** Size/State attribute and use the region as a "keep together" type of function.
  - If the Logic Lock region is also a partition, you can preserve the place and route through the partition and remove the Logic Lock region entirely.


### 6.3.7. Customizing the Shape of Logic Lock Regions

To create custom shaped Logic Lock regions, you can perform logic operations. Non-rectangular Logic Lock regions can help you exclude certain resources, or place parts of your design around specific device resources to improve performance.

**Attention:** There is no undo feature for the Logic Lock shapes for 17.1.

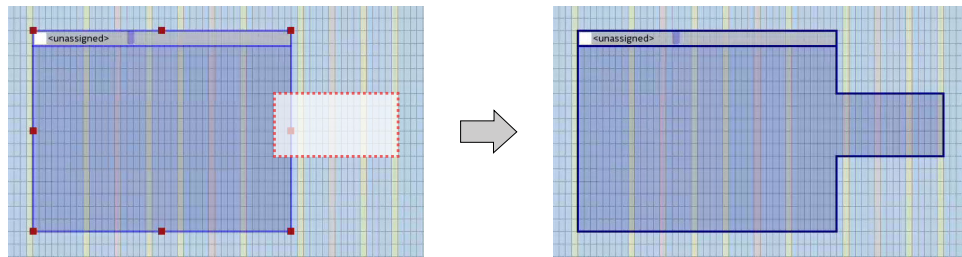
#### 6.3.7.1. Adding a New Shape to a Logic Lock Region

To add a new shape to an existing Logic Lock region, perform the following steps in the Chip Planner:

1. Select the Logic Lock region.
2. In the **Navigation** toolbar, click the **Add Logic Lock Region** icon .
3. Click and drag to generate the shape you want to add. The new shape merges automatically with the selected Logic Lock region.

**Attention:** If you selected more than one region, the operation appends the new shape to all them.

Figure 69. Using the Add Logic Lock Region Feature



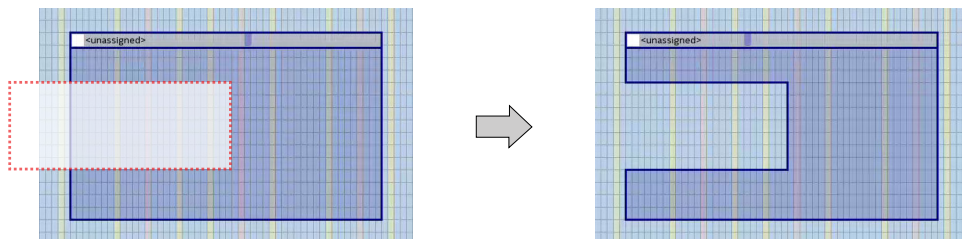
### 6.3.7.2. Subtracting Shape from Logic Lock Region

To subtract a shape from an existing Logic Lock region, perform the following steps in the Chip Planner:

1. Select the Logic Lock region.
2. In the **Navigation** toolbar, click the **Subtract Logic Lock Region** icon
3. Click and drag the shape you want to subtract. The modified region displays automatically.

The operation performs in all selected regions.

Figure 70. Using the Subtract Logic Lock Region Feature

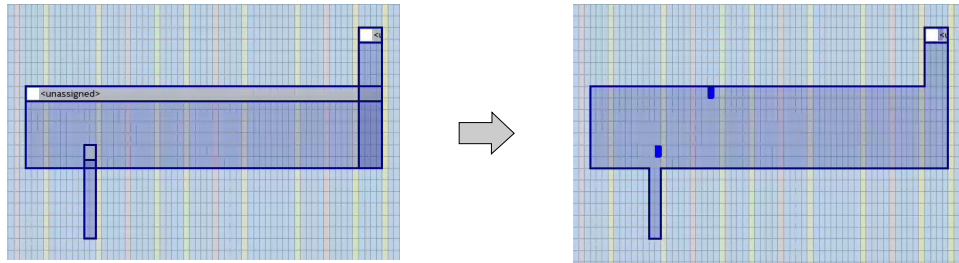


### 6.3.7.3. Merging Logic Lock Regions

To merge two or more Logic Lock regions, perform the following steps:

1. Ensure that no more than one of the regions that you intend to merge has logic assignments.
2. Arrange the regions into the locations where you want the resultant region.
3. Select all the individual regions that you want to merge by clicking each of them while pressing the Shift key.
4. Right-click the title bar of any of the selected Logic Lock regions and select **Logic Lock Regions > Merge Logic Lock Region**. The individual regions that you select merge to create a single new region.  
If you select multiple named regions, the **Merge Logic Lock Region** option is deactivated.

**Figure 71. Using the Merge Logic Lock Region command**



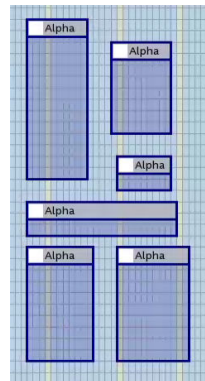
**Related Information**

[Creating Logic Lock Regions](#) on page 139

**6.3.7.4. Noncontiguous Logic Lock Regions**

You can create disjointed regions by using the Logic Lock region manipulation tools. Noncontiguous regions act as a single Logic Lock region for all Logic Lock region attributes.

**Figure 72. Noncontiguous Logic Lock Region**



**Related Information**

[Merging Logic Lock Regions](#) on page 143

**6.3.8. Placing Device Resources into Logic Lock Regions**

You can assign an entity in the design to only one Logic Lock region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a sub region without reserving the resources in the sub region.

If a Logic Lock region boundary includes part of a device resource, the Intel Quartus Prime software allocates the entire resource to that Logic Lock region.

To add an instance using the **Logic Lock Region** window, right-click the region and select **Logic Lock Properties > Add**. Alternatively, in the Intel Quartus Prime software you can drag entities from the Hierarchy viewer into a Logic Lock region's name field in the Logic Lock Regions Window.



### 6.3.8.1. Empty Logic Lock Regions

Intel Quartus Prime allows you to have Logic Lock regions with no members. Empty regions are a tool to manage space in the FPGA for future logic. This technique only works when you set the regions to **Reserved**

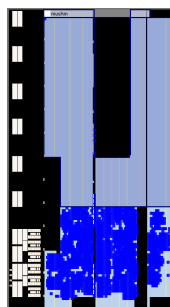
Some reasons to use empty Logic Lock regions are:

- Preliminary floorplanning.
- Complex incremental builds.
- Team based design and interconnect logic.
- Confining logic placements.

Since Logic Lock regions do not reserve any routing resources, the Fitter may use the area for routing purposes.

Use the **Core Only** attribute for empty Logic Lock regions. When you include periphery resources in empty regions, you restrict the periphery component placement, which can result in a no fit design. After you name the empty region, you can perform the same manipulations as with any populated Logic Lock Region.

**Figure 73. Logic Placed Outside of an Empty region**



The figure shows an empty Logic Lock region and the logic around it. However, some IOs, HSSIIO, and PLLs are in the empty region. This placement happens because the output port connects to the IO, and the IO is always part of the root\_partition (top-level partition).

### 6.3.8.2. Pin Assignment

A Logic Lock region incorporates all device resources within its boundaries, including memory and pins. The Intel Quartus Prime Pro Edition software does not include pins automatically when you assign an entity to a region, unless the **Core Only** attribute is off.

You can manually assign pins to Logic Lock regions; however, this placement puts location constraints on the region. The software only obeys pin assignments to locked regions that border the periphery of the device. The locked regions must include the I/O pins as resources.

### 6.3.8.3. Reserved Logic Lock Regions

The **Reserved** attribute instructs the Fitter to only place the entities and nodes that you specifically assigned to the Logic Lock region in the Logic Lock region.

The Intel Quartus Prime software honors all entity and node assignments to Logic Lock regions. Occasionally entities and nodes do not occupy an entire region, which leaves some of the region's resources unoccupied.

To increase the region's resource utilization and performance, Intel Quartus Prime software by default fills the unoccupied resources with other nodes and entities that have not been assigned to another region. To prevent this behavior, turn on **Reserved** in the **Logic Lock Regions** window.

#### 6.3.8.4. Virtual Pins

A virtual pin is an I/O element that the Compiler temporarily maps to a logic element, and not to a pin during compilation. The software implements virtual pins as LUTs. To assign a Virtual Pin, use the Assignment Editor. You can create virtual pins by assigning the **Virtual Pin** logic option to an I/O element.

When you apply the **Virtual Pin** assignment to an input pin, the pin no longer appears as an FPGA pin; the Compiler fixes the virtual pin to GND in the design. The virtual pin is not a floating node.

Use virtual pins only for I/O elements in lower-level design entities that become nodes after you import the entity to the top-level design; for example, when compiling a partial design.

**Note:** The **Virtual Pin** logic option must be assigned to an input or output pin. If you assign this option to a bidirectional pin, tri-state pin, or registered I/O element, Synthesis ignores the assignment. If you assign this option to a tri-state pin, the Fitter inserts an I/O buffer to account for the tri-state logic; therefore, the pin cannot be a virtual pin. You can use multiplexer logic instead of a tri-state pin if you want to continue to use the assigned pin as a virtual pin. Do not use tri-state logic except for signals that connect directly to device I/O pins.

In the top-level design, you connect these virtual pins to an internal node of another module. By making assignments to virtual pins, you can place those pins in the same location or region on the device as that of the corresponding internal nodes in the top-level module. You can use the **Virtual Pin** option when compiling a Logic Lock module with more pins than the target device allows. The **Virtual Pin** option can enable timing analysis of a design module that more closely matches the performance of the module after you integrate it into the top-level design.

To display all assigned virtual pins in the design with the Node Finder, you can set **Filter Type** to **Pins: Virtual**. To access the Node Finder from the Assignment Editor, double-click the **To** field; when the arrow appears on the right side of the field, click and select **Node Finder**.

#### Related Information

- [Assigning Virtual Pins with a Tcl command](#) on page 155
- [Node Finder Command \(View Menu\)](#)  
In *Intel Quartus Prime Help*

#### 6.3.8.5. Example: Placement Best Practices for Intel Arria® 10 FPGAs

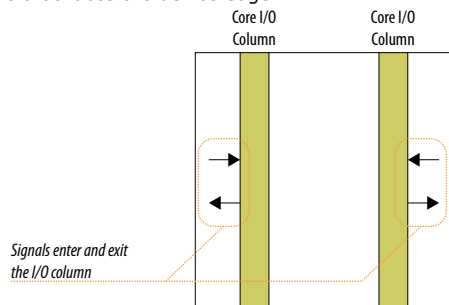
Logic Lock regions must take into account the device topology.

**Note:** As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.

This example describes how I/O Columns constrain locations in Logic Lock regions in designs targeting Intel Arria® 10 FPGAs.

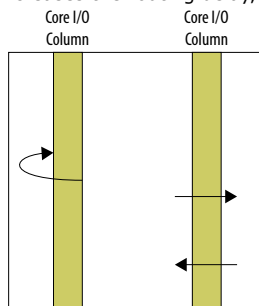
**Figure 74. I/O Columns in Intel Arria 10 FPGAs**

Intel Arria 10 FPGAs have I/O columns located in the middle of the device. Signals can only enter or exit these columns from the side that faces the device edge.



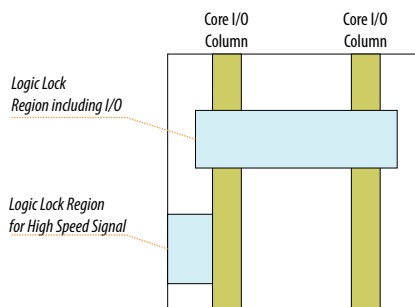
**Figure 75. Signals Crossing I/O Columns in Intel Arria 10 FPGAs**

Routing a signal to cross the I/O column increases the routing delay, and can reduce design performance.



**Figure 76. Strategic Placement for Logic Lock Regions in Intel Arria 10 FPGAs**

- If a Logic Lock region contains a register that interface with the I/O column, place the Logic Lock region so that the region covers the I/O column and the core logic, for better access to the I/O column adjacent to the outer column edge.
- For high speed signal, you can get best results if you place the Logic Lock region on the outside of the I/O column, because the fitter is less likely to cross the column and incur delay.



### Related Information

- [Early Place Flow](#)  
In *Intel Quartus Prime Pro Edition User Guide: Compiler*
- [Floorplanning a Partial Reconfiguration Design](#)  
In *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*

## 6.3.9. Hierarchical Regions

Logic Lock regions are fully hierarchical. Parent regions must completely contain all child regions. The **Reserved** and **Core-Only** assignments also apply hierarchically.

Logic Lock assignments follow the same precedence as other constraints and assignments.

You can assign an entity in the design to only one Logic Lock region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a sub region without reserving the resources in the sub region.

## 6.3.10. Additional Intel Quartus Prime Logic Lock Design Features

To complement the **Logic Lock Regions Window**, the Intel Quartus Prime software has additional features to help you design with Logic Lock regions.

### 6.3.10.1. Intel Quartus Prime Revisions Feature

When you evaluate different Logic Lock regions in your design, you might want to experiment with different configurations to achieve your desired results. The Intel Quartus Prime Revisions feature allows you to organize the same project with different settings until you find an optimum configuration.

To use the Revisions feature, choose **Project > Revisions**. You can create a revision from the current design or any previously created revisions. Each revision can have an associated description. You can use revisions to organize the placement constraints created for your Logic Lock regions.

## 6.3.11. Logic Lock Regions Window

The Logic Lock Regions Window provides a summary of all Logic Lock regions defined in your design. Use the Logic Lock Regions Window to create, assign elements, and modify properties of a Logic Lock region.

Open the Logic Lock Regions Window in the Chip Planner by clicking **View > Logic Lock Window**, and in Intel Quartus Prime by clicking **Assignments > Logic Lock Window**.



**Figure 77. Logic Lock Regions Window**

Region Name	Members	Width	Height	Origin	Reserved	Core-Only	Size/State	Routing Region
Logic Lock Regions								
inst2	inst2	21	20	X1_Y1	Off	On	Fixed/Locked	Unconstrained
inst1	inst1	2	1	X221_Y79	On	On	Fixed/Locked	Whole chip
inst3	inst3	2	1	X221_Y80	Off	On	Auto/Floating	Unconstrained
<<new>>								

You can customize the Logic Lock Regions Window by dragging and dropping the columns to change their order; you can also show and hide optional columns by right-clicking any column heading and then selecting the appropriate columns in the shortcut menu.

### Logic Lock Regions Properties Dialog Box

Use the **Logic Lock Regions Properties** dialog box to view and modify detailed information about your Logic Lock region, such as which entities and nodes are assigned to your region, and which resources are required.

To open the **Logic Lock Regions Properties** dialog box, right-click the region and select **Logic Lock Regions Properties...**

### Related Information

- [Attributes of a Logic Lock Region](#) on page 139
- [Creating Logic Lock Regions with the Logic Lock Regions Window](#) on page 140
- [Logic Lock Regions Window](#)  
In *Intel Quartus Prime Help*

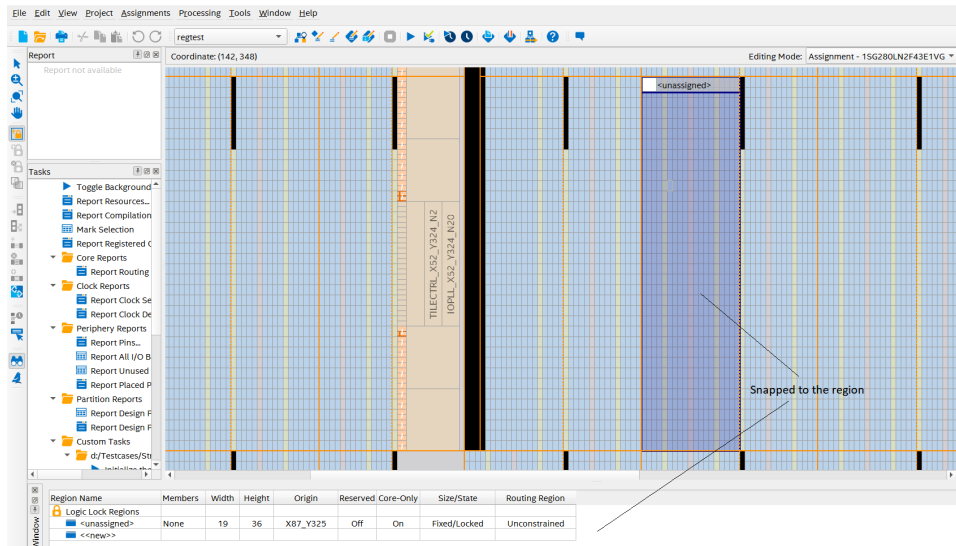
## 6.3.12. Snapping to a Region

Chip Planner supports snap-to-lab for the Logic Lock region, where the Logic Lock region created is always snapped to a lab. It is applicable to clock regions in Intel Arria 10 and Intel Agilex FPGAs, and clock sector in Intel Stratix 10 FPGA.

By default, Logic Lock regions are always snapped to the lab. You can change the default by clicking **View > Logic Lock Regions > Snap Logic Lock Region to**.

*Note:* As observed in the following image, clock regions or sectors are visible (with orange boundary) when you perform an operation in Logic Lock region (for example, create, resize, or move the region) and snap-to-clock-region. It is not visible for snap-to-lab.

Figure 78. Snapped to the Region



When you snap to a region, the Logic Lock region boundaries are displayed in the interactive mode. You can observe following behaviors:

- **Creating Region:** Left-click on the mouse to create the Logic Lock region. Upon releasing the mouse, the created Logic Lock region snaps to the containing clock region or sector.
- **Resize region (and resize diagonal):** Left-click on the mouse and drag the Logic Lock region handle. Upon releasing the mouse, the Logic Lock region resizes and snaps to the containing clock region or sector.
- **Move region:** Select and drag the Logic Lock region to highlight the clock region boundaries. Upon releasing the mouse button, the Logic Lock region moves to the new position and snaps to the containing clock region or sector.
  - **Same place and route regions are moved:** Both Logic Lock regions move and snap to the containing clock sectors.
  - **Only place | route region is moved:** The selected region moves and snaps to the clock sector, and prompts warning if the new location or size of the region does not adhere to 'place bboxes contained within route bboxes' rule.
- **Subtract or make a hole:** When performing subtract in the snap-to-clock-region mode, you create a region where the region is snapped to a clock region or a sector, and then subtract away.

## 6.4. Using User-Defined Clock Regions in the Chip Planner

You can easily create and manipulate clock regions in the Chip Planner and make clock assignments to the regions.

You can create a user-defined clock region assignment to ensure that a given global clock signal is available to resources in a certain area of the device throughout all future design iterations. In instances of congestion involving global signal resources, you may specify a smaller clock region assignment to prevent a signal from using congested clock resources in other sectors.



If you create user-defined clock regions and subsequently compile the design, those user-defined clock regions are then shown as Fitter-defined clock regions, and can no longer be edited.

### Summary of User-Defined Clock Region Feature Support


Feature	Clock Region Support
Shapes of clock regions.	Limited to rectangular regions that snap to clock sector grids.
Peripheral element assignments.	Limited to clocking design elements.
Clock region name.	Identified by the source clocking design element.
Support for multiple instances in the same regions.	Create one region per clock design element, and then specify the same definition for multiple clock design elements to assign to the same clock region.


### Using Clock Region Assignments in Intel Stratix 10 and Intel Agilex Devices

You can constrain clock regions to a rectangle whose dimensions are defined by the sector grid, as seen in the Clock Sector Region layer of the Chip Planner. The rectangle is defined by the coordinates of its bottom-left and top-right corners. For example,  $SX0, SY0, SX1, SY1$  constrains the clock to a  $2 \times 2$  region, from the bottom left of sector 0,0 to the top right of sector 1,1.

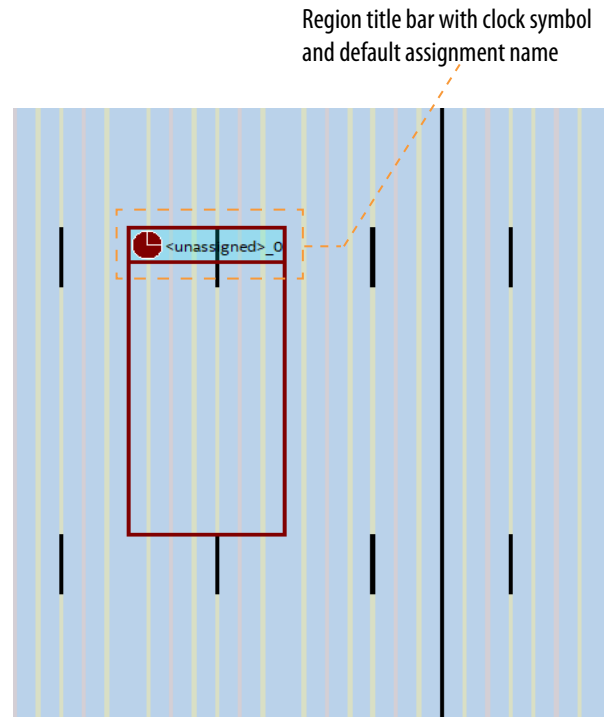
The bounding rectangle can also be specified in chip coordinates, for example  $X37 Y181 X273 Y324$ ; however, such a constraint should be sector-aligned. The Fitter automatically snaps to the smallest sector-aligned rectangle that encompasses the original assignment.

#### 6.4.1. Creating Clock Assignments with the Chip Planner

1. Select the **Create Clock Assignment**  icon, OR click **View > Clock Assignments > Create Clock Assignment**.
2. Click and drag on the Chip Planner floorplan to create a clock region of your preferred location and size. The region that you draw snaps to the smallest clock sector capable of containing the region. During interactive clock assignment operations such as creating, moving, or resizing, clock sector grids are shown in orange to facilitate positioning of the clock region relative to the clock sectors.

A clock symbol  in the region title bar identifies it as a clock region. By default, the newly created clock region has the name *unassigned*, until you assign a clock signal from the context menu.

**Figure 79. Newly Created Clock Region**



### 6.4.2. Resizing a Clock Assignment

1. Select the clock assignment. Handles appear on each side of the region and at the corners.
2. Position the crosshairs over the handle of your choice and the resize mouse cursor appears.
3. Hold the left mouse button and drag the resize cursor to expand or shrink the boundary of the clock assignment. When you release the mouse button, the clock assignment boundaries snap to the nearest containing clock sector grid.

### 6.4.3. Moving a Clock Assignment

1. Select the clock assignment.
2. Position the crosshairs over the clock assignment title bar and the move cursor appears.
3. Hold the left mouse button and drag the clock assignment to the desired new location.

### 6.4.4. Deleting a Clock Region Assignment





1. Select the clock assignment that you want to delete.
2. Right-click the clock assignment title bar to display the context menu, OR select **View** from the main menu bar.
3. Click **Clock Assignments > Delete Clock Assignment**.
4. You are prompted to confirm that you want to delete the selected clock assignment. Click **Yes** to confirm the deletion.

The specified clock region assignment is deleted from the system.

### 6.4.5. Assigning a Clock Signal to a Clock Region

1. Right-click the clock assignment title bar to display the context menu, OR select **View** from the main menu bar.
2. Click **Clock Assignments > Set Clock Signal Name**.
3. In the **Set Clock Signal Name** dialog box, browse to or type the desired clock signal name.
4. Click Ok.

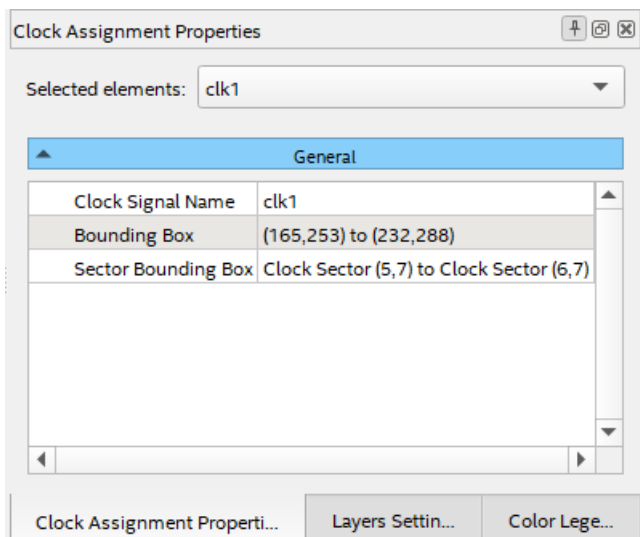
The system renames the clock assignment according to the name of the specified clock signal.

### 6.4.6. Clock Assignment Properties

The Clock Assignment Properties pane displays properties of the selected clock assignment.

By default, the Clock Assignment Properties pane appears on a tab at the right side of the Chip Planner.

**Figure 80. Clock Assignment Properties Pane**



## 6.5. Scripting Support

You can run procedures and specify the settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt.

### Related Information

- [Tcl Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*
- [Command Line Scripting](#)  
In *Intel Quartus Prime Pro Edition User Guide: Scripting*

### 6.5.1. Creating Logic Lock Assignments with Tcl commands

The Intel Quartus Prime software supports Tcl commands to create or modify Logic Lock assignments.

**Note:** Specify node names by using the full hierarchy path to the node.

#### Create or Modify a Placement Region

You can create the Logic Lock region from the GUI, or add the region directly to the QSF. The QSF entry contains the X/Y coordinates of the vertices and the Placement Region name.

The following assignment creates a new placement region with bounding box coordinates X46 Y36 X65 Y49:

```
set_instance_assignment -name PLACE_REGION "X46 Y36 X65 Y49" -to <node names>
```

- You can use the same command format to modify an existing assignment.
- To specify a non-rectangular or disjoint region, use a semicolon (;) as the delimiter between two or more bounding boxes.
- Assign multiple instances to the same region with multiple PLACE\_REGION instance assignments.

#### Create or Modify a Routing Region

The following assignment creates a routing region with bounding box coordinates X5 Y5 X30 Y30:

```
set_instance_assignment -name ROUTE_REGION -to <node names> "X5 Y5 X30 Y30"
```

- You can use the same command format to modify an existing assignment.
- All instances with a routing region assignment must have a respective placement region; the routing region must fully contain the placement region.

#### Specify a Region as Reserved

The following assignment reserves an existing region:

```
set_instance_assignment -name <instance name> RESERVE_PLACE_REGION -to <node names> ON
```

- You can only reserve placement regions.



### Specify a Region as Core Only

By default, the Intel Quartus Prime Pro Edition software includes pins in Logic Lock assignments. To specify a region as core only (that is, periphery logic in the instance that is not constrained), use the following assignment:

```
set_instance_assignment -name <instance name> CORE_ONLY_PLACE_REGION -to <node names> ON
```

### Related Information

[Creating Logic Lock Regions](#) on page 139

## 6.5.2. Assigning Virtual Pins with a Tcl command

Use the following Tcl command to turn on the virtual pin setting for a pin called `my_pin`:

```
set_instance_assignment -name VIRTUAL_PIN ON -to my_pin
```

### Related Information

- [Virtual Pins](#) on page 146
- [Node Finder Command \(View Menu\)](#)  
In *Intel Quartus Prime Help*

## 6.5.3. Logic Lock Region Assignment Examples

These examples show the syntax of Logic Lock region assignments in the `.qsf` file. Optionally, enter these assignments in the Assignment Editor, the Logic Lock Regions Window, or the Chip Planner.

### Example 1. Assign Rectangular Logic Lock Region

Assigns a rectangular Logic Lock region to a lower right corner location of (10,10), and an upper right corner of (20,20) inclusive.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"
```

### Example 2. Assign Non-Rectangular Logic Lock Region

Assigns instance with full hierarchical path "`x|y|z`" to non-rectangular L-shaped Logic Lock region. The software treats each set of four numbers as a new box.

```
set_instance_assignment -name PLACE_REGION -to x|y|z "X10 Y10 X20 Y50; X20 Y10 X50 Y20"
```

### Example 3. Assign Subordinate Logic Lock Instances

By default, the Intel Quartus Prime software constrains every child instance to the Logic Lock region of its parent. Any constraint to a child instance intersects with the constraint of its ancestors. For example, in the following example, all logic beneath "`a|b|c|d`" constrains to box (10,10), (15,15), and not (0,0), (15,15). This result occurs because the child constraint intersects with the parent constraint.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"  
set_instance_assignment -name PLACE_REGION -to a|b|c|d "X0 Y0 X15 Y15"
```

#### Example 4. Assign Multiple Logic Lock Instances

By default, a Logic Lock region constraint allows logic from other instances to share the same region. These assignments place instance *c* and instance *g* in the same location. This strategy is useful if instance *c* and instance *g* are heavily interacting.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X20 Y20"
```

#### Example 5. Assigned Reserved Logic Lock Regions

Optionally reserve an entire Logic Lock region for one instance and any of its subordinate instances.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment -name RESERVE_PLACE_REGION -to a|b|c ON

# The following assignment causes an error. The logic in e|f|g is not
# legally placeable anywhere:
# set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X20 Y20"

# The following assignment does *not* cause an error, but is effectively
# constrained to the box (20,10), (30,20), since the (10,10),(20,20) box is
# reserved
# for a|b|c
set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X30 Y20"
```

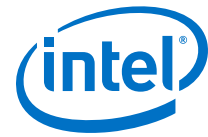
## 6.6. Analyzing and Optimizing the Design Floorplan Revision History

The following revision history applies to this chapter:

**Table 35. Document Revision History**

Document Version	Intel Quartus Prime Version	Changes
2019.07.30	19.3.0	Added new <i>Using User-Defined Clock Regions in the Chip Planner</i> section.
2019.07.01	19.1.0	Added new "Snapping to a Region" topic that describes the <b>Snap Logic Lock Region to</b> option.
2019.04.01	19.1.0	<ul style="list-style-type: none"> <li>Added new "Viewing Selected Contents" topic that describes a new report listing selected design elements.</li> </ul>
2018.09.24	18.1.0	<ul style="list-style-type: none"> <li>Added topic: <i>Viewing Clock Sector Utilization</i></li> <li>Added topic: <i>Viewing the Source and Destination of Placed Nodes.</i></li> <li>Renamed topic: <i>Generating Fan-In and Fan-Out Connections to Viewing Fan-In and Fan-Out Connections of Placed Resources.</i></li> </ul>
2018.05.07	18.0.0	<ul style="list-style-type: none"> <li>Added recommendations for using iterative methods for floorplanning.</li> </ul>
2017.11.06	17.1.0	<ul style="list-style-type: none"> <li>Changed instances of <i>LogicLock Plus</i> to <i>Logic Lock</i>.</li> <li>Added support for auto-sized Logic Lock regions.</li> <li>Added support for empty Logic Lock regions.</li> <li>Added topics: <i>Considerations on Using Auto Sized Regions, Creating Partitions and Logic Lock Regions with the Design Partition Planner and Chip Planner.</i></li> </ul>

*continued...*



Document Version	Intel Quartus Prime Version	Changes
2017.05.08	17.0.0	<ul style="list-style-type: none"> <li>Chapter reorganization and content update.</li> <li>Added figures: Clock Regions, Path List in the Locate History Window, Show Physical Routing, Using the Add Rectangle Feature, Using the Subtract Rectangle Feature, Creating a Hole in a LogicLock Region, Noncontiguous LogicLock Region, Routing Regions, Logic Placed Outside of an Empty Region.</li> <li>Updated figures: HSSI Channel Blocks, Highlight Routing, High-Speed and Low Power Tiles in an Arria 10 Device, Show Delays Highlight Routing, Viewing Assignments in the Chip Planner, LogicLock Plus Regions Window, Using the Merge LogicLock Plus Region Command.</li> <li>Created topics: <i>Adding Rectangle to a LogicLock Plus Region</i>, <i>Subtracting Rectangle from a LogicLock Plus Region</i>.</li> <li>Moved topic: <i>Viewing Critical Paths to Timing Closure and Optimization</i> chapter and renamed to <i>Critical Paths</i>.</li> <li>Renamed topic: <i>Creating Non-Rectangular LogicLock Plus Regions to Merging LogicLock Plus Regions</i>.</li> <li>Renamed topic: <i>Chip Planner Overview</i> to <i>Design Floorplan Analysis in the Chip Planner</i>.</li> <li>Renamed chapter from <i>Analyzing and Optimizing the Design Floorplan with the Chip Planner</i> to <i>Analyzing and Optimizing the Design Floorplan</i>.</li> </ul>
2016.10.31	16.1.0	<ul style="list-style-type: none"> <li>Implemented Intel rebranding.</li> <li>Added topic describing how to create a hole in a LogicLock Plus region.</li> </ul>
2016.05.02	16.0.0	Updated information on creating LogicLock Plus regions.
2015.11.02	15.1.0	<ul style="list-style-type: none"> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> <li>Added information on how to use LogicLock regions.</li> </ul>
2015.05.04	15.0.0	Added information about color coding of LogicLock regions.
2014.12.15	14.1.0	Updated description of Virtual Pins assignment to clarify that assigned input is not available.
June 2014	14.0.0	Updated format
November 2013	13.1.0	Removed HardCopy device information.
May 2013	13.0.0	Updated "Viewing Routing Congestion" section Updated references to Quartus UI controls for the Chip Planner
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	<ul style="list-style-type: none"> <li>Updated for the 11.0 release.</li> <li>Edited "LogicLock Regions"</li> <li>Updated "Viewing Routing Congestion"</li> <li>Updated "Locate History"</li> <li>Updated Figures 15-4, 15-9, 15-10, and 15-13</li> <li>Added Figure 15-6</li> </ul>
December 2010	10.1.0	<ul style="list-style-type: none"> <li>Updated for the 10.1 release.</li> </ul>
July 2010	10.0.0	<ul style="list-style-type: none"> <li>Updated device support information</li> <li>Removed references to Timing Closure Floorplan; removed "Design Analysis Using the Timing Closure Floorplan" section</li> <li>Added links to online Help topics</li> <li>Added "Using LogicLock Regions with the Design Partition Planner" section</li> </ul>

**continued...**



Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"><li>• Updated "Viewing Critical Paths" section</li><li>• Updated several graphics</li><li>• Updated format of Document revision History table</li></ul>
November 2009	9.1.0	<ul style="list-style-type: none"><li>• Updated supported device information throughout</li><li>• Removed deprecated sections related to the Timing Closure Floorplan for older device families. (For information on using the Timing Closure Floorplan with older device families, refer to previous versions of the Quartus Prime Handbook, available in the Documentation Archive.)</li><li>• Updated "Creating Nonrectangular LogicLock Regions" section</li><li>• Added "Selected Elements Window" section</li><li>• Updated table 12-1</li></ul>
May 2008	8.0.0	<ul style="list-style-type: none"><li>• Updated the following sections:<ul style="list-style-type: none"><li>"Chip Planner Tasks and Layers"</li><li>"LogicLock Regions"</li><li>"Back-Annotating LogicLock Regions"</li><li>"LogicLock Regions in the Timing Closure Floorplan"</li></ul></li><li>• Added the following sections:<ul style="list-style-type: none"><li>"Reserve LogicLock Region"</li><li>"Creating Nonrectangular LogicLock Regions"</li><li>"Viewing Available Clock Networks in the Device"</li></ul></li><li>• Updated Table 10-1</li><li>• Removed the following sections:<ul style="list-style-type: none"><li>Reserve LogicLock Region Design Analysis Using the Timing Closure Floorplan</li></ul></li></ul>

## 7. Using the ECO Compilation Flow

In a typical FPGA project development cycle, the specification of the programmable logic portion of the design can change during the design process. The Intel Quartus Prime software supports these last-minute, targeted *engineering change orders* (ECOs), even after full compilation is complete.

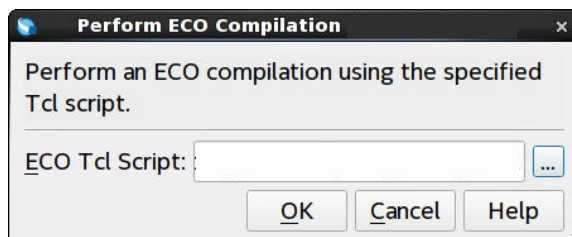
ECOs typically occur during the design verification stage. For example, during verification you may determine that the design requires a small change, such as a netlist connection change, correcting a LUT logic error, or placing a node in a new location. Implementing an ECO change, rather than changing RTL and fully recompiling the design, requires significantly less time, and changes only the affected logic.

You specify the ECO commands in a Tcl script using the `::quartus::eco` package.

**Note:** The Intel Quartus Prime Pro Edition software supports ECOs for Intel Stratix 10 and Intel Agilex devices only.

### 7.1. ECO Compilation Flow

1. Identify an ECO modification you want to make in a compiled design.
2. Determine if ECO commands support the change, by reviewing [ECO Commands](#) on page 162 and [ECO Command Limitations](#) on page 170.
3. Create a Tcl script, as [ECO Tcl Script Example](#) on page 160 shows.
4. Before running ECO compilation, click **Project** ► **Archive Project** and archive the compilation database and output file set.
5. Click **Processing** ► **Start** ► **Perform ECO Compilation**.



6. Specify the **ECO Tcl Script** file, and click **OK**. The Fitter processes the ECO commands and updates the finalized netlist. The Fitter generates an error if you specify any commands incorrectly. The changes apply when the Fitter processing completes.
7. View the ECO results in post-fit analysis tools, such as the Compilation Report, Timing Analyzer, Netlist Viewer, or Chip Planner. To view ECO changes in the Fitter report, click **Processing** ► **Compilation Report** ► **Fitter** ► **ECO Changes**.

Figure 81. Example of ECO Changes Report

	Action	Node	Original Value	Changed Value	Iteration
1	Modify Lutmask	i16	0x1	0x2	Current

As an alternative to the GUI methods, you can use the following commands to run the ECO Tcl scripts. If running from command line, any active Intel Quartus Prime GUI application does not refresh. Close and reopen the project to refresh the GUI.

```
$ quartus_fit -s
load_package eco
project_open <project_name>
eco_load_design
...
eco_commit_design
project_close
```

*Note:* If you rerun the Fitter on a design after implementing an ECO, the Fitter overwrites the ECO changes. Update RTL, IP parameters, and recompile the design to permanently implement the ECO changes.

## 7.2. ECO Tcl Script Example

The following shows an example ECO Tcl script that places existing nodes in new locations:

Figure 82. ECO Tcl Script Example

```
1 # Place Nodes i11 and i8 in New Locations
2
3 place_node -name i11 -location "X24 Y63"
4 place_node -name i8 -location "X24 Y63 X24 Y63"
```

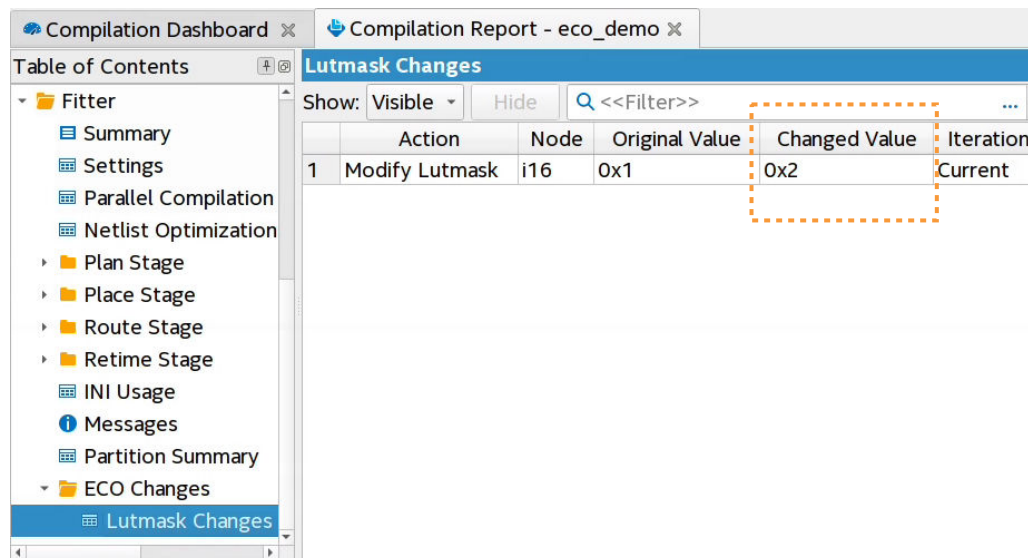




### 7.3. Viewing ECO Compilation Reports

The Compiler generates a report showing the details of each ECO compilation that you run successfully. You can view the report contents in the **ECO Changes** report under **Fitter** in the Compilation Report.

**Figure 83. Example of ECO Changes Report**



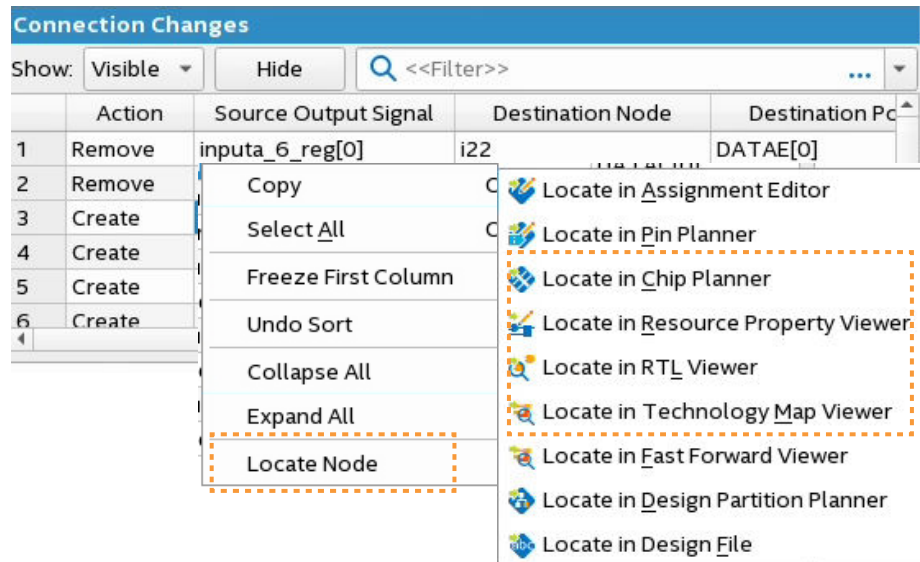
Alternatively, you can view this data in the generated `fit.eco` file. The Compiler organizes the report output according to the category of ECO change, such as Placement Changes. The table specifies the "Changes in Previous ECO Runs" and "Changes in Current ECO Run".

**Figure 84. ECO Report Example**

```
+-----+
; Placement Changes
+-----+-----+-----+
; Name           ; Location ; Iteration ;
+-----+-----+-----+
; GND-ECO_INSERTED ; X78_Y72 ; Previous ;
; new_node        ; X78_Y72 ; Previous ;
; new_wirelut     ; X78_Y72 ; Previous ;
; new_wirelut     ; X24_Y24 ; Current  ;
+-----+-----+-----+
```

Locate nodes from the Fitter's ECO reports to confirm ECO changes.

Figure 85. Locate Node from Fitter ECO Reports



## 7.4. ECO Commands

The Intel Quartus Prime Pro Edition software supports the following ECO commands:

*Note:* Check available arguments by running `<eco_command> -h|help|long_help`.

[ECO Command Quick Reference](#) on page 163

[make\\_connection](#) on page 163

[remove\\_connection](#) on page 164

[modify\\_lutmask](#) on page 165

[adjust\\_pll\\_refclk](#) on page 165

[modify\\_io\\_slew\\_rate](#) on page 166

[modify\\_io\\_current\\_strength](#) on page 166

[modify\\_io\\_delay\\_chain](#) on page 166

[create\\_new\\_node](#) on page 167

[remove\\_node](#) on page 168

[place\\_node](#) on page 168

[unplace\\_node](#) on page 169

[create\\_wirelut](#) on page 169



## 7.4.1. ECO Command Quick Reference

Table 36. ECO Command Quick Reference

ECO Change	ECO Commands
<b>Route</b>	<code>make_connection -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt;</code> <code>remove_connection -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt;</code>
<b>Tie-Off</b>	<code>make_connection -tieoff &lt;VCC/GND&gt; -to &lt;node&gt; -port &lt;port&gt;</code>
<b>Lutmask</b>	<code>modify_lutmask -to &lt;node&gt; [-eqn &lt;lut equation&gt;] [-mask 0x00]</code>
<b>Slew Rate</b>	<code>modify_io_slew_rate &lt;value&gt; -to &lt;pin_name&gt;</code>
<b>Current Strength</b>	<code>modify_io_current_strength &lt;value&gt; -to &lt;pin_name&gt;</code>
<b>Delay Chains</b>	<code>modify_io_delay_chain &lt;value&gt; -type &lt;io_type&gt; -to &lt;pin_name&gt;</code>
<b>Update MIF</b>	<code>update_mif_files</code>
<b>IOPLL Ref Clock</b> (Intel Stratix 10 devices only)	<code>adjust_pll_refclk -to &lt;pll name&gt; -refclk &lt;freq&gt;</code>
<b>Create New Node</b>	<code>create_new_node -type &lt;LUT FF&gt; -name &lt;name&gt;</code>
<b>Remove Node</b>	<code>remove_node -name &lt;name&gt;</code>
<b>Place Node</b>	<code>place_node -name &lt;name&gt; [-location &lt;location&gt;]</code>
<b>Unplace Node</b>	<code>unplace_node -name &lt;name&gt;</code>
<b>Create Wirelut</b>	<code>create_wirelut -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt; [-location &lt;location&gt;]</code>

### 7.4.2. make\_connection

#### Description

Connects the source signal to the destination block port. If the port has an existing connection, the command removes the previous connection and connects it to the signal you specify. The actual routing change occurs implicitly when appropriate. You can locate node names by right-clicking a node in Netlist Viewer, and then clicking **Properties**.

`make_connection` also supports adding connections from and to Hyper-Registers. In the case of Hyper-Registers, the command first disconnects the destination port, before making a new connection. You can run the `make_connection` command to specify a replacement signal source or destination.

If a port is shared among multiple RAM slice atoms, then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

### Usage

The following example connects `top|a_out` to the D input port of node `top|x`.

```
make_connection -from top|a_out -to top|x -port D
```

### Arguments

*from* Output net of the source block of the new connection.

*to* Name of the destination block.

*port* The input port name of the destination block.

### Options

*tieoff* To explicitly tie off an input port to VCC or GND.

VCC or GND

Example:

```
make_connection -tieoff VCC -to {node1} -port DATAA
```

*to* Name of the destination block.

*port* The input port name of the destination block.

## 7.4.3. remove\_connection

### Description

Disconnects the `src` signal from the destination block port. The actual routing change occurs implicitly. You can locate node names by right-clicking a node in Netlist Viewer, and then clicking **Properties**.

If a port is shared among multiple RAM slice atoms, then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

### Usage

The following example disconnects `top|a_out` from the D input port of node `top|x`, and set `top|x:D` to a disconnected state.

```
remove_connection -from top|a_out -to top|x -port D
```

### Arguments

*from* Output net of the source block of the current connection.

*to* Name of the destination block.



*port* The input port name of the destination block.

## 7.4.4. modify\_lutmask

### Description

Modifies the lutmask of the matching destination node, with the lutmask value (*-mask*) in binary or hexadecimal, or with the equivalent lutmask value (*-eqn*) computed from specified logical equation.

### Usage

The following example disconnects `top|a_out` from the D input port of node `top|x`, and set `top|x:D` to a disconnected state.

```
modify_lutmask -to top|lut_c -eqn {a&b&c}  
modify_lutmask -to top|lut_a -mask 0xFF00FF00  
modify_lutmask -to top|lut_b -mask 0b111111111001010
```

### Arguments

*eqn* The logical equation of the inputs (A, B, C, D, E, F). The supported lexical tokens include AND(' & '), OR(' | '), XOR(' ^ '), NOT(' ! '), OPEN\_BRACE(' ( '), CLOSE\_BRACE(' ) '). Specify *-mask* or *-eqn*.

*to* Destination atom name.

*mask* The lutmask value to be modified in binary or hexadecimal format. Specify *-mask* or *-eqn*.

**Note:** When you view the lutmask equations in the Resource Property Viewer, the equations display in terms of F0/F1/F2/F3 LUTs for A, B, C and D inputs. For LUTs also using E or F inputs, you must combine these sub-functions using the connectivity that the ALM diagram shows for the E and F muxes.

## 7.4.5. adjust\_pll\_refclk

### Description

Changes the IOPLL frequencies by modifying the input reference clock frequency. The following stipulations apply:

- Maintain the original refclk and outclk ratios.
- The IOPLLs you change cannot generate IP clocks.
- Cascaded IOPLLs must connect directly (no clock gates in between them).
- IOPLLs cannot be in 'nondedicated' compensation modes.
- For all IOPLLs, outclks duty cycle equals 50 and phase shift equals 0.
- No support for Intel Agilex devices.

### Usage

The following example adjusts the `*pll_main*` IOPLL by modifying the input clock frequency to 100 MHz.

```
adjust_pll_refclk -to {*pll_main*} -refclk 100
```

### Arguments

`to` Instance name of upstream IOPLL that you want to adjust. Escape any `[` or `]` characters in the target name.

`refclk` New refclk frequency value in MHz.

## 7.4.6. modify\_io\_slew\_rate

### Description

Implements the I/O pin slew setting rate that you specify for the I/O pin.

### Usage

```
modify_io_slew_rate 1 -to top|ipin
```

### Arguments

`to` Instance name of destination pin that you want to modify.

## 7.4.7. modify\_io\_current\_strength

### Description

Implements the change to the I/O pin current strength setting that you specify for the I/O pin.

### Usage

```
modify_io_current_strength 3mA -to top|ipin
```

### Arguments

`to` Instance name of the destination pin that you want to modify.

## 7.4.8. modify\_io\_delay\_chain

### Description

Implements the change to the delay chain settings that you specify for the I/O pin.

### Usage

```
modify_io_delay_chain 3 -to top|ipin -type input
```



## Arguments

*type* Specifies one of the following I/O types: *input*, *output*, *oe*, *io\_12\_lane\_input*, *io\_12\_lane\_input\_strobe*

*to* Instance name of I/O pin that you want to modify the delay chain settings.

## 7.4.9. create\_new\_node

### Description

Creates a LUT cell or flip-flop in the design netlist. The following use cases apply:

- Adding a gate to fix a logic bug.
- Adding a wire LUT to help add hold delay.
- Creates a new flip-flop or flip-flops where needed.

The name of the new node is hierarchical. Therefore, when creating node *a|b|c|d*, you must ensure that hierarchy *a|b|c* exists in the netlist. If the source or destination node lies under a partition, the new LUT inserts under that partition.

*Note:* This command does not support extended or arithmetic LUTs.

After creating the new node, you can run the following commands to connect, modify the lutmask, or place the new node:

1. Run *make\_connection* to connect to the new LUT's DATA inputs and output port.
2. Run *modify\_lutmask* to change the lutmask for the new LUT.
3. Run *place\_node* to place (and subsequently route) the new LUT.

This flow ensures that all routing requirements are analyzed when determining a legal placement for the new node.

### Usage

The following example creates a *new\_lut* LUT node with input ports *DATAA* and *DATAB*, and with outputs connected accordingly. *modify\_lutmask* the modifies the lutmask to perform A&B logic. *place\_node* next places the new LUT. The connections route after node placement is complete.

```
create_new_node -name new_lut -type lut
make_connection -from src_a -to new_lut -port DATAA
make_connection -from src_b -to new_lut -port DATAB
make_connection -from new_lut -to dst_reg -port D
modify_lutmask -to new_lut -eqn A&B
place_node -name new_lut
```

To connect to a new flip-flop node that you create, use the `make_connection` command to connect to the flip-flop data port (D), and control ports (CLK, ENA, SCLR, CLRN), and from its output Q port. You must place the new flip-flop node with the `place_node` command. The connections are automatically routed after the `place_node` command.

```
create_new_node -name my_ff -type ff
make_connection -from reg0 -to my_ff -port D
make_connection -from clk -to my_ff -port CLK
make_connection -from my_ff -to reg1 -port D
place_node -name my_ff -location "X10 Y10 X10 Y10"
```

### Arguments

*name* Name of the new LUT of flip-flop node.

## 7.4.10. remove\_node

### Description

Removes a LUT cell or flip-flop from the design netlist.

### Usage

The following example deletes a flip-flop node with the name `ff1`:

```
remove_node -name ff1
```

### Arguments

*name* Name of the LUT of flip-flop node to delete.

## 7.4.11. place\_node

### Description

Places the node that you specify in a location that the ECO Fitter selects. Optionally, you can specify the `location` argument to assign a specific device region location. You can also run this command for nodes already placed by the Fitter.

`place_node` also supports placement of newly added or existing flip-flops. `place_node` does not support Hyper-Register locations.

### Usage

The following examples show three placement cases. For `node1`, the ECO Fitter determines the placement location. For `node2`, the command specifies the exact LAB location constraint. For `node3`, the command specifies a placement region constraint.

```
place_node -name node1 # let ECO Fitter decide placement
place_node -name node2 -location FF_X20_Y60_N17 # place node at specific
location
place_node -name node3 -location "X10 Y10 X20 Y20" # place node in region
place_node -name my_ff -location "X10 Y10 X10 Y10" # place flip-flop in region
```





### Arguments

*name* Name of the node.

*location* Device region coordinates (X1 Y1 X10 Y10) (X1 Y1)  
(FF\_X20\_Y60\_N17).

## 7.4.12. unplace\_node

### Description

Unplaces the node that you specify. `unplace_node` supports moving larger clouds of logic. To simplify the process of moving a larger cloud of logic, such as an entire ALM, you can first unplace all of the nodes. The Fitter does not perform placement legality checking until you re-place the final ALM cell.

### Usage

The following example unplaces a node with the name `ff1`:

```
unplace_node -name ff1
```

### Arguments

*name* Name of the node to unplace.

## 7.4.13. create\_wirelut

### Description

Creates and inserts a wire LUT node in the connection that you specify. The ECO Fitter places the new LUT and routes the modified connections automatically. Optionally, you can specify the `location` argument to specify a particular device region location constraint.

The name of the new node is hierarchical. Therefore, when creating node `a|b|c|d`, you must ensure that hierarchy `a|b|c` exists in the netlist. If the source or destination node lies under a partition, the new wire LUT inserts under that partition.

`create_wirelut` also supports adding connections from and to Hyper-Registers. In the case of Hyper-Registers, the command first disconnects the destination port, before making a new connection. You can run the `create_wirelut` command to specify a replacement signal source or destination.

If a port is shared among multiple atoms (for example, RAM), then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

## Usage

The following example creates the `my_wirelut` wire LUT, connects `my_wirelut` output to the D input port of `dest_node`, and connects the output of `src_output` to the input port of the wire LUT. Finally, the ECO Fitter places the new node within region (20, 20) to (40, 40) and routes automatically.

```
create_wirelut -name my_wirelut -from src_output -to dest_node \  
-port D -location "X20 Y20 X40 Y40"
```

## Arguments

*name* Name of the node.

*From* Source of the connection.

*To* Name of destination node.

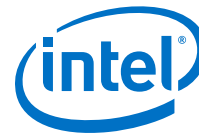
*Port* Input port name of destination node.

*location* Device region coordinates (X1 Y1 X10 Y10) (X1 Y1) (FF\_X20\_Y60\_N17).

## 7.5. ECO Command Limitations

The ECO commands have the following limitations due to connection dependencies within Intel FPGA devices.

- You cannot use ECO commands to modify dedicated connections.
- You cannot modify dedicated connections within a single ALM. This limitation applies to direct connections between LUT and flip-flop nodes.
- You can connect from or to a Hyper-Register. However, you cannot remove connections from or to a Hyper-Register because removing a connection from a Hyper-Register would leave the routing dangling. As an alternative, you can use `make_connection` to change a Hyper-Register connection immediately, without removing the previous connection first.
- Use of the `place_node` command with `location` arguments does not overwrite Partial Reconfiguration region constraints.
- If a LAB already has the maximum number of legal connections where a node is placed, the `place_node` or `make_connection` commands can fail, preventing the connection to the first placed node that cannot be legalized. You can then either move the original node to a different location, or move other nodes from the LAB to free up routing resources.
- The Fitter may fail to apply some I/O related ECO modifications, such as `modify_io_slew_rate`, `modify_io_current_strength`, and `modify_io_delay_chain`, if called using a command-line Tcl script or in interactive context. That is, any case that calls the `eco_load_design` command directly. To ensure all I/O modifications are applied successfully, use the standard ECO Tcl script approach this document describes.



The recommended order for creating and placing new LUTs is:

1. Create the node by using the `create_new_node` command.
2. Make connections to and from the node by using the `make_connection` command.
3. Update the lutmask by using the `modify_lutmask` command.
4. Place the node by using the `place_node` command.

This flow ensures that analysis includes all routing requirements when determining a legal placement for the new node. For example:

```
set lut_name new_lut
create_new_node -name $lut_name -type lut
make_connection -from input1 -to $lut_name -port DATAA
make_connection -from input2 -to $lut_name -port DATAB
make_connection -from $lut_name -to output_dest -port DATAD
modify_lutmask -to $lut_name -eqn {A&B}
place_node -name $lut_name -location "X80 Y80 X85 Y95"
```

**Note:** To minimize issues with name matching caused by escaped characters, it can be useful to surround entity names with `{ }` characters, instead of `" "`. This technique is particularly useful if entity names contain backslashes or any other special characters.

## 7.6. Interactive ECO Fitting

The `quartus_fit` executable supports ECO changes in an interactive shell through `quartus_fit -s`.

In an interactive context, the ECO Fitter legalizes the changes, when appropriate. For example, for `make_connection` changes immediately after node creation, the Fitter does not attempt to route the connections immediately; rather, the Fitter waits until after the placement of the node prior to routing.

### 7.6.1. `eco_load_design` and `eco_commit_design` Commands

#### Description

- `eco_load_design`—loads the final netlist in the ECO context.
- `eco_commit_design`—commits the ECO modified netlist to disk while running in interactive mode.

#### Usage

The following example shows an interactive ECO session to modify the `lut_x` lutmask for project `top`. If the ECO modifications are legal, the `eco_commit_design` command commits the final netlist.

```
$ quartus_fit -s
>> load_package eco
>> project_open top
>> eco_load_design
>> modify_lutmask -to lut_x -eqn B&C # some ECO changes
>> eco_commit_design
>> project_close
```

## 7.7. Using the ECO Compilation Flow Revision History

The following revision history applies to this chapter:

**Table 37. Document Revision History**

Document Version	Intel Quartus Prime Version	Changes
2020.09.28	20.3	<ul style="list-style-type: none"> <li>Revised chapter title to "Using the ECO Compilation Flow."</li> <li>Added descriptions of new <code>unplace_node</code> and <code>delete_node</code> commands.</li> <li>Described new support for placement of flip-flop nodes and exact locations in <code>place_node</code> command topic.</li> <li>Described new support for creation of flip-flop nodes in <code>create_new_node</code> command topic.</li> <li>Updated limitations in "ECO Command Limitations to remove obsolete limitations."</li> <li>Revised wording of introduction.</li> <li>Added report screenshots to "Viewing ECO Compilation Reports" topic.</li> </ul>
2020.05.08	20.1	<ul style="list-style-type: none"> <li>Added descriptions of new <code>create_new_node</code>, <code>place_node</code>, and <code>create_wirelut</code> commands.</li> <li>Referenced support for multi-node ECOs in <code>make_connection</code>, <code>remove_connection</code>, and <code>create_wirelut</code> command topics.</li> <li>Referenced Support for ECO connections to Hyper-Registers in the <code>make_connection</code> topic.</li> <li>Described updates to ECO reporting in "Viewing ECO Compilation Reports."</li> <li>Updated limitations in "ECO Command Limitations."</li> <li>Added ECO Command Quick Reference</li> </ul>
2019.09.30	19.3.0	<ul style="list-style-type: none"> <li>Added information about <code>tieoff</code> option for <code>make_connection</code> command.</li> <li>Added support for <code>modify_io_slew_rate</code> command.</li> <li>Added support for <code>modify_io_current_strength</code> command.</li> <li>Added support for <code>modify_io_delay_chain</code> command.</li> <li>Added "Viewing ECO Compilation Reports" topic.</li> <li>Added information about <code>num</code> option for <code>modify_lutmask</code> command.</li> <li>Mentioned RTL Viewer for locating node names.</li> <li>Added device support note.</li> </ul>
2019.07.01	19.2.0	<ul style="list-style-type: none"> <li>First release of chapter.</li> </ul>



## 8. Intel Quartus Prime Pro Edition Design Optimization User Guide Archives

---

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
20.1	<a href="#">Intel Quartus Prime Pro Edition User Guide: Design Optimization</a>
19.3	<a href="#">Intel Quartus Prime Pro Edition User Guide: Design Optimization</a>
19.1	<a href="#">Intel Quartus Prime Pro Edition User Guide: Design Optimization</a>
18.1	<a href="#">Intel Quartus Prime Pro Edition User Guide: Design Optimization</a>
18.0	<a href="#">Design Optimization User Guide Intel Quartus Prime Pro Edition</a>



## A. Intel Quartus Prime Pro Edition User Guides

---

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Pro Edition FPGA design flow.

### Related Information

- [Intel Quartus Prime Pro Edition User Guide: Getting Started](#)  
Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- [Intel Quartus Prime Pro Edition User Guide: Platform Designer](#)  
Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- [Intel Quartus Prime Pro Edition User Guide: Design Recommendations](#)  
Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.
- [Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)  
Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- [Intel Quartus Prime Pro Edition User Guide: Design Optimization](#)  
Describes Intel Quartus Prime Pro Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).
- [Intel Quartus Prime Pro Edition User Guide: Programmer](#)  
Describes operation of the Intel Quartus Prime Pro Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.
- [Intel Quartus Prime Pro Edition User Guide: Block-Based Design](#)  
Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.



- [Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration](#)  
Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Simulation](#)  
Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec\*, Cadence\*, Mentor Graphics\*, and Synopsys that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis](#)  
Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics\*, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools](#)  
Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin\*.
- [Intel Quartus Prime Pro Edition User Guide: Debug Tools](#)  
Describes a portfolio of Intel Quartus Prime Pro Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or “tapping”) signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- [Intel Quartus Prime Pro Edition User Guide: Timing Analyzer](#)  
Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- [Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization](#)  
Describes the Intel Quartus Prime Pro Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- [Intel Quartus Prime Pro Edition User Guide: Design Constraints](#)  
Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- [Intel Quartus Prime Pro Edition User Guide: PCB Design Tools](#)  
Describes support for optional third-party PCB design tools by Mentor Graphics\* and Cadence\*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- [Intel Quartus Prime Pro Edition User Guide: Scripting](#)  
Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.