



Intel[®] Quartus[®] Prime Standard Edition User Guide

Design Optimization

Updated for Intel[®] Quartus[®] Prime Design Suite: **18.1**



[Subscribe](#)

[Send Feedback](#)

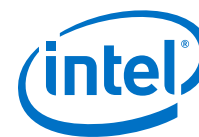
UG-20177 | 2018.11.12

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. Guidelines for I/O Assignments.....7
 - 1.2.2. Guidelines for Time Constraints..... 7
 - 1.2.3. Partitions and Floorplan Assignments for Incremental Compilation.....8
 - 1.3. Trade-Offs and Limitations..... 8
 - 1.3.1. Preserving Results and Enabling Teamwork..... 9
 - 1.3.2. Reducing Area.....9
 - 1.3.3. Reducing Critical Path Delay..... 9
 - 1.3.4. Reducing Power Consumption..... 10
 - 1.3.5. Reducing Runtime..... 10
 - 1.4. Intel Quartus Prime Software Tools for Design Optimization..... 11
 - 1.4.1. Design Visualization Tools..... 11
 - 1.4.2. Advisors..... 11
 - 1.4.3. Design Exploration..... 12
 - 1.5. Design Space Explorer II..... 12
 - 1.5.1. How DSE II Works..... 13
 - 1.5.2. Performing a Design Exploration with the DSE II Utility..... 15
 - 1.6. Design Optimization Overview Revision History..... 16
- 2. Optimizing the Design Netlist..... 17**
 - 2.1. When to Use the Netlist Viewers: Analyzing Design Problems 17
 - 2.2. Intel Quartus Prime Design Flow with the Netlist Viewers..... 18
 - 2.3. RTL Viewer Overview..... 19
 - 2.3.1. Maximizing Readability in RTL Viewer..... 20
 - 2.3.2. Running the RTL Viewer..... 21
 - 2.4. State Machine Viewer Overview..... 21
 - 2.5. Technology Map Viewer Overview..... 21
 - 2.6. Netlist Viewer User Interface..... 22
 - 2.6.1. Netlist Navigator Pane..... 25
 - 2.6.2. Properties Pane..... 25
 - 2.6.3. Netlist Viewers Find Pane..... 27
 - 2.7. Schematic View..... 27
 - 2.7.1. Display Schematics in Multiple Tabbed View..... 27
 - 2.7.2. Schematic Symbols..... 28
 - 2.7.3. Select Items in the Schematic View..... 32
 - 2.7.4. Shortcut Menu Commands in the Schematic View..... 33
 - 2.7.5. Filtering in the Schematic View..... 33
 - 2.7.6. View Contents of Nodes in the Schematic View..... 34
 - 2.7.7. Moving Nodes in the Schematic View..... 35
 - 2.7.8. View LUT Representations in the Technology Map Viewer..... 36
 - 2.7.9. Zoom Controls..... 36
 - 2.7.10. Navigating with the Bird's Eye View..... 37
 - 2.7.11. Partition the Schematic into Pages..... 37
 - 2.7.12. Follow Nets Across Schematic Pages..... 38



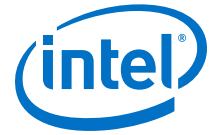
2.8. State Machine Viewer.....	38
2.8.1. State Diagram View.....	39
2.8.2. State Transition Table.....	39
2.8.3. State Encoding Table.....	39
2.8.4. Switch Between State Machines.....	40
2.9. Cross-Probing to a Source Design File and Other Intel Quartus Prime Windows.....	40
2.10. Cross-Probing to the Netlist Viewers from Other Intel Quartus Prime Windows.....	41
2.11. Viewing a Timing Path.....	41
2.12. Optimizing the Design Netlist Revision History.....	42
3. Timing Closure and Optimization.....	44
3.1. Optimize Multi Corner Timing.....	44
3.2. Critical Paths.....	44
3.2.1. Viewing Critical Paths.....	45
3.3. Design Evaluation for Timing Closure.....	45
3.3.1. Review Compilation Results.....	45
3.3.2. Review Details of Timing Paths.....	56
3.3.3. Adjusting and Recompiling.....	59
3.4. Design Analysis.....	60
3.4.1. Ignored Timing Constraints.....	60
3.4.2. I/O Timing.....	60
3.4.3. Register-to-Register Timing Analysis.....	61
3.5. Timing Optimization.....	65
3.5.1. Displaying Timing Closure Recommendations for Failing Paths.....	65
3.5.2. Timing Optimization Advisor.....	66
3.5.3. Optional Fitter Settings.....	67
3.5.4. I/O Timing Optimization Techniques.....	69
3.5.5. Register-to-Register Timing Optimization Techniques.....	73
3.5.6. Logic Lock (Standard) Assignments.....	79
3.5.7. Location Assignments.....	81
3.5.8. Metastability Analysis and Optimization Techniques.....	81
3.6. Periphery to Core Register Placement and Routing Optimization	82
3.6.1. Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box.....	83
3.6.2. Setting Periphery to Core Optimizations in the Assignment Editor.....	83
3.6.3. Viewing Periphery to Core Optimizations in the Fitter Report.....	84
3.7. Scripting Support.....	85
3.7.1. Initial Compilation Settings.....	85
3.7.2. I/O Timing Optimization Techniques	86
3.7.3. Register-to-Register Timing Optimization Techniques.....	86
3.8. Timing Closure and Optimization Revision History.....	87
4. Area Optimization.....	90
4.1. Resource Utilization Information.....	90
4.1.1. Flow Summary Report.....	90
4.1.2. Fitter Reports.....	90
4.1.3. Analysis and Synthesis Reports.....	91
4.1.4. Compilation Messages.....	91
4.2. Optimizing Resource Utilization.....	91
4.2.1. Using the Resource Optimization Advisor.....	92
4.2.2. Resource Utilization Issues Overview.....	92
4.2.3. I/O Pin Utilization or Placement.....	92



- 4.2.4. Logic Utilization or Placement..... 93
- 4.2.5. Routing..... 97
- 4.3. Scripting Support.....99
 - 4.3.1. Initial Compilation Settings.....100
 - 4.3.2. Resource Utilization Optimization Techniques..... 101
- 4.4. Area Optimization Revision History..... 102
- 5. Analyzing and Optimizing the Design Floorplan..... 103**
 - 5.1. Design Floorplan Analysis in the Chip Planner.....103
 - 5.1.1. Starting the Chip Planner..... 104
 - 5.1.2. Chip Planner GUI Components..... 104
 - 5.1.3. Viewing Architecture-Specific Design Information..... 106
 - 5.1.4. Viewing Available Clock Networks in the Device..... 107
 - 5.1.5. Viewing I/O Banks.....108
 - 5.1.6. Viewing High-Speed Serial Interfaces (HSSI)..... 108
 - 5.1.7. Viewing the Source and Destination of Placed Nodes..... 108
 - 5.1.8. Viewing Fan-In and Fan-Out Connections of Placed Resources..... 109
 - 5.1.9. Generating Immediate Fan-In and Fan-Out Connections..... 110
 - 5.1.10. Exploring Paths in the Chip Planner..... 110
 - 5.1.11. Viewing Assignments in the Chip Planner..... 112
 - 5.1.12. Viewing High-Speed and Low-Power Tiles in the Chip Planner..... 113
 - 5.1.13. Viewing Design Partition Placement..... 114
 - 5.2. Logic Lock (Standard) Regions..... 114
 - 5.2.1. Attributes of a Logic Lock (Standard) Region..... 115
 - 5.2.2. Creating Logic Lock (Standard) Regions..... 115
 - 5.2.3. Customizing the Shape of Logic Lock Regions..... 118
 - 5.2.4. Placing Logic Lock (Standard) Regions..... 119
 - 5.2.5. Placing Device Resources into Logic Lock (Standard) Regions..... 120
 - 5.2.6. Hierarchical (Parent and Child) Logic Lock (Standard) Regions..... 123
 - 5.2.7. Additional Intel Quartus Prime Logic Lock (Standard) Design Features..... 124
 - 5.2.8. Logic Lock (Standard) Regions Window..... 124
 - 5.3. Using Logic Lock (Standard) Regions in the Chip Planner..... 125
 - 5.3.1. Viewing Connections Between Logic Lock (Standard) Regions in the Chip Planner..... 125
 - 5.3.2. Using Logic Lock (Standard) Regions with the Design Partition Planner..... 126
 - 5.4. Scripting Support..... 126
 - 5.4.1. Initializing and Uninitializing a Logic Lock (Standard) Region..... 126
 - 5.4.2. Creating or Modifying Logic Lock (Standard) Regions..... 126
 - 5.4.3. Obtaining Logic Lock (Standard) Region Properties..... 127
 - 5.4.4. Assigning Logic Lock (Standard) Region Content..... 127
 - 5.4.5. Save a Node-Level Netlist for the Entire Design into a Persistent Source File.. 127
 - 5.4.6. Setting Logic Lock (Standard) Assignment Priority..... 128
 - 5.4.7. Assigning Virtual Pins with a Tcl command..... 128
 - 5.5. Analyzing and Optimizing the Design Floorplan Revision History..... 128
- 6. Netlist Optimizations and Physical Synthesis..... 131**
 - 6.1. Physical Synthesis Optimizations..... 131
 - 6.1.1. Enabling Physical Synthesis Optimization..... 132
 - 6.1.2. Physical Synthesis Options..... 132
 - 6.1.3. Perform Register Retiming for Performance..... 133
 - 6.1.4. Preventing Register Movement During Retiming..... 134
 - 6.2. Applying Netlist Optimizations..... 135



6.2.1. WYSIWYG Primitive Resynthesis.....	136
6.2.2. Saving a Node-Level Netlist.....	137
6.3. Viewing Synthesis and Netlist Optimization Reports.....	138
6.4. Scripting Support.....	138
6.4.1. Synthesis Netlist Optimizations.....	139
6.4.2. Physical Synthesis Optimizations.....	139
6.4.3. Back-Annotating Assignments.....	140
6.5. Netlist Optimizations and Physical Synthesis Revision History.....	140
7. Engineering Change Orders with the Chip Planner.....	142
7.1. Engineering Change Orders.....	142
7.1.1. Performance Preservation.....	143
7.1.2. Compilation Time.....	143
7.1.3. Verification.....	143
7.1.4. Change Modification Record.....	144
7.2. ECO Design Flow.....	144
7.3. The Chip Planner Overview.....	146
7.3.1. Opening the Chip Planner.....	146
7.3.2. The Chip Planner Tasks and Layers.....	147
7.4. Performing ECOs with the Chip Planner (Floorplan View).....	147
7.4.1. Creating, Deleting, and Moving Atoms.....	147
7.4.2. Check and Save Netlist Changes.....	147
7.5. Performing ECOs in the Resource Property Editor.....	147
7.5.1. Logic Elements.....	148
7.5.2. Adaptive Logic Modules.....	150
7.5.3. FPGA I/O Elements.....	151
7.5.4. FPGA RAM Blocks.....	155
7.5.5. FPGA DSP Blocks.....	156
7.6. Change Manager.....	157
7.6.1. Complex Changes in the Change Manager.....	158
7.6.2. Managing Signal Probe Signals.....	158
7.6.3. Exporting Changes.....	158
7.7. Scripting Support.....	158
7.8. Common ECO Applications.....	158
7.8.1. Adjust the Drive Strength of an I/O with the Chip Planner.....	159
7.8.2. Modify the PLL Properties With the Chip Planner.....	160
7.8.3. PLL Properties.....	161
7.8.4. Modify the Connectivity between Resource Atoms.....	163
7.9. Post ECO Steps.....	164
7.10. Engineering Change Orders with the Chip Planner Revision History.....	164
A. Intel Quartus Prime Standard Edition User Guides.....	166



1. Design Optimization Overview

In a typical design flow, the early stages of development concentrate on meeting timing, area and power goals. Once the design meets those goals, the efforts focus on improving performance. This chapter introduces techniques and tools in the Intel® Quartus® Prime software that you can use to achieve the highest design performance.

Optimization of a FPGA design requires a multi-dimensional approach that meets the design goals while reducing area, critical path delay, power consumption, and runtime. The Intel Quartus Prime software includes advisors to address each of these issues. By implementing the advisor's suggestions, you can reduce the time spent on design iterations.

Related Information

[Intel Quartus Prime Design Software - Support Center](#)

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.

Related Information

[Guaranteeing Silicon Performance with FPGA Timing Models](#)
Intel FPGA White Paper (PDF)

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 settings and options provide the best trade-off between compilation time, resource utilization, and timing performance. Before compiling a design in the Intel Quartus Prime software, consider the following guidelines.

1.2.1. Guidelines for I/O Assignments

In a FPGA design, I/O standards and drive strengths affect I/O timing.

- 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 Standard Edition User Guide: Design Constraints

1.2.2. Guidelines for Time Constraints

For best results, use real time requirements. Applying more demanding timing requirements than the design needs can cause the Compiler to trade off by increasing resource usage, power utilization, or compilation time.

Comprehensive timing requirement settings achieve the best results for the following reasons:

- Correct timing assignments enable the software to work hardest to optimize the performance of the timing-critical parts of the design and make trade-offs for performance. This optimization can also save area or power utilization in non-critical parts of the design.
- If enabled, the Intel Quartus Prime software performs physical synthesis optimizations based on timing requirements.
- Depending on the **Fitter Effort** setting, the Fitter can reduce runtime if the design meets the timing requirements.

The Intel Quartus Prime Timing Analyzer determines if the design implementation meets the timing requirement. The Compilation Report shows whether the design meets the timing requirements, while the timing analysis reporting commands provide detailed information about the timing paths.

Related Information

- [Timing Closure and Optimization](#) on page 44
- [Using the Intel Quartus Prime Timing Analyzer](#)
In Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
- [Intel Quartus Prime Timing Analyzer Cookbook](#)



- [Advanced Settings \(Fitter\)](#)
In *Intel Quartus Prime Help*

1.2.3. Partitions and Floorplan Assignments for Incremental Compilation

The Intel Quartus Prime incremental compilation feature enables hierarchical and team-based design flows in which you can compile parts of your design while other parts of your design remain unchanged. You can also Import parts of your design from separate Intel Quartus Prime projects.

Using incremental compilation for your design with good design partitioning methodology helps to achieve timing closure. Creating design partitions on some of the major blocks in your design and assigning them to Logic Lock (Standard)[™] regions, reduces Fitter time and improves the quality and repeatability of the results. Logic Lock (Standard) regions are flexible, reusable floorplan location constraints that help you place logic on the target device. When you assign entity instances or nodes to a Logic Lock (Standard) region, you direct the Fitter to place those entity instances or nodes inside the region during fitting.

Using incremental compilation helps you achieve timing closure block by block and preserve the timing performance between iterations, which aid in achieving timing closure for the entire design. Incremental compilation may also help reduce compilation times.

Note: If you plan to use incremental compilation, you must create a floorplan for your design. If you are not using incremental compilation, creating a floorplan is optional.

Related Information

- [Reducing Compilation Time](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*
- [Best Practices for Incremental Compilation Partitions and Floorplan Assignments](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*

1.3. Trade-Offs and Limitations

Many optimization goals can conflict with one another, so you might need to resolve conflicting goals.

Table 1. Examples of Trade offs in Design Optimization

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 severe limit design feasibility as far as 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. Preserving Results and Enabling Teamwork

For some Intel Quartus Prime Fitter algorithms, small changes to the design can have a large impact on the final result. For example, a critical path delay can change by 10% or more because of seemingly insignificant changes. If you are close to meeting your timing objectives, you can use the Fitter algorithm to your advantage by changing the fitter seed, which changes the pseudo-random result of the Fitter.

Conversely, if you cannot meet timing on a portion of your design, you can partition that portion and prevent it from recompiling if an unrelated part of the design is changed. This feature, known as incremental compilation, can reduce the Fitter runtimes by up to 70% if the design is partitioned, such that only small portions require recompilation at any one time.

When you use incremental compilation, you can apply design optimization options to individual design partitions and preserve performance in other partitions by leaving them untouched. Many optimization techniques often result in longer compilation times, but by applying them only on specific partitions, you can reduce this impact and complete iterations more quickly.

In addition, by physically floorplanning your partitions with Logic Lock (Standard) regions, you can enable team-based flows and allow multiple people to work on different portions of the design.

Related Information

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Designs](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*

1.3.2. Reducing Area

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 90
- [Netlist Optimizations and Physical Synthesis](#) on page 131

1.3.3. Reducing Critical Path Delay

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, realistic constraints are 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.

If you use incremental compilation, you can help resolve inter-partition timing requirements by locking down results, one partition at a time, or by guiding the placement of the partitions with Logic Lock (Standard) regions. You might improve the timing on such paths by placing the partitions optimally to reduce the length of critical paths. Once the inter-partition timing requirements are met, use incremental compilation to preserve the results and work on partitions that have not met timing requirements.

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

[Critical Paths](#) on page 44

1.3.4. Reducing Power Consumption

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.

Related Information

[Power Optimization](#)

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

1.3.5. Reducing Runtime

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 based on your project requirements.

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

You can also reduce compilation time with your iterations by using incremental compilation. Use incremental compilation when you want to change parts of your design, while keeping most of the remaining logic unchanged.

Related Information

[Reducing Compilation Time](#)

In *Intel Quartus Prime Standard Edition User Guide: Compiler*



1.4. Intel Quartus Prime Software Tools for Design Optimization

The Intel Quartus Prime software offers tools that you can use to optimize a design.

1.4.1. Design Visualization Tools

The Intel Quartus Prime software provides tools that display graphical representations of a design.

Table 2. 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.
Design Partition Planner	Displays designs at partition and entity levels, and can display connectivity between entities
Design Partition Planner and Chip Planner (With incremental compilation)	Allow you to partition and layout the design at a higher level.
Chip Planner	Allow you to make floorplan assignments, implement engineering change orders (ECOs), perform power analysis, and visualize critical paths and routing congestion.

Related Information

- [Design Floorplan Analysis in the Chip Planner](#) on page 103
- [Optimizing the Design Netlist](#) on page 17
- [RTL Viewer Overview](#) on page 19

1.4.2. Advisors

The Intel Quartus Prime software includes several advisors to help you optimize your design and reduce compilation time.

The advisors provide recommendations based on the project settings and design constraints. Those recommendations can help you to fit the project, meet timing or power requirements, or improve the design performance.

The advisors organize the recommendations from general to specific. Where applicable, the categories are divided into of stages presented by complexity.

The advisors are:

- Resource Optimization Advisor
- Timing Optimization Advisor
- Power Optimization Advisor
- Compilation Time Advisor
- Pin Optimization Advisor
- Incremental Compilation Advisor



Related Information

- [Compilation Time Advisor](#)
In *Intel Quartus Prime Standard Edition User Guide: Compiler*
- [Advisors in the Intel Quartus Prime Software](#)
In *Intel Quartus Prime Help*
- [Timing Optimization Advisor](#) on page 66
- [Power Optimization Advisor](#)
In *Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization*

1.4.3. Design Exploration

The Design Space Explorer II tool (DSE II) provides an easy and efficient way for you to run experiments on your design settings. You can run a single compilation locally on your PC or remotely using compute farm resources.

Related Information

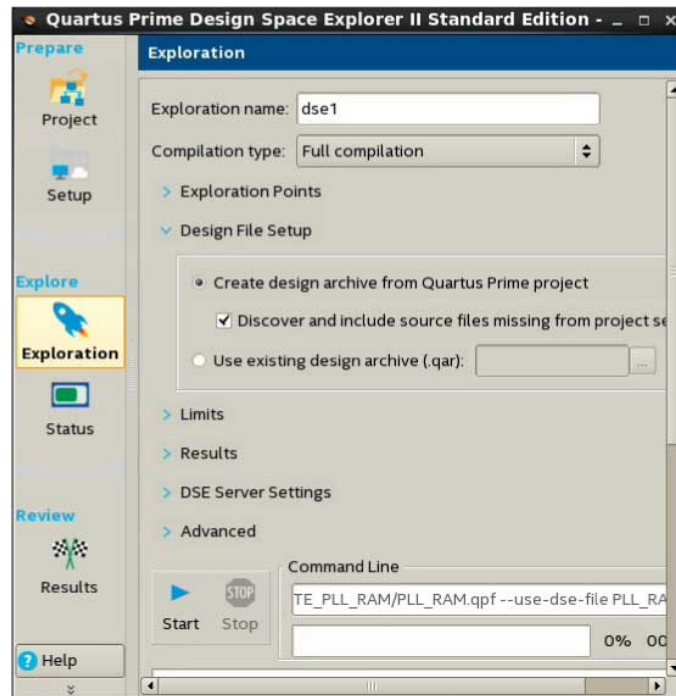
[Design Space Explorer II](#) on page 12

1.5. 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.

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 1. Design Space Explorer II User Interface



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

- [Design Space Explorer II Tool](#)
In *Intel Quartus Prime Help*
- [Using Design Space Explorer](#)
21 Minute Online Course

1.5.1. How DSE II Works

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.

Related Information

[Design Space Explorer II for Power-Driven Optimization](#)

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

1.5.1.1. Use of 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.

Related Information

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

1.5.1.2. 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 explorations 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*

1.5.1.3. 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*

1.5.2. Performing a Design Exploration with the DSE II Utility

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**.

Related Information

- [Design Space Explorer II Tool](#)
In *Intel Quartus Prime Help*
- [Using Design Space Explorer](#)
21 Minute Online Course



1.6. Design Optimization Overview Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.
2018.05.07	18.0.0	<ul style="list-style-type: none">General topic reorganization.
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">Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Minor changes for HardCopy.
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.

Related Information

[Documentation Archive](#)

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

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, State Machine 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 18
- [State Machine Viewer Overview](#) on page 21
- [RTL Viewer Overview](#) on page 19
- [Technology Map Viewer Overview](#) on page 21
- [Filtering in the Schematic View](#) on page 33
- [Viewing a Timing Path](#) on page 41

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, State Machine 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 and State Machine 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. You can also view state machine transitions and transition equations with the State Machine Viewer. 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 18
- [State Machine Viewer Overview](#) on page 21
- [RTL Viewer Overview](#) on page 19
- [Technology Map Viewer Overview](#) on page 21

2.2. Intel Quartus Prime Design Flow with the Netlist Viewers

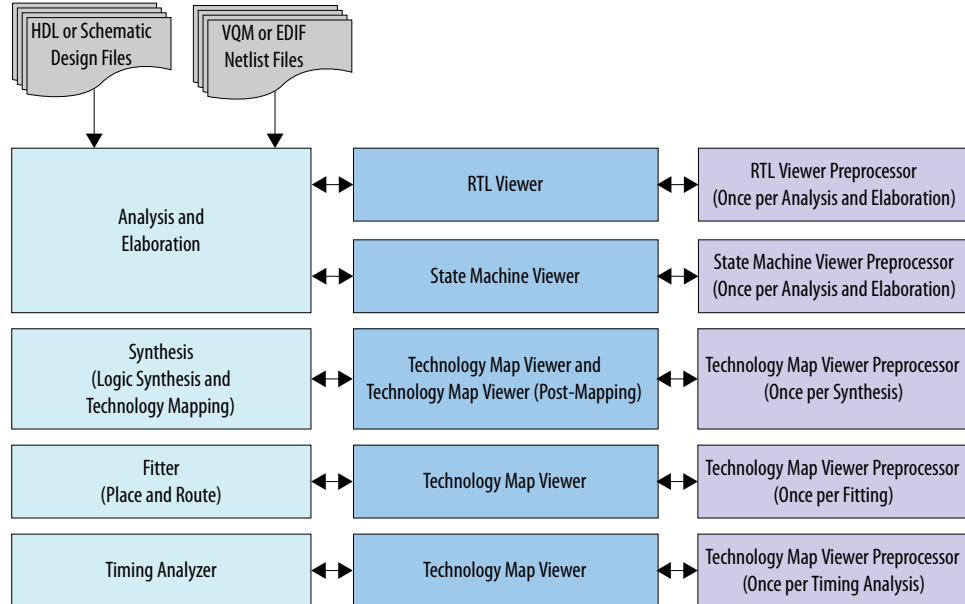
When you first open one of the Netlist Viewers after compiling the design, a preprocessor stage runs automatically before the Netlist Viewer opens.

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 2. 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 or State Machine 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 integrated 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 Intel Quartus Prime integrated 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 a 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 22

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.
- Converts state machine logic into a state diagram, state transition table, and state encoding table, which appear in the State Machine Viewer.

Related Information

[State Machine Viewer Overview](#) on page 21



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. State Machine Viewer Overview

The State Machine Viewer presents a high-level view of finite state machines in your design. The State Machine Viewer provides a graphical representation of the states and their related transitions, as well as a state transition table that displays the condition equation for each of the state transitions, and encoding information for each state.

To run the State Machine Viewer, on the Tools menu, point to **Netlist Viewers** and click **State Machine Viewer**. To open the State Machine Viewer for a particular state machine, double-click the state machine instance in the RTL Viewer.

2.5. 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 a 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 41
- [View Contents of Nodes in the Schematic View](#) on page 34
- [Netlist Viewer User Interface](#) on page 22

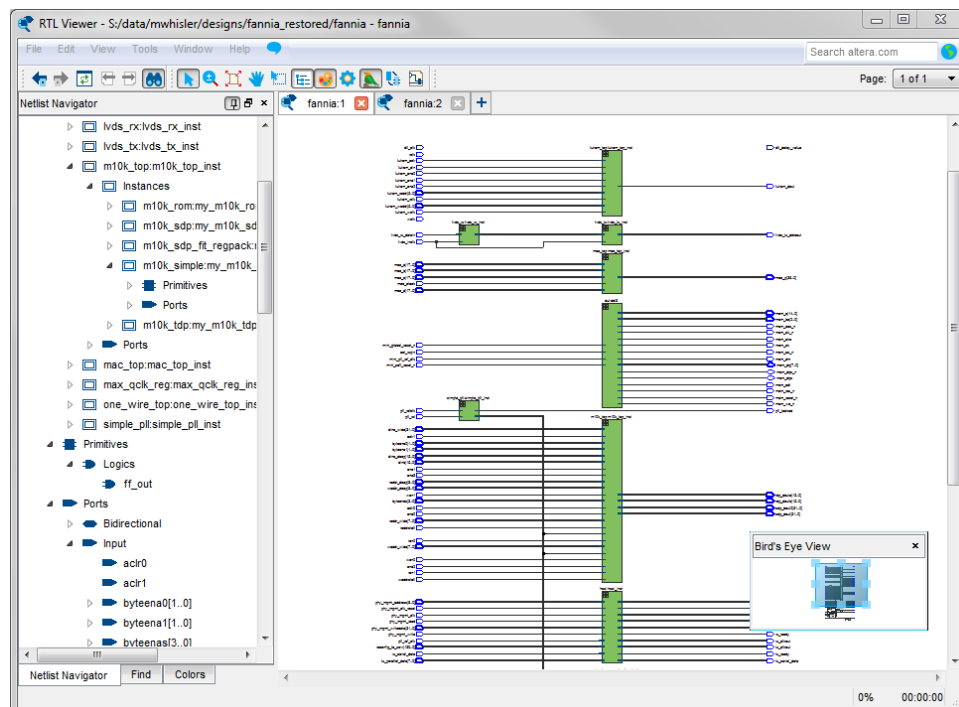
2.6. Netlist Viewer User Interface

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 3. RTL Viewer



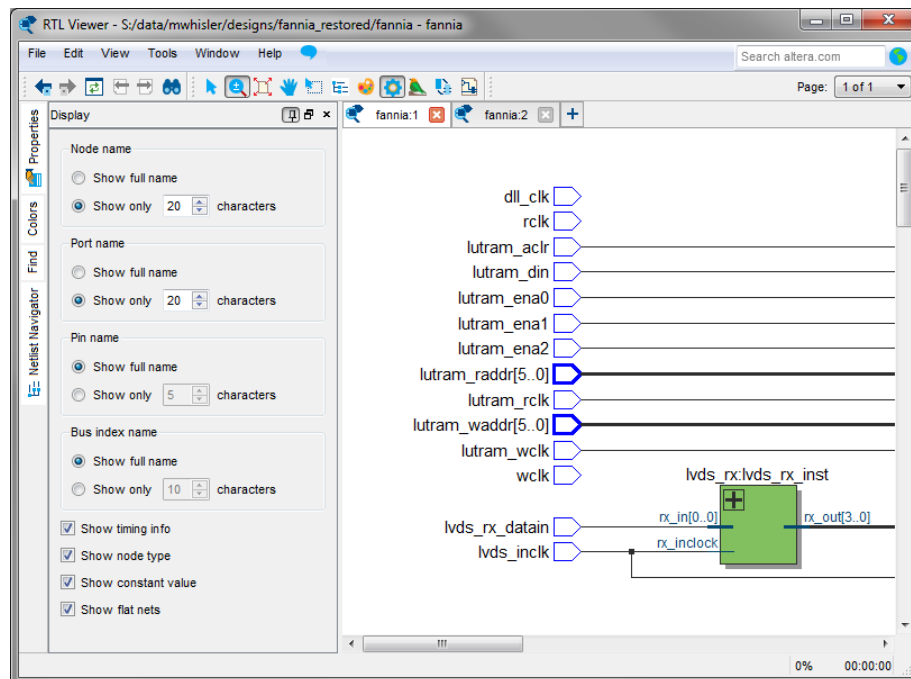


Netlist Viewers also contain a toolbar that provides tools to use in the schematic view.

- 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 4. 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), one Technology Map Viewer (Post-Mapping), and one State Machine Viewer 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 19
- [Technology Map Viewer Overview](#) on page 21
- [Netlist Navigator Pane](#) on page 25
- [Netlist Viewers Find Pane](#) on page 27
- [Properties Pane](#) on page 25

2.6.1. 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 3. Netlist Navigator Pane Elements

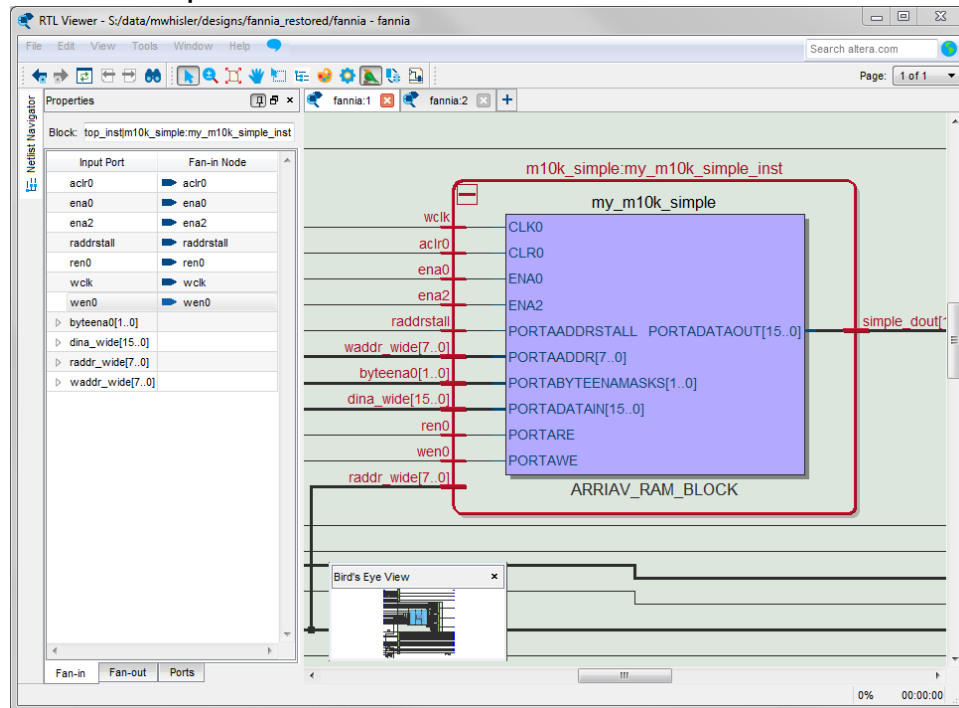
Elements	Description
Instances	Modules or instances in the design that can be expanded to lower hierarchy levels.
State Machines	State machine instances in the design that can be viewed in the State Machine Viewer.
Primitives	Low-level nodes that cannot be expanded to any lower hierarchy level. These primitives include: <ul style="list-style-type: none">• Registers and gates that you can view in the RTL Viewer when using Intel Quartus Prime integrated synthesis.• Logic cell atoms in the Technology Map Viewer or in the RTL Viewer when using a VQM or EDIF from third-party synthesis software 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.
Ports	The I/O ports in the current level of hierarchy. <ul style="list-style-type: none">• 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.• When a pin represents a bus or an array of pins, expand the pin entry in the list view to see individual pin names.

2.6.2. Properties Pane

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

Figure 5. 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 4. 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.



2.6.3. 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.

2.7. 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.

2.7.1. 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

2.7.2. 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 5. 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.</p> <p>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, CARRY, 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"> • DFFEA (data flipflop with enable and asynchronous load) primitive with additional <code>ALOAD</code> asynchronous load and <code>ADATA</code> data signals • DFFEAS (data flipflop with enable and synchronous and asynchronous load), which has <code>ASDATA</code> as the secondary data port
<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...

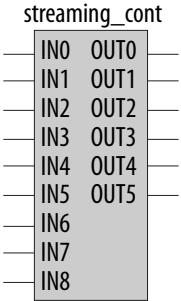
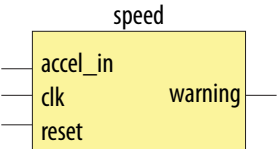
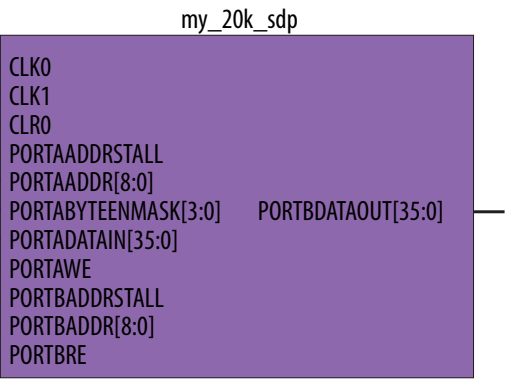
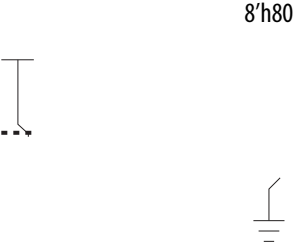
Symbol	Description
	
<p>State Machine Instance</p> 	<p>A finite state machine instance in the design.</p>
<p>RAM</p> 	<p>A synchronous memory instance with registered inputs and optionally registered outputs. The symbol shows the device family and the type of memory block. This figure shows a true dual-port 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 6. Symbol Available Only in the State Machine Viewer

The following table lists and describes the symbol open only in the State Machine Viewer.


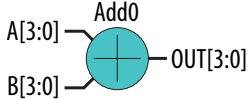
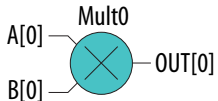
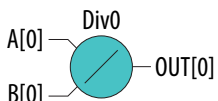
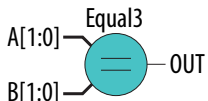
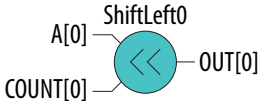
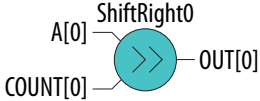
Symbol	Description
	The node representing a state in a finite state machine. State transitions are indicated with arcs between state nodes. The double circle border indicates the state connects to logic outside the state machine, and a single circle border indicates the state node does not feed outside logic.

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
	An adder operator: $OUT = A + B$
	A multiplier operator: $OUT = A \times B$
	A divider operator: $OUT = A / B$
	Equals
	A left shift operator: $OUT = (A \ll COUNT)$
	A right shift operator: $OUT = (A \gg COUNT)$

continued...

Symbol	Description
	<p>A modulo operator: $OUT = (A \% B)$</p>
	<p>A less than comparator: $OUT = (A < B : A > B)$</p>
	<p>A multiplexer: $OUT = DATA [SEL]$ The data range size is $2^{\text{sel range size}}$</p>
	<p>A selector: A multiplexer with one-hot select input and more than two input signals</p>
	<p>A binary number decoder: $OUT = (\text{binary_number}(IN) == x)$ for $x = 0$ to $x = 2^{(n+1)} - 1$</p>

Related Information

- [Partition the Schematic into Pages](#) on page 37
- [Follow Nets Across Schematic Pages](#) on page 38

2.7.3. 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 25

2.7.4. 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.

2.7.5. 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, ports of a node, or states in a state machine 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.

When filtering in a state diagram in the State Machine Viewer, sources and destinations refer to the previous and next transition states or paths between transition states in the state diagram. The transition table and encoding table also reflect the filtering.

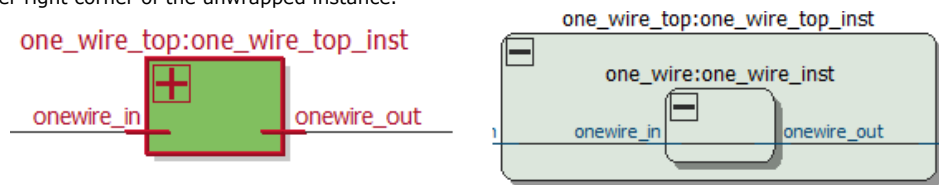
2.7.6. View Contents of Nodes in the Schematic View

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 6. 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 7. 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.

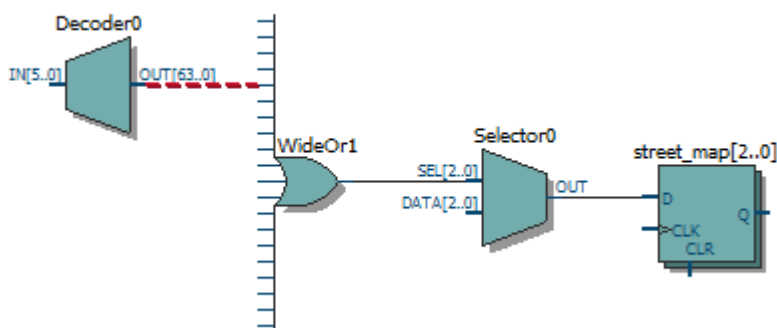


Figure 8. Display Nets Across Hierarchies

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.

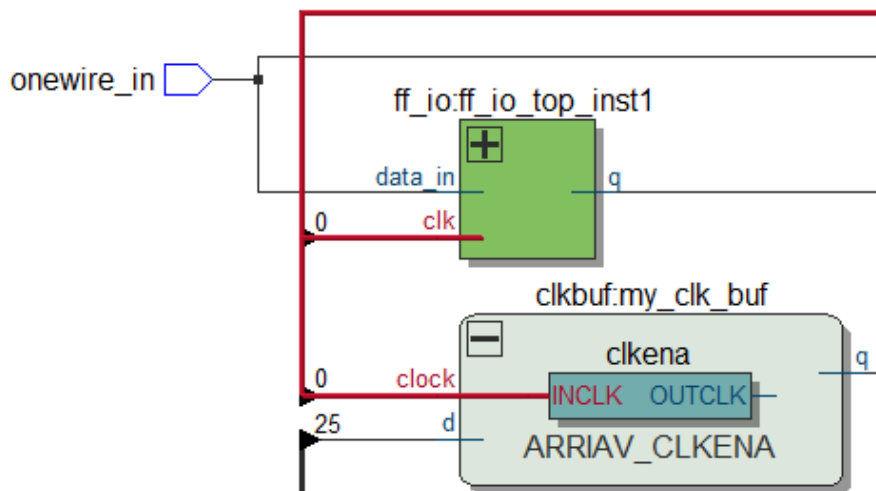
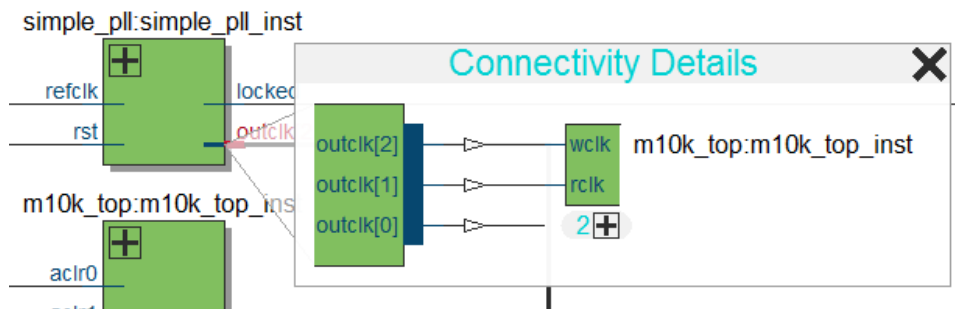


Figure 9. Show Connectivity Details

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



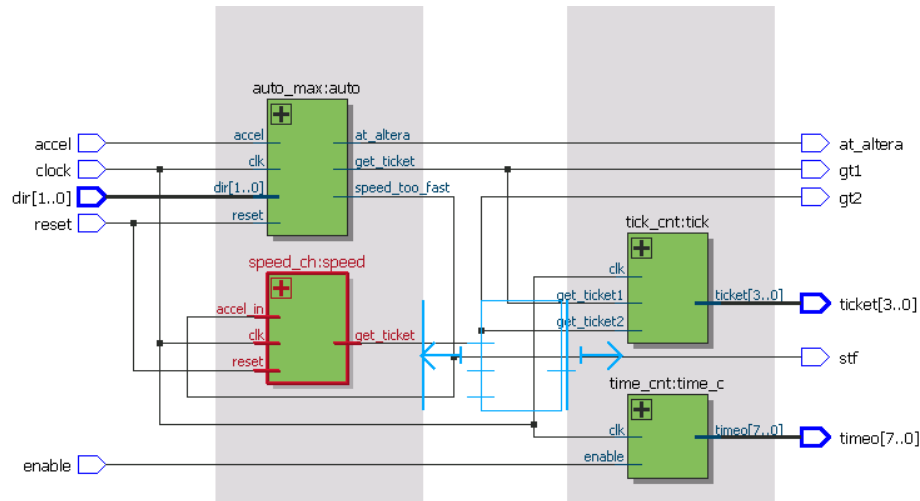
You can double-click objects in the **Connectivity Details** window to navigate to them quickly. If the plus symbol appears, you can further unwrap objects in the view. This can be very useful when tracing a signal in a complex netlist.

2.7.7. 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 10. Legal Placement when Moving Nodes



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

2.7.8. 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 25

2.7.9. 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 33

2.7.10. 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.

2.7.11. 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 22

2.7.12. 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 28

2.8. State Machine Viewer

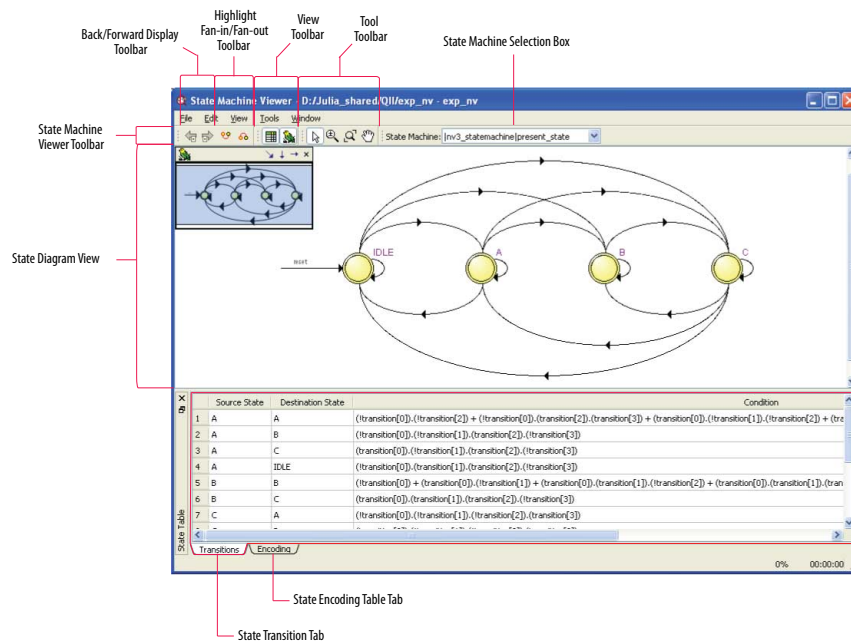
The State Machine Viewer displays a graphical representation of the state machines in your design.

You can open the State Machine Viewer in any of the following ways:

- On the Tools menu, point to **Netlist Viewers** and click **State Machine Viewer**.
- Double-click a state machine instance in the RTL Viewer

Figure 11. The State Machine Viewer

The following figure shows an example of the State Machine Viewer for a simple state machine and lists the components of the viewer.





2.8.1. State Diagram View

The state diagram view appears at the top of the State Machine Viewer. It contains a diagram of the states and state transitions.

The nodes that represent each state are arranged horizontally in the state diagram view with the initial state (the state node that receives the reset signal) in the left-most position. Nodes that connect to logic outside of the state machine instance are represented by a double circle. The state transition is represented by an arc with an arrow pointing in the direction of the transition.

When you select a node in the state diagram view, and turn on the **Highlight Fan-in** or **Highlight Fan-out** command from the View menu or the State Machine Viewer toolbar, the respective fan-in or fan-out transitions from the node are highlighted in red.

Note: An encrypted block with a state machine displays encoding information in the state encoding table, but does not display a state transition diagram or table.

2.8.2. State Transition Table

The state transition table on the **Transitions** tab at the bottom of the State Machine Viewer displays the condition equation for each state transition.

Each row in the table represents a transition (each arc in the state diagram view). The table has the following columns:

- **Source State**—the name of the source state for the transition
- **Destination State**—the name of the destination state for the transition
- **Condition**—the condition equation that causes the transition from source state to destination state

To see all of the transitions to and from each state name, click the appropriate column heading to sort on that column.

The text in each column is left-aligned by default; to change the alignment and to make it easier to see the relevant part of the text, right-click the column and click **Align Right**. To revert to left alignment, click **Align Left**.

Click in any cell in the table to select it. To select all cells, right-click in the cell and click **Select All**; or, on the Edit menu, click **Select All**. To copy selected cells to the clipboard, right-click the cells and click **Copy Table**; or, on the Edit menu, point to **Copy** and click **Copy Table**. You can paste the table into any text editor as tab-separated columns.

2.8.3. State Encoding Table

The state encoding table on the **Encoding** tab at the bottom of the State Machine Viewer displays encoding information for each state transition.

To view state encoding information in the State Machine Viewer, you must synthesize your design with the **Start Analysis & Synthesis** command. If you have only elaborated your design with the **Start Analysis & Elaboration** command, the encoding information is not displayed.

2.8.3.1. Select Items in the State Machine Viewer

You can select and highlight each state node and transition in the State Machine Viewer. To select a state transition, click the arc that represents the transition.

When you select a node or transition arc in the state diagram view, the matching state node or equation conditions in the state transition table are highlighted; conversely, when you select a state node or equation condition in the state transition table, the corresponding state node or transition arc is highlighted in the state diagram view.

2.8.4. Switch Between State Machines

A design may contain multiple state machines. To choose which state machine to view, use the **State Machine** selection box located at the top of the State Machine Viewer. Click in the drop-down box and select the necessary state machine.

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

The RTL Viewer, Technology Map Viewer, and State Machine 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.



2.10. 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, State Machine 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**.

2.11. 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 Standard 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 Standard Edition User Guide: Timing Analyzer*

2.12. Optimizing the Design Netlist Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	<ul style="list-style-type: none">• Initial release in Intel Quartus Prime Standard Edition User Guide.• Added link to <i>Viewing a Timing Path</i>.
2016.05.03	16.0.0	Removed Schematic Viewer topic.
2015.11.02	15.1.0	Added information for the following new features and feature updates: <ul style="list-style-type: none">• Nets visible across hierarchies• Connection Details• Display Settings• Hand Tool• Area Selection Tool• New default behavior for Show/Hide Instance Pins (default is now off)
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">• Updated screenshots• Updated chapter for the Intel Quartus Prime software version 10.0, including major user interface changes
November 2009	9.1.0	<ul style="list-style-type: none">• Updated devices• Minor text edits

continued...

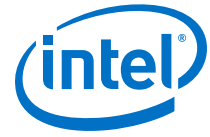


Document Version	Intel Quartus Prime Version	Changes
March 2009	9.0.0	<ul style="list-style-type: none"> Chapter 13 was formerly Chapter 12 in version 8.1.0 Updated Figure 13-2, Figure 13-3, Figure 13-4, Figure 13-14, and Figure 13-30 Added "Enable or Disable the Auto Hierarchy List" on page 13-15 Updated "Find Command" on page 13-44
November 2008	8.1.0	Changed page size to 8.5" × 11"
May 2008	8.0.0	<ul style="list-style-type: none"> Added Arria GX support Updated operator symbols Updated information about the radial menu feature Updated zooming feature Updated information about probing from schematic to Signal Tap Analyzer Updated constant signal information Added .png and .gif to the list of supported image file formats Updated several figures and tables 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" Renamed several sections Removed section "Customizing the Radial Menu" Moved section "Grouping Combinational Logic into Logic Clouds" Updated document content based on the Intel Quartus Prime software version 8.0

Related Information

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.



3. Timing Closure and Optimization

This chapter describes techniques to improve timing performance when designing for Intel devices. The application techniques vary between designs. Applying each technique does not always improve results.

Default settings and options in the Intel Quartus Prime software provide the best trade-off between compilation time, resource utilization, and timing performance. You can adjust these settings to determine whether other settings provide better results for your design.

3.1. 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%.

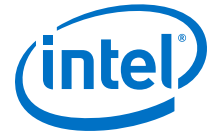
When this option is off, the Fitter optimizes designs considering only slow-corner delays from the slow-corner timing model (slowest manufactured device for a given speed grade, operating in low-voltage conditions).

3.2. Critical Paths

Critical paths are timing paths in your design that have a negative slack. 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

- [Reducing Critical Path Delay](#) on page 9
- [Displaying Path Reports with the Timing Analyzer](#) on page 61

3.2.1. 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.

3.3. Design Evaluation for Timing Closure

Follow the guidelines in this section when you encounter timing failures in a design. The guidelines show you how to evaluate compilation results of a design and how to address problems. While the guideline does not cover specific examples of restructuring RTL to improve design speed, the analysis techniques help you to evaluate changes to RTL that can help you to close timing.

3.3.1. Review Compilation Results

3.3.1.1. Review Messages

After compiling your design, review the messages in each section of the compilation report.

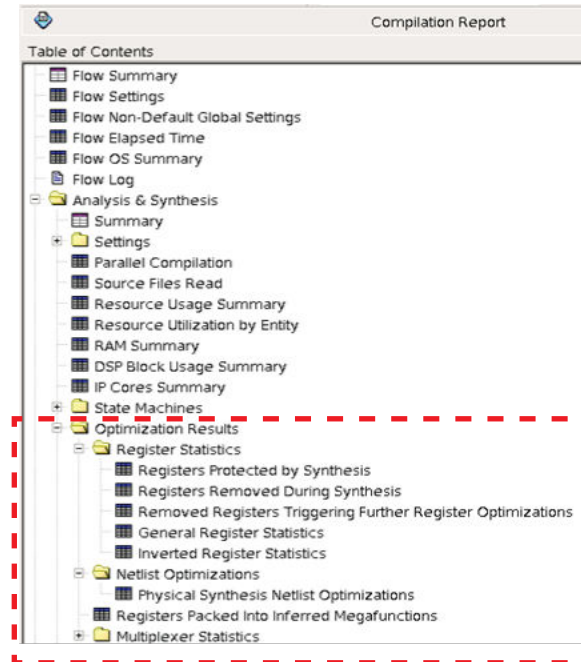
Most designs that fail timing start out with other problems that the Fitter reports as warning messages during compilation. Determine what causes a warning message, and whether to fix or ignore the warning.

After reviewing the warning messages, review the informational messages. Take note of anything unexpected, for example, unconnected ports, ignored constraints, missing files, and assumptions or optimizations that the software made.

3.3.1.2. Evaluate Physical Synthesis Results

If you enable physical synthesis options, the Compiler can duplicate and retime registers, and modify combinatorial logic during synthesis. After compilation, review the Optimization Results reports in the Analysis & Synthesis section. The reports list the optimizations performed by the physical synthesis optimizations, such as register duplication, retiming, and removal. These reports can be found in the Compilation Report panel.

Figure 12. Optimization Results Reports



When you enable physical synthesis, the compilation messages include a information about the physical synthesis algorithm performance improvement. The reported improvement is the sum of the largest improvement in each timing-critical clock domain. Although typically similar, the values for the slack improvements vary per compilation due to the random starting point of compilation algorithms.

3.3.1.3. Evaluate Fitter Netlist Optimizations

The Fitter can also perform optimizations to the design netlist. Major changes include register packing, duplicating or deleting logic cells, inverting signals, or modifying nodes in a general way such as moving an input from one logic cell to another. Find and review these reports in the Netlist Optimizations results of the Fitter section.

3.3.1.4. 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.

3.3.1.5. Evaluate Resource Usage

Evaluate a variety of resources used in the design, including global and non-global signal usage, routing utilization, and clustering difficulty.

3.3.1.5.1. 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 highlighted line has a single fan-out from a global clock.

Figure 13. Inefficient Use of a 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.

3.3.1.5.2. Routing Usage

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

Figure 14. Fitter Resource Usage Summary Report

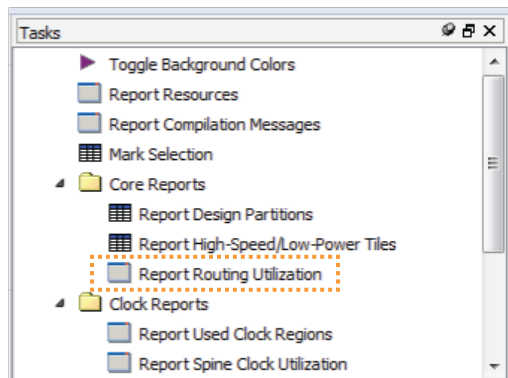
Table of Contents		Fitter Resource Usage Summary	
Resource	Usage		
43	10G TX PCSs	12 / 36 (33 %)	
44	HSSI PMA TX Serializers	12 / 36 (33 %)	
45	CHANNEL PLLS	12 / 36 (33 %)	
46	Impedance control blocks	1 / 4 (25 %)	
47	Average interconnect usage (total/H/V)	55% / 55% / 55%	
48	Peak interconnect usage (total/H/V)	88% / 88% / 90%	
49			

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.

The figure shows the **Report Routing Utilization** report.

Figure 15. Report Routing Utilization Report



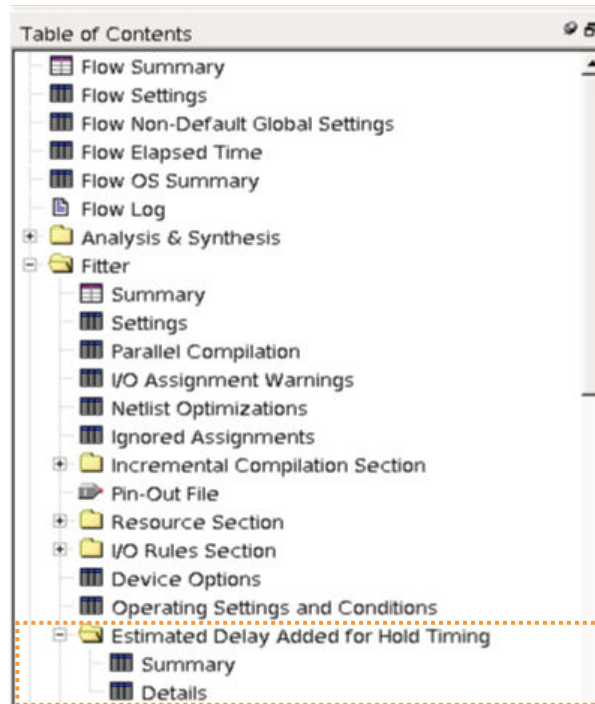
3.3.1.5.3. Wires Added for Hold

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 multicyle 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 16. Estimated Delay Added for Hold Timing Report

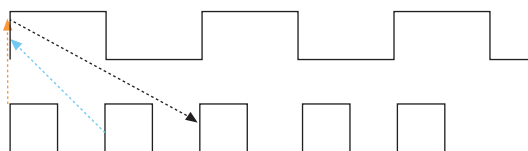


An example of an incorrect constraint which can cause the router to add wire for hold requirements is when there is data transfer from 1x to 2x clocks. Assume the design intent is to allow two cycles per transfer. Data can arrive any time in the two destination clock cycles by adding a multicycle setup 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 17. Timing Requirement Relaxed Waveform



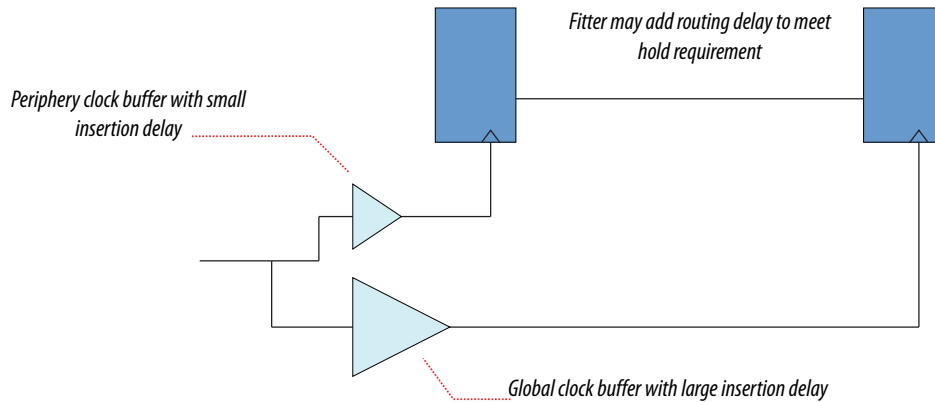
The default hold requirement, shown with the dashed blue line, can force the router to add wire to guarantee that data is delayed by one cycle. To correct the hold requirement, add a multicycle constraint with a hold option.

```
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 18. Clock Delay

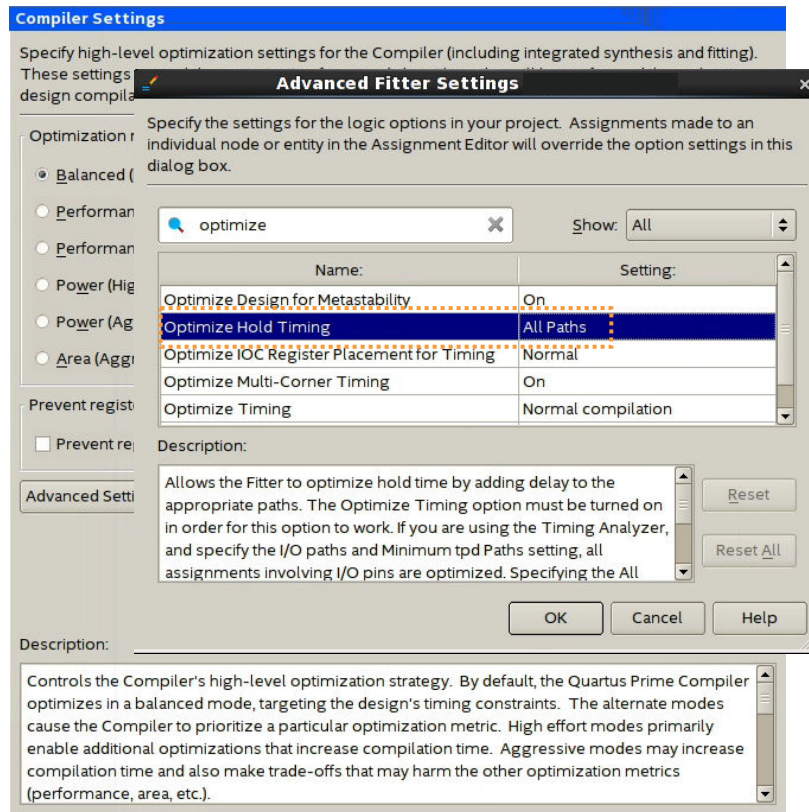


To identify cases where a path has different clock network types, review the path in the Timing Analyzer, and check nodes along the source and destination clock paths. Also, check the source and destination clock frequencies to see whether they are the same, or multiples, and whether there are multicycle 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.



Figure 19. Optimize Hold Timing Option



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.

3.3.1.6. Evaluate Other Reports and Adjust Settings Accordingly

3.3.1.6.1. 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 (Standard) 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.

3.3.1.6.2. Review Ignored Assignments

The **Compilation Report** includes details of any assignments ignored by the Fitter. Assignments typically get ignored if design names change, but assignments are not updated. Make sure any intended assignments are not being ignored.

3.3.1.6.3. Review Non-Default Settings

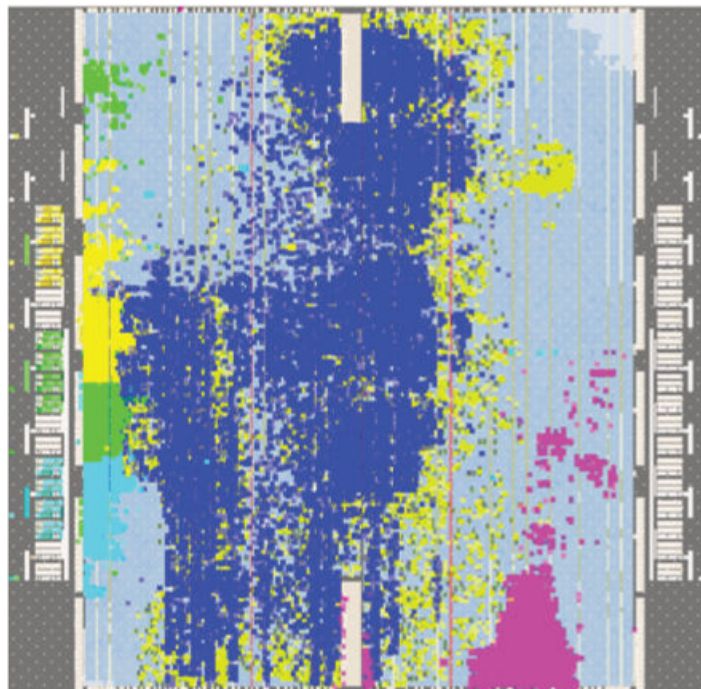
The reports from Synthesis and Fitter show non-default settings used in a compilation. Review the non-default settings to ensure the design benefits from the change.

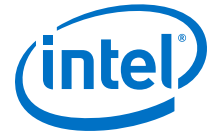
3.3.1.6.4. Review Floorplan

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 20. Floorplan with Color-Coded Entities





The following notes describe how you can use the visualization in *Floorplan with Color-Coded Entities* to check timing paths:

- 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.

3.3.1.6.5. Evaluate Placement and Routing

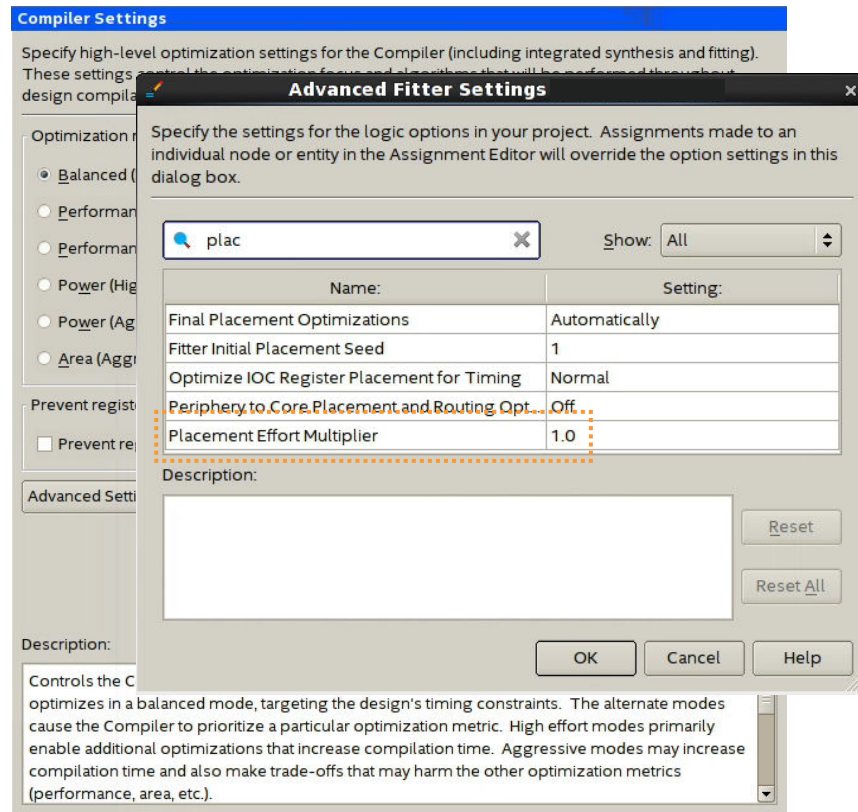
Review duration of parts of compile time in Fitter messages. If routing takes much more time than placement, then meeting timing may be more difficult than the placer predicted.

3.3.1.6.6. 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.

Figure 21. Placement Effort Multiplier



3.3.1.6.7. Adjust Fitter Effort

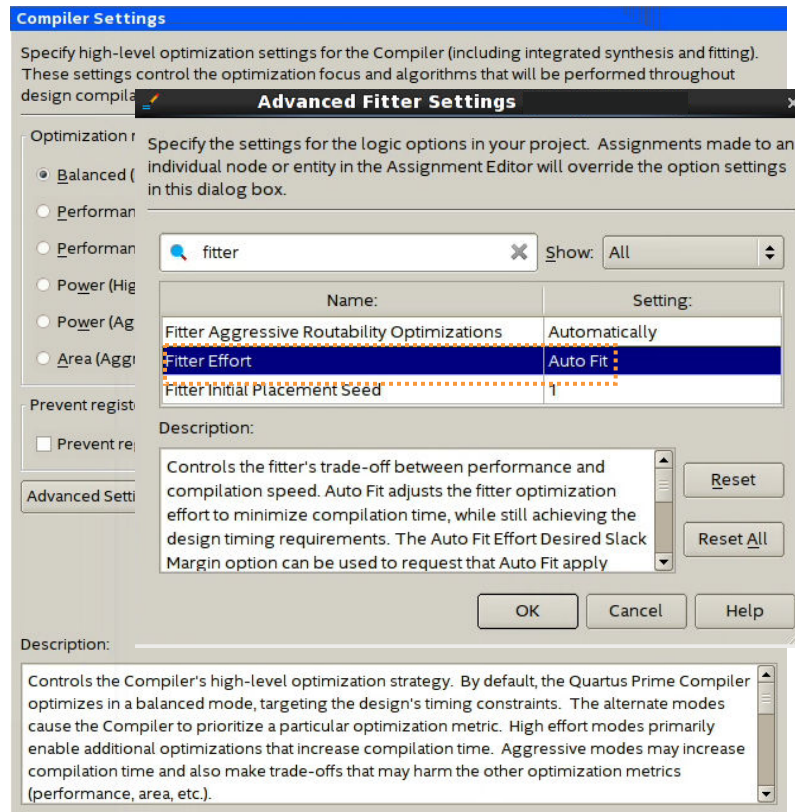
Fitter **Optimization mode** settings allow you to specify whether the Compiler focuses optimization efforts for performance, resource utilization, power, or compile times.

By default, the Fitter **Optimization mode** is set to **Balanced (Normal flow)**, which reduces Fitter effort once timing requirements are met. You can optionally select another **Optimization mode** to target performance, power, or resource usage.

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.



Figure 22. Fitter Effort



3.3.1.6.8. Review Timing Constraints

Ensure that clocks are constrained with the correct frequency requirements. Using the `derive_pll_clocks` assignment keeps generated clock settings updated. Timing Analyzer can be useful in reviewing SDC constraints. For example, under **Diagnostic** in the Task panel, the **Report Ignored Constraints** report shows any incorrect names in the design, most commonly caused by changes in the design hierarchy. Use the **Report Unconstrained Paths** report to locate unconstrained paths. Add constraints as necessary so that the design can be optimized.

3.3.1.7. 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.

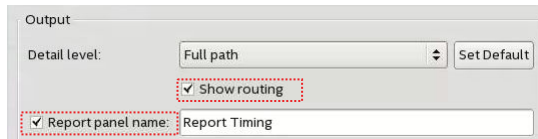
3.3.2. Review Details of Timing Paths

3.3.2.1. Show Timing Path Routing

Showing routing for a path can help uncover unusual routing delays.

In the Timing Analyzer **Report Timing** dialog box, enable the **Report panel name** and **Show routing** options, and click **Report Timing**.

Figure 23. Report Pane and Show Routing Options



The **Extra Fitter Information** tab shows a miniature floorplan with the path highlighted.

You can also locate the path in the Chip Planner to examine routing congestion, and to view whether nodes in a path are placed close together or far apart.

Related Information

[Exploring Paths in the Chip Planner](#) on page 110

3.3.2.2. Global Network Buffers

Routing paths allow you to identify global network buffers that fail timing. Buffer locations are named according to the network they drive.

- CLK_CTRL_Gn—for Global driver
- CLk_CTRL_Rn—for Regional driver

Buffers to access the global networks are located in the center of each side of the device. Buffering to route a core logic signal on a global signal network causes insertion delay. Tradeoffs 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 is demoted to local routing.

3.3.2.2.1. Source Location

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

3.3.2.2.2. Insertion Delay

If a global signal is required, consider adding half a cycle to timing by using a negative-edge triggered register to generate the signal (top figure) and use a multicycle setup constraint (bottom figure).

Figure 24. Negative-Edge Triggered Register

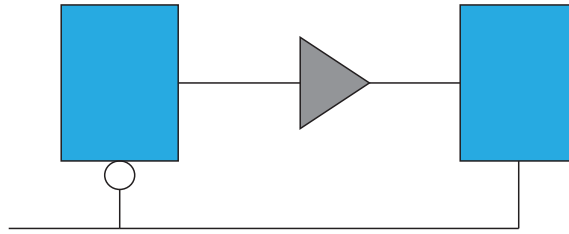
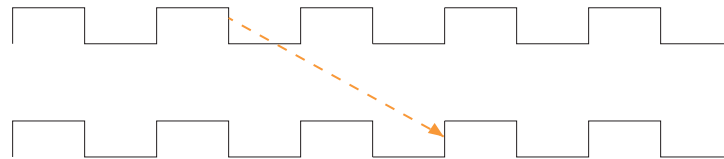


Figure 25. Multicycle Setup Constraint



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

3.3.2.2.3. 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.

3.3.2.2.4. Global Networks

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**.

Figure 26. Global Signal Assignment

To	Assignment Name /	Value	Enabled
reg_clk	Global Signal	Off	Yes

3.3.2.3. Resets and Global Networks

Reset signals are often routed 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.

3.3.2.4. 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.

3.3.2.5. Logic Depth

The **Statistics** tab in the Timing Analyzer path report shows the levels of logic in a path. If the path fails timing and the number of logic levels is high, consider adding pipelining in that part of the design.

3.3.2.6. 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 names of the converted registers include "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 is close to full, 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.

3.3.2.7. Clocking Architecture

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

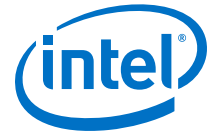
Timing failure can occur when the I/O interface at the top of the device connects to logic driven by a regional clock which is in one quadrant of the device, and placement restrictions force long paths to and from I/Os to logic across quadrants.

Use a different type of clock source to drive the logic - 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 (Standard) 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 107



3.3.2.8. Timing Closure Recommendations

The Report Timing Closure Recommendations task in the Timing Analyzer analyzes paths and provides specific recommendations based on path characteristics.

3.3.3. Adjusting and Recompiling

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 (Standard) regions, and re-evaluate paths that fail timing.

Partitioning often does not help timing closure, and must be done at the beginning of the design process. Adding partitions can increase logic utilization if it prevents cross-boundary optimizations, making timing closure harder and increasing compile times.

3.3.3.1. Using Partitions to Achieve Timing Closure

One technique to achieve timing closure is confining failing paths within individual design partitions, such that there are no failing paths passing between partitions. You can then use incremental make changes as necessary to correct the failing paths, and recompile only the affected partitions.

To use this technique:

1. In the Design Partition Planner, load timing data by clicking **View ► Show Timing Data**.
Entities containing nodes on failing paths appear in red in the Design Partition Planner.
2. Extract the entity containing failing paths by dragging it outside of the top-level entity window.
 - If there are no failing paths between the extracted entity and the top-level entity, right-click the extracted entity, and then click **Create Design Partition** to place that entity in its own partition.
3. Keep failing paths within a partition, so that there are no failing paths crossing between partitions.
If you are unable to isolate the failing paths from an extracted entity so that none are crossing partition boundaries, return the entity to its parent without creating a partition.
4. Find the partition having the worst slack value. For all the other partitions, preserve the contents and set as **Empty**.

For information about preserving the contents of a partition, refer to *Incremental Block-Based Compilation Flow* in the *Intel Quartus Prime Pro Edition User Guide: Block-Based Design*.

5. Adjust the logic in the partition and rerun the Fitter as necessary until the partition meets the timing requirements.
6. Repeat the process for all other design partitions with failing paths.

Related Information

- [Viewing Design Connectivity and Hierarchy](#)
- [Using Block-Based Compilation](#)
In *Intel Quartus Prime Pro Edition User Guide: Compiler*
- [Incremental Block-Based Compilation Flow](#)
In *Intel Quartus Prime Pro Edition User Guide: Block-Based Design*

3.4. Design Analysis

The initial compilation establishes whether the design achieves a successful fit and meets the specified timing requirements. This section describes how to analyze your design results in the Intel Quartus Prime software.

3.4.1. Ignored Timing Constraints

The Intel Quartus Prime software ignores illegal, obsolete, and conflicting constraints.

You can view a list of ignored constraints in the Timing Analyzer GUI by clicking **Reports > Report Ignored Constraints** or by typing the following command to generate a list of ignored timing constraints:

```
report_sdc -ignored -panel_name "Ignored Constraints"
```

Analyze any constraints that the Intel Quartus Prime software ignores. If necessary, correct the constraints and recompile your design before proceeding with design optimization.

You can view a list of ignored assignment in the **Ignored Assignment Report** generated by the Fitter.

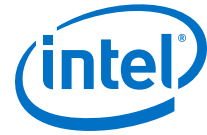
Related Information

[Creating I/O Requirements](#)

3.4.2. I/O Timing

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 Standard Edition Handbook Volume 3*

3.4.3. 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.

3.4.3.1. Displaying Path Reports with the Timing Analyzer

The Timing Analyzer generates reports with information about all valid register-to-register paths. To view all timing summaries, double-click **Report All Summaries** in the **Tasks** pane.

If any clock domains have failing paths (highlighted in red in the **Report** pane), right-click the clock name listed in the **Clocks Summary** pane and select **Report Timing** to get more details.

When you select a path in the **Summary of Paths** tab, the path detail pane displays all the path information. The **Extra Fitter Information** tab offers visual representation of the path location on the physical device. This can reveal whether the timing failure is distance related, due to the source and destination node being too close or too far.

The **Data Path** tab displays the Data Arrival Path and the Data Required Path. You can determine the path segments contributing the most to the timing violations with the incremental information. The **Waveform** tab shows the signals in the time domain, and plots the slack between arrival data and required data.

The RTL Viewer or Technology Map Viewer provide schematic (gate-level or technology-mapped) representations of the design netlist, and can help you to assess which areas in a design can benefit from reducing the number of logic levels. To locate a timing path in one of the viewers, right-click a path in the timing report, point to **Locate**, and select either **Locate in RTL Viewer** or **Locate in Technology Map Viewer**. You can also investigate the physical layout of a path in detail with the Chip Planner.

Related Information

- [When to Use the Netlist Viewers: Analyzing Design Problems](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*
- [Generating Timing Reports](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*

3.4.3.2. 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 110
- [Design Evaluation for Timing Closure](#) on page 45

3.4.3.3. 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 27. Different Value in From Clock and To Clock Field

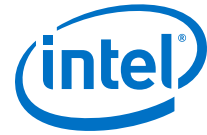
Setup Transfers						
	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	clk _{in}	clk _{in}	21	0	0	0
2	clk _{in}	clk _{out}	false path	0	0	0
3	clk _{out}	clk _{out}	31	0	0	0

- 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_clock_transfers](#)

In *Intel Quartus Prime Help*

3.4.3.4. Tips for Analyzing Paths from/to the Source and Destination of Critical Path

When analyzing the failing paths in a design, it is often helpful to get a fuller picture of the interactions around the paths.

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

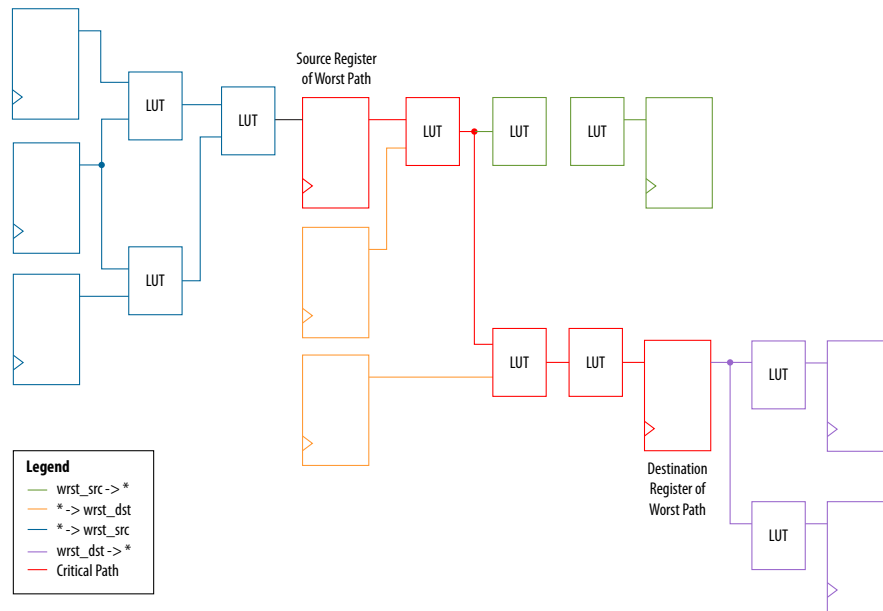
1. In the project directory, run the `report_timing` command to find the nodes in a critical path.
2. Copy the code below in a `.tcl` file, and replace the first two variable with the node names from the **From Node** and **To Node** columns of the worst path. The script analyzes the path between the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

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

Figure 28. 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.

3.4.3.5. 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

3.4.3.6. 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.

3.5. Timing Optimization

Use the following guidelines if your design does not meet its timing requirements.

3.5.1. Displaying Timing Closure Recommendations for Failing Paths

Use the **Timing Closure Recommendations** report to get specific recommendations about failing paths in your design and changes that you can make to potentially fix the failing paths.

1. In the **Tasks** pane of the Timing Analyzer, select the **Report Timing Closure Recommendations** task to open the **Report Timing Closure Recommendations** dialog box.
2. Select paths based on the clock domain, filter by nodes on path, and choose the number of paths to analyze.
3. After running the **Report Timing Closure Recommendations** task in the Timing Analyzer, examine the reports in the **Report Timing Closure Recommendations** folder in the **Report** pane of the Timing Analyzer GUI. Each recommendation has star symbols (*) associated with it. Recommendations with more stars are more likely to help you close timing on your design.

The reports give you the most probable causes of failure for each analyzed path, and show recommendations that may help you fix the failing paths.

The reports are organized into sections, depending on the type of issues found in the design, such as large clock skew, restricted optimizations, unbalanced logic, skipped optimizations, coding style that has too many levels of logic between registers, or region or partition constraints specific to your project.

For detailed analysis of the critical paths, run the `report_timing` command on specified paths. In the **Extra Fitter Information** tab of the **Path** report panel, you can see detailed fitter-related information that may help you visualize the issue.

Related Information

- [Fast Forward Timing Closure Recommendations](#)
- [Report Timing Closure Recommendations Dialog Box](#)
In *Intel Quartus Prime Help*

3.5.2. 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.

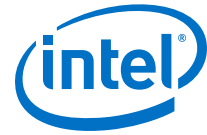
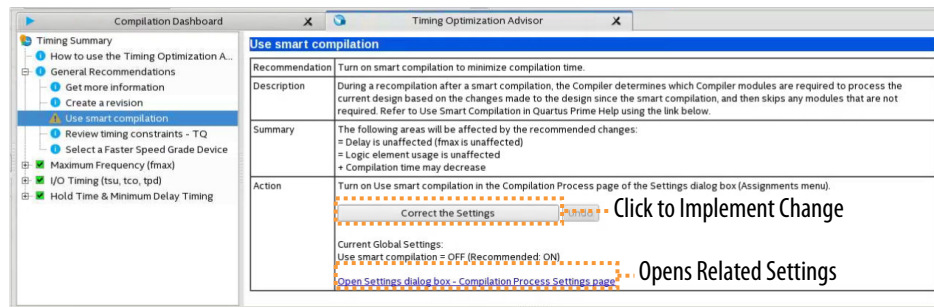


Figure 29. Timing Optimization Advisor



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.

3.5.3. 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](#)
In *Intel Quartus Prime Help*

3.5.3.1. Optimize Hold Timing

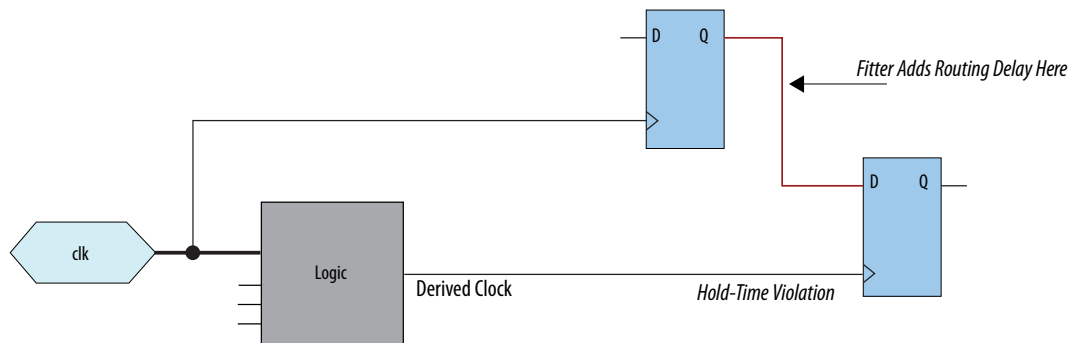
The **Optimize Hold Timing** option directs the Intel Quartus Prime software to optimize minimum delay timing constraints. Check your device information to determine whether the Intel Quartus Prime software optimizes hold timing for all paths or only for I/O paths and minimum t_{PD} paths.

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 30. 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

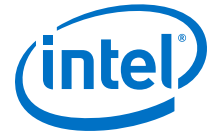
In *Intel Quartus Prime Standard Edition User Guide: Design Recommendations*

3.5.3.2. 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 8. Fitter Aggressive Routability Optimizations Logic Option Settings**

Settings	Description
Always	The Fitter always performs aggressive routability optimizations. If you set the Fitter Aggressive Routability Optimizations logic option to Always , 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 Automatically or Never .
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 Automatically or Never .

3.5.4. 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.

You can apply the suggestions this section to all Intel FPGA families and to the family of CPLDs.

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 9. 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 Input Delay from Pin to Input Register or set Decrease Input Delay to Input Register = ON	Yes	N/A
6	Decrease the value of Input Delay from Pin to Internal Cells or set Decrease Input Delay to Internal Cells = ON	Yes	N/A
7	Decrease the value of Delay from Output Register to Output Pin or set Increase Delay to Output Pin = OFF (refer to <i>Fast Input, Output, and Output Enable Registers</i>)	N/A	Yes

continued...



Order	Technique	Affects t_{SU}	Affects t_{CO}
8	Increase the value of Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations (refer to <i>Fast Input, Output, and Output Enable Registers</i>)	Yes	N/A
9	Use PLLs to shift clock edges	Yes	Yes
10	Use the Fast Regional Clock (refer to <i>Change How Hold Times are Optimized for MAX[®] II Devices</i>)	N/A	Yes
11	For MAX II or MAX V family devices, set Guarantee I/O Paths Have Zero Hold Time at Fast Corner to OFF , or When T_{SU} and T_{PD} Constraints Permit (refer to <i>Change How Hold Times are Optimized for MAX II Devices</i>)	Yes	N/A
12	Increase the value of Delay to output enable pin or set Increase delay to output enable pin (refer to <i>Use PLLs to Shift Clock Edges</i>)	N/A	Yes

[Optimize IOC Register Placement for Timing Logic Option](#) on page 70

[Fast Input, Output, and Output Enable Registers](#) on page 71

[Programmable Delays](#) on page 71

[Use PLLs to Shift Clock Edges](#) on page 72

[Use Fast Regional Clock Networks and Regional Clocks Networks](#) on page 72

[Spine Clock Limitations](#) on page 73

[Change How Hold Times are Optimized for Devices](#) on page 73

Related Information

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

3.5.4.1. 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.

Note: The option does not apply to series devices because they do not contain I/O registers.

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:

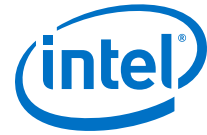
Note: To optimize registers with these characteristics, use other Intel Quartus Prime Fitter optimizations.

- Have combinational logic between the register and the pin
- Are part of a carry or cascade 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](#)

In *Intel Quartus Prime Help*



3.5.4.2. 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.

In series devices, which have no I/O registers, these assignments lock the register into the LAB adjacent to the I/O pin if there is a pin location assignment for that I/O pin.

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 (Standard) 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.

In series devices, the Fitter duplicates the register and places it in each distinct LAB location that is next to an I/O pin with a pin location assignment.

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

3.5.4.3. 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.

For Arria®, Cyclone®, MAX II, MAX V, and Stratix® series devices, 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:

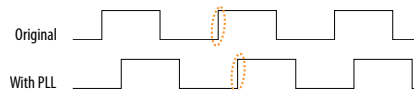
Related Information

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

3.5.4.4. 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 31. 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**.

3.5.4.5. 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.



3.5.4.6. 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 (Standard) regions, or if the Logic Lock (Standard) regions are not aligned to your clock region boundaries, create additional Logic Lock (Standard) regions and further constrain your logic.
- If Periphery features ignore Logic Lock (Standard) 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 107
- [Layers Settings and Editing Modes](#) on page 104
- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)
In *Intel Quartus Prime Help*

3.5.4.7. Change How Hold Times are Optimized for Devices

For devices, you can use the **Guarantee I/O Paths Have Zero Hold Time at Fast Corner** option to control how hold time is optimized by the Intel Quartus Prime software.

3.5.5. 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 Standard Edition User Guide: Design Recommendations*

3.5.5.1. 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 (Standard) 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.

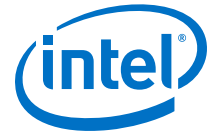
Related Information

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

3.5.5.2. 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
 - Change State Machine Encoding
 - Prevent Shift Register Inference
 - Use Other Synthesis Options Available in Your Synthesis Tool
5. To optimize for performance using physical synthesis, apply the following options:
 - Enable physical synthesis
 - Perform automatic asynchronous signal pipelining
 - Perform register duplication
 - Perform register retiming
 - Perform logic to memory mapping
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 (Standard) 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

[Initial Compilation: Required Settings](#)

In Intel Quartus Prime Standard Edition User Guide: Design Optimization

3.5.5.3. 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.

Note: If you want the performance gain from physical synthesis only on parts of your design, you can apply the physical synthesis options on specific instances with the Assignment Editor.

Related Information

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

3.5.5.4. Turn Off Extra-Effort Power Optimization Settings

If power optimization settings are set to **Extra Effort**, your design performance can be affected. If timing performance is more important than power, set the power optimization setting to **Normal**.

Related Information

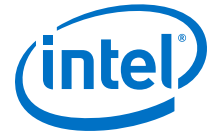
- [Power Optimization](#)
- [Power Optimization Logic Option](#)
In *Intel Quartus Prime Help*

3.5.5.5. 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.



To achieve best performance with push-button compilation, follow the recommendations in the following sections for other synthesis settings. You can use DSE II to experiment with different Intel Quartus Prime synthesis options to optimize your design for the best performance.

Related Information

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

3.5.5.6. 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.

Note: If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

3.5.5.7. 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.

3.5.5.8. Change State Machine Encoding

State machines can be encoded using various techniques. One-hot encoding, which uses one register for every state bit, usually provides the best performance. If your design contains state machines, changing the state machine encoding to one-hot can improve performance at the cost of area.

Related Information

[State Machine Processing Logic Option online help](#)

3.5.5.9. Duplicate Logic for Fan-Out Control

Oftentimes, timing failures occur not because of the high fan-out registers, but because of the location of those registers. Duplicating registers, where source and destination registers are physically close, can help improve slack on critical paths.

Synthesis tools support options or attributes that specify the maximum fan-out of a register. When using Intel Quartus Prime integrated 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.

3.5.5.10. 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.

3.5.5.11. Use Other Synthesis Options Available in Your Synthesis Tool

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

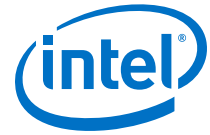
- Turn on register balancing or retiming
- Turn on register pipelining
- Turn off resource sharing

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

3.5.5.12. 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

[Design Space Explorer II](#) on page 12

3.5.5.13. 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*

3.5.6. Logic Lock (Standard) Assignments

Using Logic Lock (Standard) assignments to improve timing performance is only recommended for older devices, such as the MAX II family. For other device families, especially for larger devices such as Arria and Stratix series devices, do not use Logic Lock (Standard) assignments to improve timing performance. For these devices, use the feature for performance preservation and to floorplan your design.

Logic Lock (Standard) assignments do not always improve the performance of the design. In many cases, you cannot improve upon results from the Fitter by making location assignments. If there are existing Logic Lock (Standard) assignments in your design, remove the assignments if your design methodology permits it. Recompile the design, and then check if the assignments are making the performance worse.

When making Logic Lock (Standard) assignments, it is important to consider how much flexibility to give the Fitter. Logic Lock (Standard) assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher Fitter effort, but reduce the chance of design overconstraint.

The following types of Logic Lock (Standard) assignments are available, listed in the order of decreasing flexibility:

- Auto size, floating location regions
- Fixed size, floating location regions
- Fixed size, locked location regions

If you are unsure about the best size or location of a Logic Lock (Standard) region, the **Auto/Floating** options are useful for your first pass. After you determine where a Logic Lock (Standard) region must go, modify the Fixed/Locked regions, as Auto/Floating Logic Lock (Standard) regions can hurt your overall performance. To determine what to put into a Logic Lock (Standard) region, refer to the timing analysis results and analyze the critical paths in the Chip Planner. The register-to-register timing paths in the Timing Analyzer section of the Compilation Report help you recognize patterns.

Related Information

[Analyzing and Optimizing the Design Floorplan](#) on page 103

3.5.6.1. Hierarchy Assignments

For a design with the hierarchy shown in the figure, which has failing paths in the timing analysis results similar to those shown in the table, `mod_A` is probably a problem module. In this case, a good strategy to fix the failing paths is to place the `mod_A` hierarchy block in a Logic Lock (Standard) region so that all the nodes are closer together in the floorplan.

Figure 32. Design Hierarchy

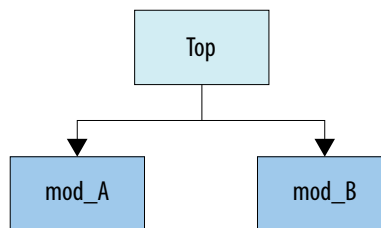
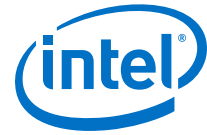


Table 10. Failing Paths in a Module Listed in Timing Analysis

From	To
mod_A reg1	mod_A reg9
mod_A reg3	mod_A reg5
<i>continued...</i>	



From	To
mod_A reg4	mod_A reg6
mod_A reg7	mod_A reg10
mod_A reg0	mod_A reg2

Hierarchical Logic Lock (Standard) regions are also important if you are using an incremental compilation flow. Place each design partition for incremental compilation in a separate Logic Lock (Standard) region to reduce conflicts and ensure good results as the design develops. You can use the auto size and floating location regions to find a good design floorplan, but fix the size and placement to achieve the best results in future compilations.

Related Information

[Analyzing and Optimizing the Design Floorplan](#) on page 103

3.5.7. Location Assignments

If a small number of paths are failing to meet their timing requirements, you can use hard location assignments to optimize placement.

Location assignments are less flexible for the Intel Quartus Prime Fitter than Logic Lock (Standard) assignments. Additionally, if you are familiar with your design, you can enter location constraints in a way that produces better results.

Note: Improving fitting results, especially for larger devices, such as Arria and Stratix series devices, can be difficult. Location assignments do not always improve the performance of the design. In many cases, you cannot improve upon the results from the Fitter by making location assignments.

3.5.8. 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.

This chapter describes how to enable metastability analysis and identify the register synchronization chains in your design, provides details about metastability reports, and provides additional guidelines for managing metastability.

Related Information

- [Understanding Metastability in FPGAs](#)

- Managing Metastability with the Intel Quartus Prime Software
In *Intel Quartus Prime Standard Edition User Guide: Design Recommendations*

3.6. 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.

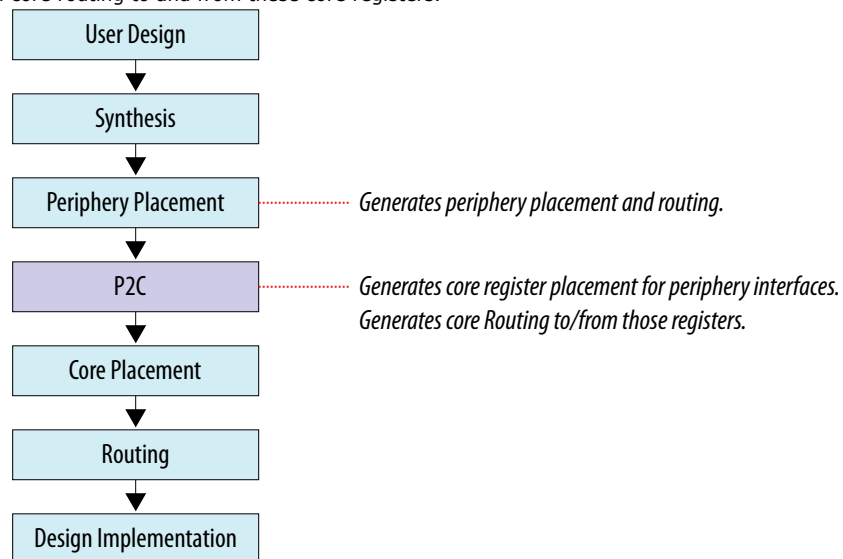
Note: The **Periphery to Core Register Placement and Routing Optimization** option applies in both directions, periphery to core and core to periphery.

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 33. Periphery to Core Register Placement and Routing Optimization (P2C) Flow

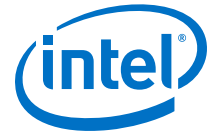
P2C runs after periphery placement, and generates placement for core registers on corresponding P2C/C2P paths, and core routing to and from these core registers.



[Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box](#) on page 83

[Setting Periphery to Core Optimizations in the Assignment Editor](#) on page 83

[Viewing Periphery to Core Optimizations in the Fitter Report](#) on page 84



3.6.1. 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.

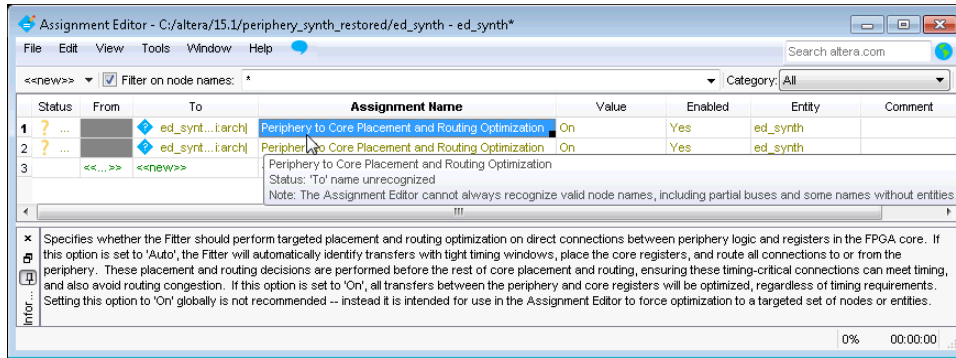
3.6.2. Setting Periphery to Core Optimizations in the Assignment Editor

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.



3.6.3. 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.

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 11. Fitter Report - Periphery to Core Transfer Optimization (P2C) Summary

From Path	To Path	Status
Node 1	Node 2	Placed and Routed —Core register is locked. Periphery to core/core to periphery routing is committed.
Node 3	Node 4	Placed but not Routed —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	Not Optimized —This occurs when P2C is set to Auto and the path is not optimized due to one of the following issues: <ol style="list-style-type: none"> a. The delay requirement is impossible to achieve. 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. c. P2C encountered unresolvable routing congestion for this particular path.

Periphery to Core Transfer Optimization Summary			
	From	To	Status
1	Periphery to Core Transfer		
2	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(1)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_single_port_afi_rdata_valid_regs_srs_out[0]	Placed but Not Routed
3	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(2)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[63]	Placed but Not Routed
4	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(1)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[11]	Placed but Not Routed
5	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[7]	Placed but Not Routed
6	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[3]	Placed but Not Routed
7	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[0]	Placed but Not Routed
8	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[79]	Placed but Not Routed
9	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[75]	Placed but Not Routed
10	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[73]	Placed but Not Routed
11	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[71]	Placed but Not Routed
12	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[69]	Placed but Not Routed
13	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(1)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[35]	Placed but Not Routed
14	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[31]	Placed but Not Routed
15	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[27]	Placed but Not Routed
16	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(2)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[23]	Placed but Not Routed



3.7. 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

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

3.7.1. 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 12. 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 13. 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

3.7.2. I/O Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the I/O timing optimization settings.

Table 14. 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

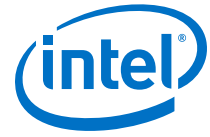
3.7.3. Register-to-Register Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the settings described in *Register-to-Register Timing Optimization Techniques*.

Table 15. 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
Perform Physical Synthesis for Combinational Logic (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC	ON, OFF	Global, Instance
Perform Register Duplication (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION	ON, OFF	Global, Instance
Perform Register Retiming (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_RETIMING	ON, OFF	Global, Instance
Perform Automatic Asynchronous Signal Pipelining (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING	ON, OFF	Global, Instance
Physical Synthesis Effort (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_EFFORT	NORMAL, EXTRA, FAST	Global

continued...



3. Timing Closure and Optimization

UG-20177 | 2018.11.12

Setting Name	.qsf File Variable Name	Values	Type
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

Related Information

Register-to-Register Timing Optimization Techniques on page 73

3.8. Timing Closure and Optimization Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.11.12	18.1.0	<ul style="list-style-type: none"> Updated "Placement Effort Multiplier" figure and text descriptions in "Adjust Placement Effort" topic. Updated "Fitter Effort" figure and text descriptions in "Adjust Fitter Effort" topic. Updated "Optimize Hold Timing Option" screenshot in "Wires Added for Hold" topic.
2018.09.24	18.1.0	<ul style="list-style-type: none"> Initial release in Intel Quartus Prime Standard Edition User Guide. Removed duplicated topic: <i>Resource Utilization Optimization Techniques</i>. The topic is now in the <i>Area Optimization</i> chapter.
2017.11.06	17.1.0	<ul style="list-style-type: none"> Moved Topic: Design Evaluation for Timing Closure after Initial Compilation: Optional Fitter Settings. Updated logic options about resource utilization optimization settings.
2017.05.08	17.0.0	<ul style="list-style-type: none"> Added topic: <i>Critical Paths</i>. Updated <i>Register-to-Register Timing</i> and renamed to <i>Register-to-Register Timing Analysis</i>. Renamed topic: <i>Timing Analysis with the Timing Analyzer</i> to <i>Displaying Path Reports with the Timing Analyzer</i>. Removed (LUT-Based Devices) remark from topic titles. Renamed topic: <i>Optimizing Timing (LUT-Based Devices)</i> to <i>Timing Optimization</i>. Renamed topic: <i>Debugging Timing Failures in the Timing Analyzer</i> to <i>Displaying Timing Closure Recommendations for Failing Paths</i>. Renamed topic: <i>Improving Register-to-Register Timing Summary</i> to <i>Improving Register-to-Register Timing</i>.
2016.05.02	16.0.0	<ul style="list-style-type: none"> Stated limitations about deprecated physical synthesis options. Added information about monitoring clustering difficulty.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Added: <i>Periphery to Core Register Placement and Routing Optimization</i>. Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. Updated DSE II content.
continued...		

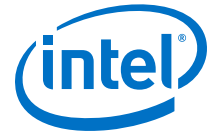


Document Version	Intel Quartus Prime Version	Changes
June 2014	14.0.0	<ul style="list-style-type: none"> • Dita conversion. • Removed content about obsolete devices that are no longer supported in QII software v14.0: Arria GX, Arria II, Cyclone III, Stratix II, Stratix III. • Replaced Megafunction content with IP core content.
November 2013	13.1.0	<ul style="list-style-type: none"> • Added Design Evaluation for Timing Closure section. • Removed Optimizing Timing (Macrocell-Based CPLDs) section. • Updated Optimize Multi-Corner Timing and Fitter Aggressive Routability Optimization. • Updated Timing Analysis with the Timing Analyzer to show how to access the Report All Summaries command. • Updated Ignored Timing Constraints to include a help link to <i>Fitter Summary Reports</i> with the Ignored Assignment Report information.
May 2013	13.0.0	<ul style="list-style-type: none"> • Renamed chapter title from Area and Timing Optimization to Timing Closure and Optimization. • Removed design and area/resources optimization information. • Added the following sections: Fitter Aggressive Routability Optimization. Tips for Analyzing Paths from/to the Source and Destination of Critical Path. Tips for Locating Multiple Paths to the Chip Planner. Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles.
November 2012	12.1.0	<ul style="list-style-type: none"> • 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.
June 2012	12.0.0	<ul style="list-style-type: none"> • 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. • Minor text edits throughout the chapter.
November 2011	11.1.0	<ul style="list-style-type: none"> • 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 • Updated Table 13-6 • Added the "Spine Clock Limitations" section • Removed the Change State Machine Encoding section from page 19 • Removed Figure 13-5 • Minor text edits throughout the chapter
May 2011	11.0.0	<ul style="list-style-type: none"> • Reorganized sections in "Initial Compilation: Optional Fitter Settings" section • Added new information to "Resource Utilization" section • Added new information to "Duplicate Logic for Fan-Out Control" section • Added links to Help • Additional edits and updates throughout chapter

continued...

3. Timing Closure and Optimization

UG-20177 | 2018.11.12



Document Version	Intel Quartus Prime Version	Changes
December 2010	10.1.0	<ul style="list-style-type: none">• Added links to Help• Updated device support• Added "Debugging Timing Failures in the Timing Analyzer" section• Removed Classic Timing Analyzer references• Other updates throughout chapter
August 2010	10.0.1	Corrected link
July 2010	10.0.0	<ul style="list-style-type: none">• Moved Compilation Time Optimization Techniques section to new <i>Reducing Compilation Time</i> chapter• Removed references to Timing Closure Floorplan• Moved Smart Compilation Setting and Early Timing Estimation sections to new <i>Reducing Compilation Time</i> chapter• Added Other Optimization Resources section• Removed outdated information• Changed references to DSE chapter to Help links• Linked to Help where appropriate• Removed Referenced Documents section

Related Information

[Documentation Archive](#)

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

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.

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).

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 44

4.2.1. Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize resource usage. To run the Resource Optimization Advisor click **Tools** ► **Advisors** ► **Resource Optimization Advisor**.

The Resource Optimization Advisor provides step-by-step advice about how to optimize resource usage (logic element, memory block, DSP block, I/O, and routing) of your design. Some of the recommendations in these categories might conflict with each other. Intel recommends evaluating the options and choosing the settings that best suit your requirements.

Related Information

[Resource Optimization Advisor Command Tools Menu](#)

4.2.2. 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*.

4.2.3. I/O Pin Utilization or Placement

Resolve I/O resource problems with these guidelines.

4.2.3.1. Guideline: Use I/O Assignment Analysis

To help with pin placement, click **Processing** ► **Start** ► **Start I/O Assignment Analysis**. The **Start I/O Assignment Analysis** command allows you to check your I/O assignments early in the design process. You can use this command to check the legality of pin assignments before, during, or after compilation of your design. If design files are available, you can use this command to accomplish more thorough legality checks on your design's I/O pins and surrounding logic. These checks include proper reference voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

Common issues with I/O placement relate to the fact that differential standards have specific pin pairings and certain I/O standards might be supported only on certain I/O banks.

If your compilation or I/O assignment analysis results in specific errors relating to I/O pins, follow the recommendations in the error message. Right-click the message in the **Messages** window and click **Help** to open the Intel Quartus Prime Help topic for this message.



4.2.3.2. 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 Standard Edition User Guide: Design Constraints*

4.2.4. 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.

4.2.4.1. 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 Standard Edition User Guide: Design Recommendations*

4.2.4.2. 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 integrated 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

[Intel Quartus Prime Integrated Synthesis](#)

4.2.4.3. 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

4.2.4.4. 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

4.2.4.5. 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

4.2.4.6. 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 (Standard) 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 (Standard) 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 (Standard) 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 (Standard) 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 (Standard) assignments in the Chip Planner, in the Logic Lock (Standard) 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 103

4.2.4.7. 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.

If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions, and the synthesis does not flatten it across partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

4.2.4.8. 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](#)

4.2.4.9. 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](#)



4.2.4.10. 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 integrated 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.

4.2.4.11. 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.

4.2.5. Routing

Resolve routing resource problems with these guidelines.

4.2.5.1. 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](#)

4.2.5.2. 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](#)

4.2.5.3. 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.

4.2.5.4. 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 (Standard) 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 (Standard) 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 (Standard) 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 (Standard) 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 (Standard) assignments in the Chip Planner, in the Logic Lock (Standard) 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 103

4.2.5.5. 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 integrated 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

[Intel Quartus Prime Integrated Synthesis](#)

4.2.5.6. 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.

4.2.5.7. 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.

2. Use the following Tcl command to make a global assignment:

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

3. 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

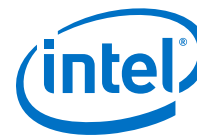
- [Tcl Scripting](#)
In *Intel Quartus Prime Standard Edition User Guide: Scripting*
- [Command Line Scripting](#)
In *Intel Quartus Prime Standard Edition User Guide: Scripting*

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 16. 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 17. Resource Utilization Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers ⁽¹⁾	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
Perform Physical Synthesis for Combinational Logic for Reducing Area (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global, Instance
Perform Physical Synthesis for Mapping Logic to Memory (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA	ON, OFF	Global, Instance
Optimization Technique	<device family name>_OPTIMIZATION_TECHNIQUE	AREA, SPEED, BALANCED	Global, Instance
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

(1) Allowed values for this setting depend on the device family that you select.



4.4. Area Optimization Revision History

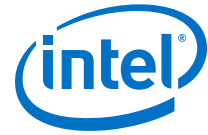
The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	<ul style="list-style-type: none">Initial release in Intel Quartus Prime Standard Edition User Guide.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>.
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">Revised topics: <i>Resolving Resource Utilization Issues</i>, <i>Guideline: Optimize Synthesis for Area, Not Speed</i>
2016.05.02	16.0.0	<ul style="list-style-type: none">Stated limitations about deprecated physical synthesis options.
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">Removed Cyclone III and Stratix III devices references.Removed Macrocell-Based CPLDs related information.Updated template.
May 2013	13.0.0	Initial release.

Related Information

[Documentation Archive](#)

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.



5. 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 (Standard) 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 (Standard) 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.

Related Information

[Managing Device I/O Pins](#)

5.1. 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.

The Chip Planner allows you to:


- Make assignment changes, such as creating and deleting resource assignments.
- Perform post-compilation changes such as creating, moving, and deleting logic cells and I/O atoms.
- Perform power and design analyses.
- Implement ECOs.
- Change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, RAMs, and digital signal processing (DSP) blocks.

The Chip Planner showcases:

- Logic Lock (Standard) 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

5.1.1. 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**:
 - Design Partition Planner
 - Compilation Report
 - **Logic Lock (Standard) Regions Window**
 - Technology Map Viewer
 - **Project Navigator** window
 - RTL source code
 - Node Finder
 - Simulation Report
 - RTL Viewer
 - Report Timing panel of the Timing Analyzer

5.1.2. Chip Planner GUI Components

5.1.2.1. Chip Planner Toolbar

The Chip Planner toolbar provides powerful tools for visual design analysis. You can access Chip Planner commands either from the **View** or the **Shortcut** menu, or by clicking the icons in the toolbars.

5.1.2.2. Layers Settings and Editing Modes

The Chip Planner allows you to control the display of resources. To determine the operations that you can perform, use the **Editing Mode**.

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. The **Design Partition Planner** preset is optimized for specific activities.



Editing Mode

The Chip Planner's **Editing Mode** determines the operations that you can perform. The **Assignment** editing mode allows you to make assignment changes that are applied by the Fitter during the next place and route operation. The **ECO** editing mode allows you to make post-compilation changes, commonly referred to as engineering change orders (ECOs).

Select the editing mode appropriate for the work that you want to perform, and a preset that displays the resources that you want to view, in a level of detail appropriate for your design.

Related Information

- [Viewing Architecture-Specific Design Information](#) on page 106
- [Layers Settings Dialog Box](#)
In *Intel Quartus Prime Help*

5.1.2.3. 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.

5.1.2.4. 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 (Standard) 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

5.1.3. 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.

In addition, you can modify the following device properties with the Chip Planner:

- LEs and ALMs
- I/O cells
- PLLs
- Registers in RAM and DSP blocks
- Connections between elements
- Placement of elements

For more information about LEs, ALMs, and other resources of an FPGA device, refer to the relevant device handbook.

Related Information

- [Layers Settings and Editing Modes](#) on page 104
- [Layers Settings Dialog Box](#)
In Intel Quartus Prime Help

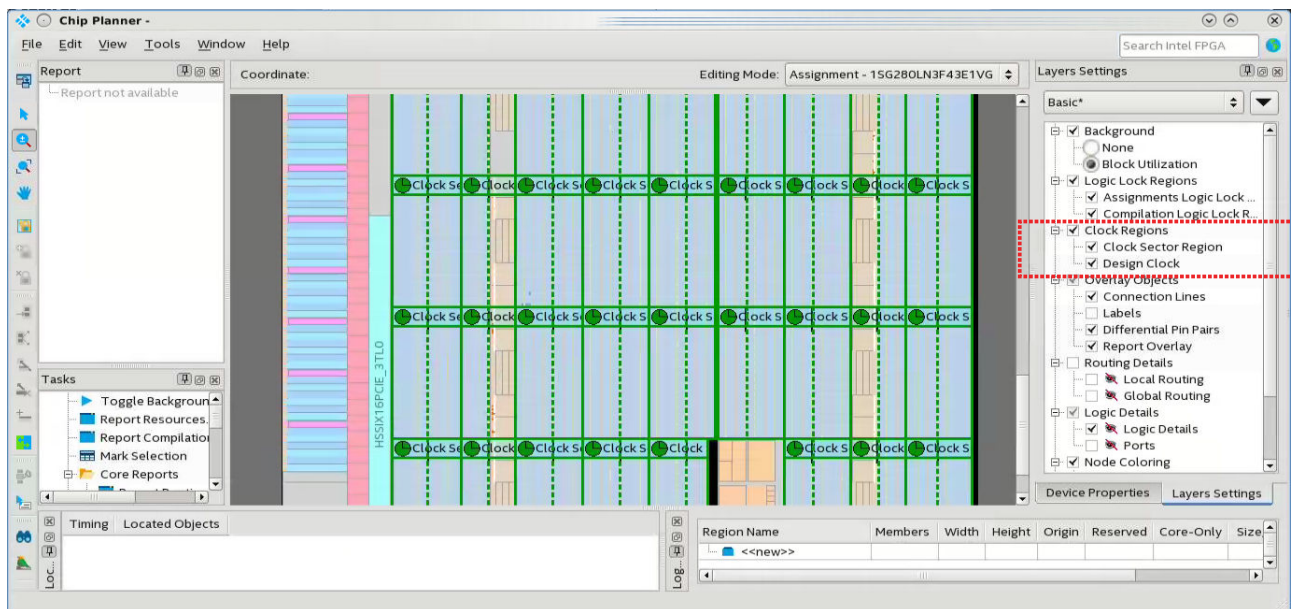


5.1.4. Viewing Available Clock Networks in the Device

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.

This global clock display feature is available for Arria V, Intel Arria 10, Cyclone V, Stratix IV, and Stratix V device families.

Figure 34. 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 73
- [Layers Settings and Editing Modes](#) on page 104
- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)
In *Intel Quartus Prime Help*

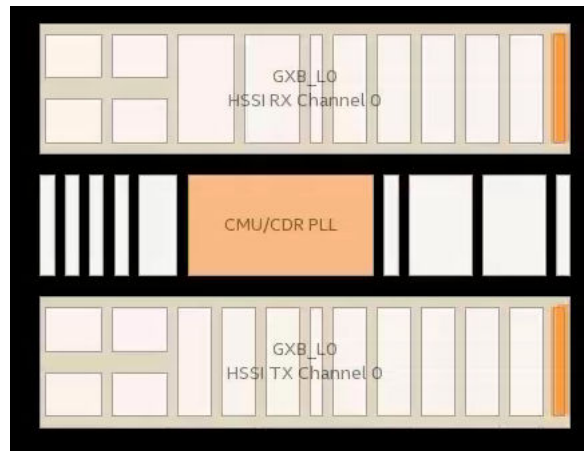
5.1.5. Viewing I/O Banks

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.

5.1.6. Viewing High-Speed Serial Interfaces (HSSI)

For selected device families, 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 35. Intel Arria 10 HSSI Channel Blocks



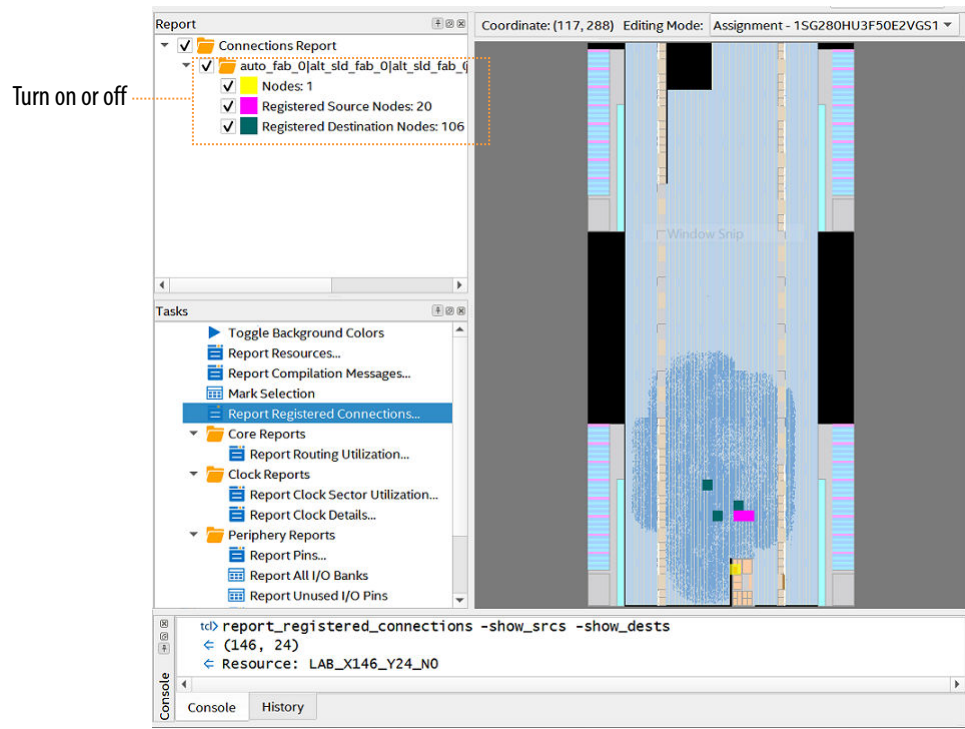
5.1.7. Viewing the Source and Destination of Placed Nodes

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 36. Report Registered Connections






Related Information

- [Viewing Fan-In and Fan-Out Connections of Placed Resources](#) on page 109
- [Expand Connections Command \(View Menu\)](#)
In *Intel Quartus Prime Help*

5.1.8. 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 108
- [Expand Connections Command \(View Menu\)](#)
In *Intel Quartus Prime Help*

5.1.9. Generating 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.

5.1.10. 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.

5.1.10.1. 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*

5.1.10.2. 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:

- Select the path you want to locate in the Timing Analysis report.
- 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 37. 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

Related Information

[Displaying Path Reports with the Timing Analyzer](#) on page 61

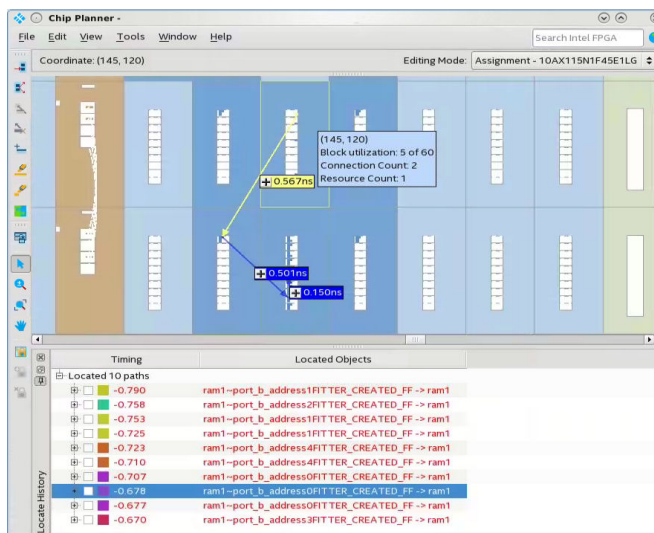


5.1.10.3. Show Delays

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 38. Show Delays Associated in a Timing Analyzer Path



5.1.10.4. 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 39. 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**.

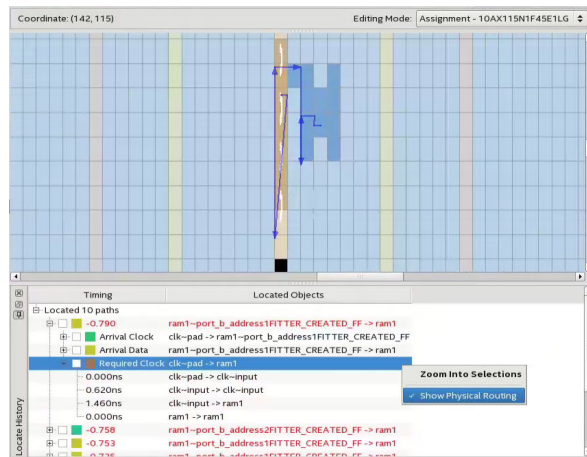
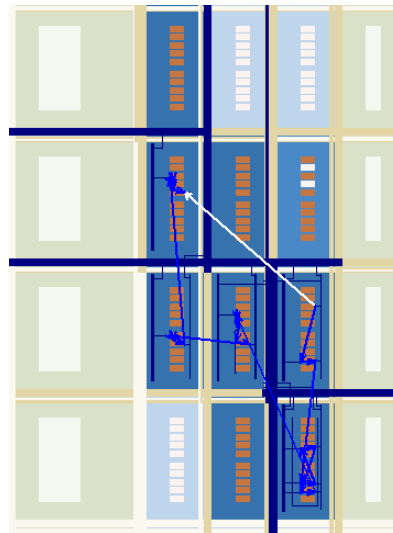


Figure 40. Highlight Routing

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](#)

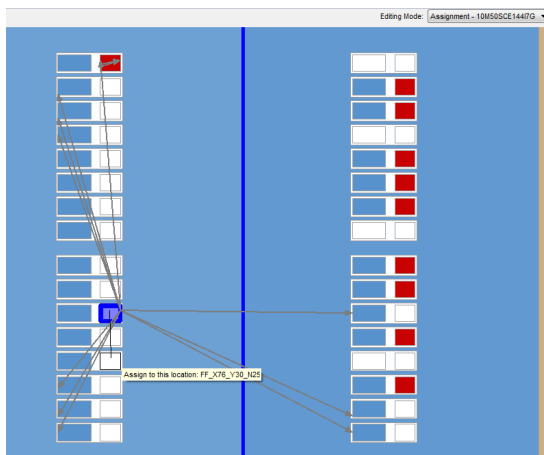
5.1.11. Viewing Assignments in the Chip Planner

You can view location assignments in the Chip Planner by using the Assignment editing mode and the **Floorplan Editing** preset in the **Layers Settings** pane.

The Chip Planner displays assigned resources in a predefined color (gray, by default).



Figure 41. Viewing Assignments in the Chip Planner



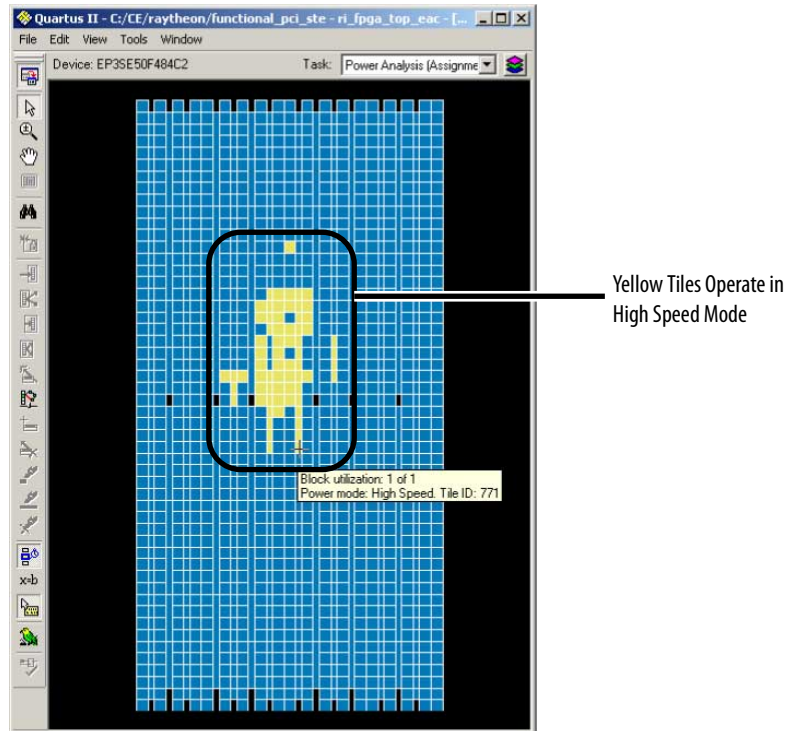
To create or move an assignment, or to make node and pin location assignments to Logic Lock (Standard) regions, drag the selected resource to a new location. The Fitter applies the assignments that you create during the next place-and-route operation.

5.1.12. 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 42. Viewing High-Speed and Low Power Tiles in a Stratix Device



Related Information

[AN 514: Power Optimization in Stratix IV FPGAs](#)

5.1.13. Viewing Design Partition Placement

With the **Report Design Partitions** command, you can view the physical placement of design partitions using the same color map as the Design Partition Planner.

The **Report Design Partitions Advanced** command opens the **Report Design Partitions Advanced** dialog box that allows you to select a partition and generate a report of the pins belonging to the partition. It highlights the selected partition's boundary ports and pins in the Chip Planner, and optionally reports the routing utilization and routing element details.

5.2. Logic Lock (Standard) Regions

Logic Lock (Standard) regions are floorplan location constraints. When you assign instances or nodes to a Logic Lock (Standard) region, you direct the Fitter to place those instances or nodes within the region. A floorplan can contain multiple Logic Lock (Standard) regions.

You can use the Design Partition Planner in conjunction with Logic Lock (Standard) regions to create a floorplan for your design.

Related Information

[Creating Logic Lock \(Standard\) Regions](#) on page 115



5.2.1. Attributes of a Logic Lock (Standard) Region

The following table lists the attributes of a Logic Lock (Standard) region. In the Intel Quartus Prime software, the Logic Lock (Standard) Regions window displays the attributes of all the Logic Lock (Standard) regions in the design.

Table 18. Attributes of Logic Lock (Standard) Regions

Name	Value	Behavior
Size	Auto Fixed	Auto allows the Intel Quartus Prime software to determine the appropriate size of a region given its contents. Fixed regions have a shape and size that you define.
Width	Number of columns	Specifies the width of the Logic Lock (Standard) region.
Height	Number of rows	Specifies the height of the Logic Lock (Standard) region.
State	Floating Locked	Floating allows the Intel Quartus Prime software to determine the location of the region on the device. Floating regions appear with a dashed boundary in the floorplan. Locked allows you to specify the location of the region. Locked regions appear with a solid boundary in the floorplan. A Locked region must have a Fixed size.
Origin	Any Floorplan Location Undetermined	Specifies the location of the Logic Lock (Standard) region on the floorplan. The origin is at the lower left corner of the Logic Lock (Standard) region.
Reserved	Off On	Prevents the Fitter from placing other logic in the region.

Related Information

[Logic Lock \(Standard\) Regions Window](#) on page 124

5.2.2. Creating Logic Lock (Standard) Regions

You can define a Logic Lock (Standard) region by its height, width, and location; Alternatively, you can specify the size or location of a region, or both, or the Intel Quartus Prime software can generate these properties automatically. The Intel Quartus Prime software bases the size and location of a region on the contents of the region and the timing requirements of the module.

The Intel Quartus Prime software displays Logic Lock (Standard) regions with colors indicating the percentage of resources available in the region. An orange Logic Lock (Standard) region indicates a nearly full Logic Lock (Standard) region.

Intel Quartus Prime software cannot automatically define the size of a region if the location is **Locked**. Therefore, if you want to specify the exact location of the region, you must also specify the size.

5.2.2.1. Creating Logic Lock (Standard) Regions with the Chip Planner

1. Click **View > Logic Lock (Standard) Regions > Create Logic Lock (Standard) 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.

5.2.2.2. Creating Logic Lock (Standard) 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 (Standard) Region**.
4. Assign the entity to the new region.

The new region has the same name as the entity.

5.2.2.3. Creating Logic Lock (Standard) Regions with the Logic Lock (Standard) Regions Window

1. Click **Assignments > Logic Lock (Standard) Regions Window**.
2. In the **Logic Lock (Standard) 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 \(Standard\) Regions Window](#) on page 124

5.2.2.4. Defining Routing Regions

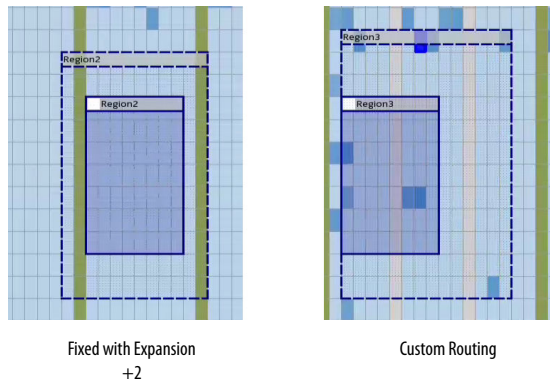
A routing region is an element of a Logic Lock region that specifies the routing area. A routing region must encompass the existing Logic Lock placement region. Routing regions cannot be set as reserved. To define the routing region, double-click the **Routing Region** cell in the **Logic Lock (Standard) Regions** window, and select an option from the drop-down menu.

Valid routing region options are:

Table 19. 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 Custom 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.

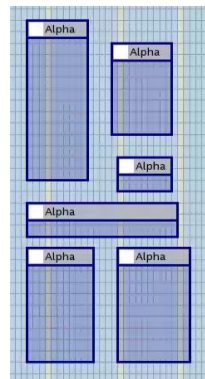
Figure 43. Routing Regions



5.2.2.5. Noncontiguous Logic Lock (Standard) Regions

You can create disjointed regions by using the Logic Lock (Standard) region manipulation tools. Noncontiguous regions act as a single Logic Lock (Standard) region for all Logic Lock (Standard) region attributes.

Figure 44. Noncontiguous Logic Lock (Standard) Region



Related Information

[Merging Logic Lock \(Standard\) Regions](#) on page 118

5.2.2.6. Considerations on Using Auto Sized Regions

If you use **Auto** Sized Logic Lock (Standard) regions, take into account:

- **Auto/Floating** regions cannot be reserved.
- Verify that your Logic Lock (Standard) 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** Sized Logic Lock (Standard) 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 (Standard) 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 (Standard) region.
 - If a partition is preserved at placement, routed, or final; a Logic Lock (Standard) region is not an effective placement boundary, because the location of the partition's logic is fixed.
 - However, if the Logic Lock (Standard) 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 (Standard) region meets your specification, you can:
 - Convert the Logic Lock (Standard) region to **Fixed** Size.
 - Leave the Logic Lock (Standard) region with **Auto** Sized attribute and use the region as a “keep together” type of function.
 - If the Logic Lock (Standard) region is also a partition, you can preserve the place and route through the partition and remove the Logic Lock (Standard) region entirely.

5.2.3. Customizing the Shape of Logic Lock Regions

To create custom shaped Logic Lock regions, you can perform logic operations. Non-rectangular Logic Lock (Standard) 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 (Standard) shapes for 17.1.

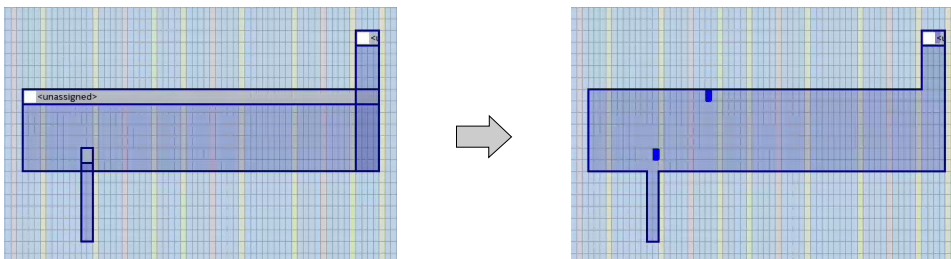
5.2.3.1. Merging Logic Lock (Standard) Regions

To merge two or more Logic Lock (Standard) 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 (Standard) regions and select **Logic Lock (Standard) Regions > Merge Logic Lock (Standard) Region**. The individual regions that you select merge to create a single new region.

Note: By default, the new Logic Lock (Standard) region has the same name as the component region containing the greatest number of resources; however, you can rename the new region. In the **Logic Lock (Standard) Regions Window**, the new region is shown as having a **Custom Shape**.

Figure 45. Using the Merge Logic Lock (Standard) Region command



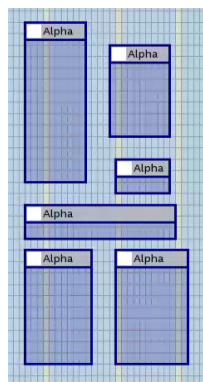
Related Information

[Creating Logic Lock \(Standard\) Regions](#) on page 115

5.2.3.2. Noncontiguous Logic Lock (Standard) Regions

You can create disjointed regions by using the Logic Lock (Standard) region manipulation tools. Noncontiguous regions act as a single Logic Lock (Standard) region for all Logic Lock (Standard) region attributes.

Figure 46. Noncontiguous Logic Lock (Standard) Region



Related Information

[Merging Logic Lock \(Standard\) Regions](#) on page 118

5.2.4. Placing Logic Lock (Standard) Regions

A fixed region must contain all resources required by the design block assigned to the region. Although the Intel Quartus Prime software can automatically place and size Logic Lock (Standard) regions to meet resource and timing requirements, you can manually place and size regions to meet your design requirements.

If you manually place or size a Logic Lock (Standard) region:

- Logic Lock (Standard) regions with pin assignments must be placed on the periphery of the device, adjacent to the pins. You must also include the I/O block within the Logic Lock (Standard) Region.
- Floating Logic Lock (Standard) regions can overlap with their ancestors or descendants, but not with other floating Logic Lock (Standard) regions.

5.2.5. Placing Device Resources into Logic Lock (Standard) Regions

You can assign an entity in the design to only one Logic Lock (Standard) 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 (Standard) region boundary includes part of a device resource, the Intel Quartus Prime software allocates the entire resource to that Logic Lock (Standard) region. When the Intel Quartus Prime software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the Logic Lock (Standard) 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 (Standard) region's name field in the Logic Lock (Standard) Regions Window.

5.2.5.1. 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:

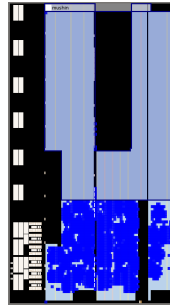
- Preliminary floorplanning.
- Complex incremental builds.
- Team based design and interconnect logic.
- Confining logic placements.

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 47. 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).

5.2.5.2. Pin Assignment

A Logic Lock (Standard) region incorporates all device resources within its boundaries, including memory and pins. The Intel Quartus Prime Standard Edition software automatically includes pins when you assign an instance to a region. You can manually exclude pins with a **Core Only** assignment.

Note: Pin assignments to Logic Lock (Standard) regions are effective only in fixed and locked regions. Pin assignments to floating regions do not influence the placement of the region.

You can assign an entity in the design to only one Logic Lock (Standard) region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a subregion without reserving the resources in the subregion.

When the Intel Quartus Prime software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the Logic Lock (Standard) region.

5.2.5.3. Reserved Logic Lock (Standard) Regions

The **Reserved** attribute instructs the Fitter to only place the entities and nodes that you specifically assigned to the Logic Lock (Standard) region in the Logic Lock (Standard) region.

The Intel Quartus Prime software honors all entity and node assignments to Logic Lock (Standard) 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** on the **Logic Lock (Standard) Region Properties > General** tab.

5.2.5.4. Excluded Resources

The Excluded Resources feature allows you to easily exclude specific device resources such as DSP blocks or M4K memory blocks from a Logic Lock (Standard) region.

For example, you can assign a specific entity to a Logic Lock (Standard) region but allow the DSP blocks of that entity to be placed anywhere on the device. Use the Excluded Resources feature on a per-Logic Lock (Standard) region member basis.

To exclude certain device resources from an entity, in the **Logic Lock (Standard) Region Properties** dialog box, highlight the entity in the **Design Element** column, and click **Edit**. In the **Edit Node** dialog box, under **Excluded Element Types**, click the **Browse** button. In the **Excluded Resources Element Types** dialog box, you can select the device resources you want to exclude from the entity. When you have selected the resources to exclude, the **Excluded Resources** column is updated in the **Logic Lock (Standard) Region Properties** dialog box to reflect the excluded resources.

Note: The Excluded Resources feature prevents certain resource types from being included in a region, but it does not prevent the resources from being placed inside the region unless you set the region's **Reserved** property to **On**. To indicate to the Fitter that certain resources are not required inside a Logic Lock (Standard) region, define a resource filter.

5.2.5.5. Logic Lock (Standard) Assignment Precedence

You can encounter conflicts during the assignment of entities and nodes to Logic Lock (Standard) regions. For example, an entire top-level entity might be assigned to one region and a node within this top-level entity assigned to another region.

To resolve conflicting assignments, the Intel Quartus Prime software maintains an order of precedence for Logic Lock (Standard) assignments. The following order of precedence, from highest to lowest, applies:

1. Exact node-level assignments
2. Path-based and wildcard assignments
3. Hierarchical assignments

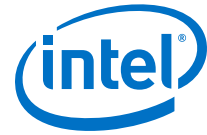
Note: To open the **Priority** dialog box, select **Logic Lock (Standard) Regions Properties > General > Priority**. You can change the priority of path-based and wildcard assignments with the **Up** and **Down** buttons in the **Priority** dialog box. To prioritize assignments between regions, you must select multiple Logic Lock (Standard) regions and then open the **Priority** dialog box from the **Logic Lock (Standard) Regions Properties** dialog box.

5.2.5.6. 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, Analysis & 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 (Standard) 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 128
- [Managing Device I/O Pins](#)
- [Node Finder Command \(View Menu\)](#)
In *Intel Quartus Prime Help*

5.2.6. Hierarchical (Parent and Child) Logic Lock (Standard) Regions

To further constrain module locations, you can define a hierarchy for a group of regions by declaring parent and child regions.

The Intel Quartus Prime software places a child region completely within the boundaries of its parent region; a child region must be placed entirely within the boundary of its parent. Additionally, parent and child regions allow you to further improve the performance of a module by constraining nodes in the critical path of a module.

To make one Logic Lock (Standard) region a child of another Logic Lock (Standard) region, in the Logic Lock (Standard) Regions window, select the new child region and dragging the new child region into its new parent region.

Note: The Logic Lock (Standard) region hierarchy does not have to be the same as the design hierarchy.

You can create both auto-sized and fixed-sized Logic Lock (Standard) regions within a parent Logic Lock (Standard) region; however, the parent of a fixed-sized child region must also be fixed-sized. The location of a locked parent region is locked relative to the device; the location of a locked child region is locked relative to its parent region. If you change the parent's location, the locked child's origin changes, but maintains the same placement relative to the origin of its parent. The location of a floating child region can float within its parent. Complex region hierarchies might result in some LABs not being used, effectively increasing the resource utilization in the device. Do not create more levels of hierarchy than you need.



The Logic Lock (Standard) Regions Window also has a recommendations toolbar; select a Logic Lock (Standard) region from the drop-down list in the recommendations toolbar to display the relevant suggestions to optimize that Logic Lock (Standard) region.

The Intel Quartus Prime software automatically creates a Logic Lock (Standard) region that encompasses the entire device. This default region is labeled `Root_Region`, and is locked and fixed.

You can customize the Logic Lock (Standard) 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 (Standard) Regions Properties Dialog Box

Use the **Logic Lock (Standard) Regions Properties** dialog box to view and modify detailed information about your Logic Lock (Standard) region, such as which entities and nodes are assigned to your region, and which resources are required.

To open the **Logic Lock (Standard) Regions Properties** dialog box, right-click the region and select **Logic Lock (Standard) Regions Properties...**

Related Information

- [Attributes of a Logic Lock \(Standard\) Region](#) on page 115
- [Creating Logic Lock \(Standard\) Regions with the Logic Lock \(Standard\) Regions Window](#) on page 116
- [Logic Lock \(Standard\) Regions Window](#)
In *Intel Quartus Prime Help*

5.3. Using Logic Lock (Standard) Regions in the Chip Planner

You can easily create Logic Lock (Standard) regions in the Chip Planner and assign resources to them.

5.3.1. Viewing Connections Between Logic Lock (Standard) Regions in the Chip Planner

You can view and edit Logic Lock (Standard) regions using the Chip Planner. To view and edit Logic Lock (Standard) regions, use **Floorplan Editing** in the **Layers Settings** window, or any layers setting mode that has the **User-assigned Logic Lock (Standard) regions** setting enabled.

The Chip Planner shows the connections between Logic Lock (Standard) regions. By default, you can view each connection as an individual line. You can choose to display connections between two Logic Lock (Standard) 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.

5.3.2. Using Logic Lock (Standard) Regions with the Design Partition Planner

You can optimize timing in a design by placing entities that share significant logical connectivity close to each other on the device.

By default, the Fitter usually places closely connected entities in the same area of the device; however, you can use Logic Lock (Standard) regions, together with the Design Partition Planner and the Chip Planner, to help ensure that logically connected entities retain optimal placement from one compilation to the next.

You can view the logical connectivity between entities with the Design Partition Planner, and the physical placement of those entities with the Chip Planner. In the Design Partition Planner, you can identify entities that are highly interconnected, and place those entities in a partition. In the Chip Planner, you can create Logic Lock (Standard) regions and assign each partition to a Logic Lock (Standard) region, thereby preserving the placement of the entities.

5.4. 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 Standard Edition User Guide: Scripting*
- [Command Line Scripting](#)
In *Intel Quartus Prime Standard Edition User Guide: Scripting*

5.4.1. Initializing and Uninitializing a Logic Lock (Standard) Region

You must initialize the Logic Lock (Standard) data structures before creating or modifying any Logic Lock (Standard) regions and before executing any of the Tcl commands listed below.

Use the following Tcl command to initialize the Logic Lock (Standard) data structures:

```
initialize_logiclock
```

Use the following Tcl command to uninitialize the Logic Lock (Standard) data structures before closing your project:

```
uninitialize_logiclock
```

5.4.2. Creating or Modifying Logic Lock (Standard) Regions

Use the following Tcl command to create or modify a Logic Lock (Standard) region:

```
set_logiclock -auto_size true -floating true -region <my_region-name>
```

Note: The command in the above example sets the size of the region to auto and the state to floating.



If you specify a region name that does not exist in the design, the command creates the region with the specified properties. If you specify the name of an existing region, the command changes all properties you specify and leaves unspecified properties unchanged.

Related Information

[Creating Logic Lock \(Standard\) Regions](#) on page 115

5.4.3. Obtaining Logic Lock (Standard) Region Properties

Use the following Tcl command to obtain Logic Lock (Standard) region properties. This example returns the height of the region named `my_region`:

```
get_logiclock -region my_region -height
```

5.4.4. Assigning Logic Lock (Standard) Region Content

Use the following Tcl commands to assign or change nodes and entities in a Logic Lock (Standard) region. This example assigns all nodes with names matching `fifo*` to the region named `my_region`.

```
set_logiclock_contents -region my_region -to fifo*
```

You can also make path-based assignments with the following Tcl command:

```
set_logiclock_contents -region my_region -from fifo -to ram*
```

5.4.5. Save a Node-Level Netlist for the Entire Design into a Persistent Source File

Make the following assignments to cause the Intel Quartus Prime Fitter to save a node-level netlist for the entire design into a `.vqm` file:

```
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT ON  
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_FILE <file name>
```

Any path specified in the file name is relative to the project directory. For example, specifying `atom_netlists/top.vqm` places `top.vqm` in the **atom_netlists** subdirectory of your project directory.

A `.vqm` file is saved in the directory specified at the completion of a full compilation.

Note: The saving of a node-level netlist to a persistent source file is not supported for designs targeting newer devices such as MAX V , Stratix IV, or Stratix V.

5.4.6. Setting Logic Lock (Standard) Assignment Priority

Use the following Tcl code to set the priority for a Logic Lock (Standard) region's members. This example reverses the priorities of the Logic Lock (Standard) region in your design.

```
set reverse [list]
for each member [get_logiclock_member_priority] {
    set reverse [insert $reverse 0 $member]
}
set_logiclock_member_priority $reverse
```

5.4.7. 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 122
- [Managing Device I/O Pins](#)
- [Node Finder Command \(View Menu\)](#)
In *Intel Quartus Prime Help*

5.5. Analyzing and Optimizing the Design Floorplan Revision History

The following revision history applies to this chapter:

Table 20. Document Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	<ul style="list-style-type: none"> • Initial release in Intel Quartus Prime Standard Edition User Guide. • Renamed topic: <i>Generating Fan-In and Fan-Out Connections to Viewing Fan-In and Fan-Out Connections of Placed Resources</i>.
2018.05.07	18.0.0	<ul style="list-style-type: none"> • Added recommendations for using iterative methods for floorplanning.
2017.11.06	17.1.0	<ul style="list-style-type: none"> • Changed instances of <i>LogicLock</i> to <i>Logic Lock (Standard)</i>.
2017.05.08	17.0.0	<ul style="list-style-type: none"> • Chapter reorganization and content update. • Added figures: Clock Regions, Creating a Hole in a LogicLock Region, Noncontiguous LogicLock Region, Routing Regions, Logic Placed Outside of an Empty Region. • Moved topic: <i>Viewing Critical Paths to Timing Closure and Optimization</i> chapter and renamed to <i>Critical Paths</i>. • Renamed topic: <i>Creating Non-Rectangular LogicLock Plus Regions to Merging LogicLock Plus Regions</i>. • Renamed topic: <i>Chip Planner Overview to Design Floorplan Analysis in the Chip Planner</i>. • Renamed chapter from <i>Analyzing and Optimizing the Design Floorplan with the Chip Planner</i> to <i>Analyzing and Optimizing the Design Floorplan</i>.
2015.11.02	15.1.0	<ul style="list-style-type: none"> • Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
continued...		



Document Version	Intel Quartus Prime Version	Changes
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"> Updated for the 11.0 release. Edited "LogicLock Regions" Updated "Viewing Routing Congestion" Updated "Locate History" Updated Figures 15-4, 15-9, 15-10, and 15-13 Added Figure 15-6
December 2010	10.1.0	<ul style="list-style-type: none"> Updated for the 10.1 release.
July 2010	10.0.0	<ul style="list-style-type: none"> Updated device support information Removed references to Timing Closure Floorplan; removed "Design Analysis Using the Timing Closure Floorplan" section Added links to online Help topics Added "Using LogicLock Regions with the Design Partition Planner" section Updated "Viewing Critical Paths" section Updated several graphics Updated format of Document revision History table
November 2009	9.1.0	<ul style="list-style-type: none"> Updated supported device information throughout 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.) Updated "Creating Nonrectangular LogicLock Regions" section Added "Selected Elements Window" section Updated table 12-1
May 2008	8.0.0	<ul style="list-style-type: none"> Updated the following sections: <ul style="list-style-type: none"> "Chip Planner Tasks and Layers" "LogicLock Regions" "Back-Annotating LogicLock Regions" "LogicLock Regions in the Timing Closure Floorplan" Added the following sections: <ul style="list-style-type: none"> "Reserve LogicLock Region" "Creating Nonrectangular LogicLock Regions" "Viewing Available Clock Networks in the Device" Updated Table 10-1 Removed the following sections: <ul style="list-style-type: none"> Reserve LogicLock Region Design Analysis Using the Timing Closure Floorplan



Related Information

[Documentation Archive](#)

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

6. Netlist Optimizations and Physical Synthesis

The Intel Quartus Prime software offers netlist and physical synthesis optimizations that improve performance of your design. Click to enable physical synthesis options during fitting. This chapter also provides guidelines for applying netlist and physical synthesis options, and for preserving compilation results through back-annotation.

Table 21. Netlist Optimization and Physical Synthesis Options

Options	Location/Description
Enable physical synthesis options.	Assignments > Settings > Compiler Settings > Advanced Settings (Fitter). Physical synthesis optimizations apply at different stages of the compilation flow, either during synthesis, fitting, or both.
Enable netlist optimization options.	Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis). Netlist optimizations operate with the atom netlist of your design, which describes a design in terms of specific primitives. An atom netlist file can be an Electronic Design Interchange Format (.edf) file or a Verilog Quartus Mapping (.vqm) file generated by a third-party synthesis tool. Intel Quartus Prime synthesis generates and internally uses the atom netlist internally

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 (Standard) based optimization flow (for legacy devices), disable physical synthesis options.

6.1. 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.

Some physical synthesis options affect only registered logic, while others affect only combinational logic. Select options based on whether you want to keep the registers intact. For example, if your verification flow involves formal verification, you might want to keep the registers intact.

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*

6.1.1. 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 physical synthesis, click **Advanced Settings (Fitter)**, and then enable **Perform Physical Synthesis for Combinational Logic for Performance** and **Perform Physical Synthesis for Combinational Logic for Fitting**.
3. View physical synthesis results in the **Netlist Optimizations** report.

6.1.2. Physical Synthesis Options

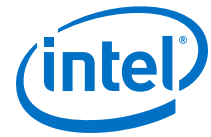
The Intel Quartus Prime software provides physical synthesis optimization options to improve fitting results. To access these options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

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 22. Physical Synthesis Options

Option	Description
Perform asynchronous signal pipelining (no Intel Arria 10 support)	Automatically inserts pipeline stages for asynchronous clear and asynchronous load signals during fitting to increase circuit performance. This option is useful for asynchronous signals that are failing recovery and removal timing because they feed registers using a high-speed clock. You can use this option if asynchronous control signal recovery and removal times are not achieving requirements. This option adds registers and potential latency to nets driving the asynchronous clear or asynchronous load ports of registers. The additional register delays can change the behavior of the signal in the design; therefore, you should use this option only if additional latency on the reset signals does not violate any design requirements. This option also prevents the promotion of signals to global routing resources.
Perform Register Duplication for Performance (no Intel Arria 10 support)	Duplicates registers based on Fitter placement information to reduce the delay of one path without degrading the delay of another. You can also duplicate combinational logic when you enable this option. The Fitter can place the new logic cell closer to critical logic without affecting the other fan-out paths of the original logic cell. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow .
Perform Register Retiming for Performance (no Arria 10 support)	Enables the movement of registers across combinational logic, allowing the Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths.
Perform Physical synthesis for combinational logic for Performance (no Intel Arria 10 support)	Performs physical synthesis optimizations on combinational logic during synthesis and fitting to increase circuit performance. Swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. Also allows the duplication of LUTs to enable further optimizations on the critical path.
Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support)	Reduces delay along critical paths. This option swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. The option also allows the duplication of LUTs to enable further optimizations on the critical path. The option causes registers that do not have a Power-Up Level logic option setting to power up with a don't care logic level (x). When the Power-Up Don't Care option is turned on, the Compiler determines when it is beneficial to change the power-up level of a register to minimize the area of the design. A power-up state of zero is maintained unless there is an immediate area advantage. The registers contained in the affected logic cells are not modified. Inputs into memory blocks, DSP blocks, and I/O elements

continued...

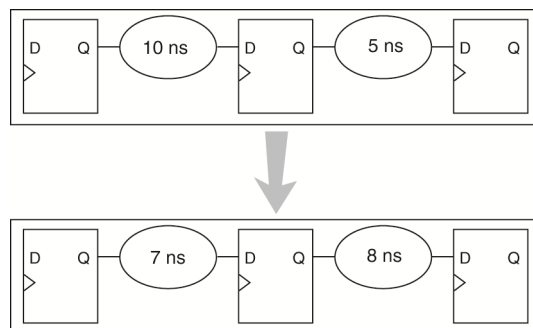


Option	Description
	(IOEs) are not swapped. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow .
Perform WYSIWYG Primitive Resynthesis	Specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the Optimization Technique logic option.
Physical Synthesis Effort Level (no Intel Arria 10 support)	Specifies the amount of effort, in terms of compile time, physical synthesis should use. Compared to the Default setting, a setting of Extra uses extra compile time to try to gain extra circuit performance. Conversely, a setting of Fast uses less compile time but may reduce the performance gain that physical synthesis is able to achieve.
Netlist Optimizations	You can use the Assignment Editor to apply the Netlist Optimizations logic option. Use this option to disable physical synthesis optimizations for parts of your design.
Allow Register Duplication	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.
Allow Register Merging	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.

6.1.3. Perform Register Retiming for Performance

The **Perform Register Retiming for Performance** option enables the movement of registers across combinational logic, allowing the Intel Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths. Register retiming can be done during Intel Quartus Prime integrated synthesis or during the Fitter stages of design compilation.

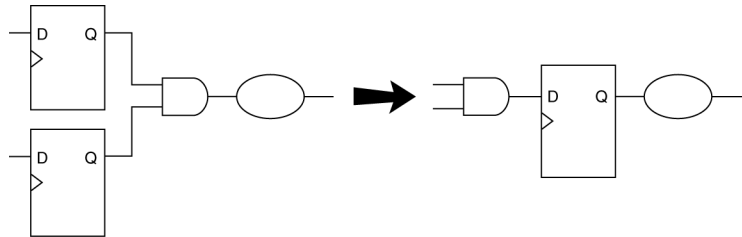
Figure 49. Reducing Critical Delay by Moving the Register Relative to Combinational Logic



Retiming can create multiple registers at the input of a combinational block from a register at the output of a combinational block. In this case, the new registers have the same clock and clock enable. The asynchronous control signals and power-up level

are derived from previous registers to provide equivalent functionality. Retiming can also combine multiple registers at the input of a combinational block to a single register.

Figure 50. Combining Registers with Register Retiming



To move registers across combinational logic to balance timing, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**. Specify your preferred option under **Optimize for performance (physical synthesis)** and **Effort level**.

6.1.4. Preventing Register Movement During Retiming

If you want to prevent register movement during register retiming, you can set the **Netlist Optimizations** logic option to **Never Allow**. You can apply this option to either individual registers or entities in the design using the Assignment Editor.

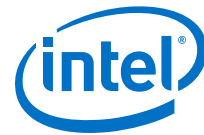
In digital circuits, synchronization registers are instantiated on cross clock domain paths to reduce the possibility of metastability. The Intel Quartus Prime software detects such synchronization registers and does not move them, even if register retiming is turned on.

The following sets of registers are not moved during register retiming:

- Both registers in a direct connection from input pin-to-register-to-register if both registers have the same clock and the first register does not fan-out to anywhere else. These registers are considered synchronization registers.
- Both registers in a direct connection from register-to-register if both registers have the same clock, the first register does not fan out to anywhere else, and the first register is fed by another register in a different clock domain (directly or through combinational logic). These registers are considered synchronization registers.

The Intel Quartus Prime software does not perform register retiming on logic cells that have the following properties:

- Are part of a cascade chain
- Contain registers that drive asynchronous control signals on another register
- Contain registers that drive the clock of another register
- Contain registers that drive a register in another clock domain
- Contain registers that are driven by a register in another clock domain



Note: The Intel Quartus Prime software does not usually retime registers across different clock domains; however, if you use the Classic Timing Analyzer and specify a global f_{MAX} requirement, the Intel Quartus Prime software interprets all clocks as related. Consequently, the Intel Quartus Prime software might try to retime register-to-register paths associated with different clocks.

To avoid this circumstance, provide individual f_{MAX} requirements to each clock when using Classic Timing Analysis. When you constrain each clock individually, the Intel Quartus Prime software assumes no relationship between different clock domains and considers each clock domain to be asynchronous to other clock domains; hence no register-to-register paths crossing clock domains are retimed.

When you use the Timing Analyzer, register-to-register paths across clock domains are never retimed, because the Timing Analyzer treats all clock domains as asynchronous to each other unless they are intentionally grouped.

- Contain registers that are constrained to a single LAB location
- Contain registers that are connected to SERDES
- Are considered virtual I/O pins
- Registers that have the **Netlist Optimizations** logic option set to **Never Allow**

The Intel Quartus Prime software assumes that a synchronization register chain consists of two registers. If your design has synchronization register chains with more than two registers, you must indicate the number of registers in your synchronization chains so that they are not affected by register retiming. To do this, perform the following steps:

1. Click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)**.
2. Modify the **Synchronization Register Chain Length** setting to match the synchronization register length used in your design. If you set a value of 1 for the **Synchronization Register Chain Length**, it means that any registers connected to the first register in a register-to-register connection can be moved during retiming. A value of $n > 1$ means that any registers in a sequence of length 1, 2, ... n are not moved during register retiming.

If you want to consider logic cells that meet any of these conditions for physical synthesis, you can override these rules by setting the **Netlist Optimizations** logic option to **Always Allow** on a given set of registers.

Related Information

[Analyzing and Optimizing the Design Floorplan](#) on page 103

6.2. 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.

Applying all the physical synthesis options at the **Extra** effort level generally produces the best results for those options, but adds significantly to the compilation time. You can also use the **Physical synthesis effort level** options to decrease the compilation time. The WYSIWYG primitive resynthesis option does not add much compilation time relative to the overall design compilation time.

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

[Design Space Explorer II](#) on page 12

6.2.1. 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/cascade 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.

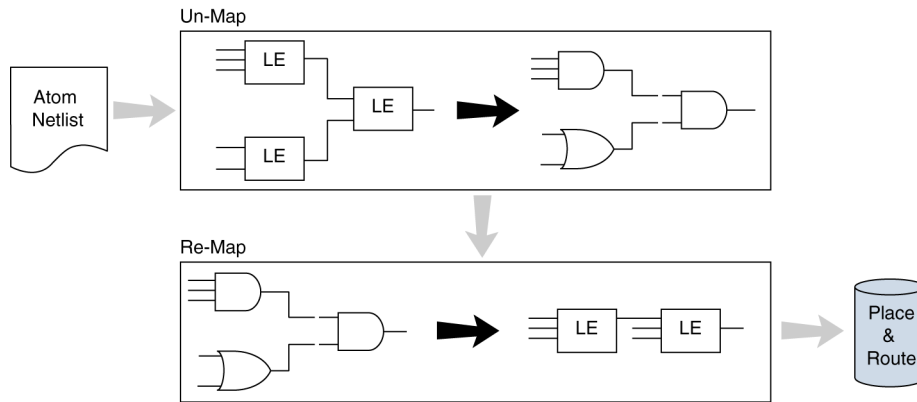
Any nodes or entities that have the **Netlist Optimizations** logic option set to **Never Allow** are not affected during WYSIWYG primitive resynthesis. You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. This option disables WYSIWYG resynthesis for parts of your design.

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 51. Intel Quartus Prime Flow for WYSIWYG Primitive Resynthesis



6.2.2. Saving a Node-Level Netlist

For non-Intel Arria 10 designs, you can preserve a node-level netlist in Verilog Quartus Mapping File (`.vqm`) format. You might need to preserve nodes if you use the Logic Lock (Standard) flow to back-annotate placement, import one design into another, or both. For all device families that support incremental compilation, you can use this feature to preserve compilation results.

Note: This feature does not support Intel Arria 10 devices.

Use the **Export version-compatible database** option to save synthesis results as an atom-based netlist in `.vqm` file format. By default, the Intel Quartus Prime software places the `.vqm` in the **atom_netlists** directory under the current project directory.

If you use the physical synthesis optimizations and want to lock down the location of all LEs and other device resources in the design with the **Back-Annotate Assignments** command, a `.vqm` file netlist is required. The `.vqm` file preserves the changes that you made to your original netlist. Because the physical synthesis optimizations depend on the placement of the nodes in the design, back-annotating the placement changes the results from physical synthesis. Changing the results means that node names are different, and your back-annotated locations are no longer valid.

You should not use an Intel Quartus Prime-generated `.vqm` file or back-annotated location assignments with physical synthesis optimizations unless you have finalized the design. Making any changes to the design invalidates your physical synthesis results and back-annotated location assignments. If you require changes later, use the new source HDL code as your input files, and remove the back-annotated assignments corresponding to the Intel Quartus Prime-generated `.vqm` file.

To back-annotate logic locations for a design that was compiled with physical synthesis optimizations, first create a `.vqm` file. When recompiling the design with the hard logic location assignments, use the new `.vqm` file as the input source file and turn off the physical synthesis optimizations for the new compilation.

If you are importing a .vqm file and back-annotated locations into another project that has any **Netlist Optimizations** turned on, you must apply the **Never Allow** constraint to make sure node names don't change; otherwise, the back-annotated location or Logic Lock (Standard) assignments are invalid.

To preserve the nodes from Intel Quartus Prime physical synthesis optimization options for devices that do not support incremental compilation, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Compilation Process Settings**. The **Compilation Process Settings** page appears.
3. Turn on **Export version-compatible database**. This setting is not available for some devices.
4. Click **OK**.

6.3. Viewing Synthesis and Netlist Optimization Reports

Physical synthesis optimizations performed during synthesis write results to the synthesis report. To access this report, perform the following steps:

1. On the Processing menu, click **Compilation Report**.
2. In the **Compilation Report** list, open the **Analysis & Synthesis** folder to view synthesis results.
3. In the **Compilation Report** list, open the **Fitter** folder to view the **Netlist Optimizations** table.

6.4. 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 Standard Edition Settings File Reference Manual](#)
For information about all settings and constraints in the Intel Quartus Prime software.

6.4.1. 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 23. 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	BALANCEDHIGH PERFORMANCE EFFOR AGGRESSIVE PERFORMANCE	Global, Instance
Power-Up Don't Care	ALLOW_POWER_UP_DONT_CARE	ON, OFF	Global
Save a node-level netlist into a persistent source file	LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT	ON, OFF	Global
	LOGICLOCK_INCREMENTAL_COMPILE_FILE	<file name>	
Allow Netlist Optimizations	ADV_NETLIST_OPT_ALLOWED	"ALWAYS ALLOW", DEFAULT, "NEVER ALLOW"	Instance

6.4.2. 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 24. Physical Synthesis Optimizations and Associated Settings

Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Perform Physical Synthesis for Combinational Logic for Performance (no Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC	ON, OFF	Global
Perform Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global

continued...



Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Advanced Physical Synthesis	ADVANCED_PHYSICAL_SYNTHESIS	ON, OFF	Global
Automatic Asynchronous Signal Pipelining	PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING	ON, OFF	Global
Perform Register Duplication for Performance (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION	ON, OFF	Global
Perform Register Retiming for Performance (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_RETIMING	ON, OFF	Global
Power-Up Don't Care	ALLOW_POWER_UP_DONT_CARE	ON, OFF	Global, Instance
Power-Up Level	POWER_UP_LEVEL	HIGH, LOW	Instance
Allow Netlist Optimizations	ADV_NETLIST_OPT_ALLOWED	"ALWAYS ALLOW", DEFAULT, "NEVER ALLOW"	Instance
Save a node-level netlist into a persistent source file (no Intel Arria 10 support)	LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT	ON, OFF	Global
	LOGICLOCK_INCREMENTAL_COMPILE_FILE	<file name>	

6.4.3. Back-Annotating Assignments

You can use the `logiclock_back_annotate` Tcl command to back-annotate resources in your design. This command can back-annotate resources in Logic Lock (Standard) regions, and resources in designs without Logic Lock (Standard) regions.

The following Tcl command back-annotates all registers in your design:

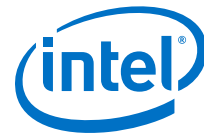
```
logiclock_back_annotate -resource_filter "REGISTER"
```

The `logiclock_back_annotate` command is in the `backannotate` package.

6.5. Netlist Optimizations and Physical Synthesis Revision History

The following revision history applies to this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.
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"> Added topic: <i>Isolating a Partition Netlist</i>.
2016.10.31	16.1.0	<ul style="list-style-type: none"> Updated physical synthesis options and procedure.
2016.05.02	16.0.0	<ul style="list-style-type: none"> Stated limitations about deprecated physical synthesis options.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
<i>continued...</i>		

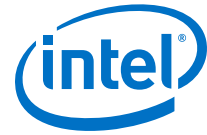


Document Version	Intel Quartus Prime Version	Changes
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations Settings to Compiler Settings. Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Removed HardCopy device information.
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"> Added links to Intel Quartus Prime Help in several sections. Removed Referenced Documents section. Reformatted Document Revision History
November 2009	9.1.0	<ul style="list-style-type: none"> Added information to "Physical Synthesis for Registers—Register Retiming" Added information to "Applying Netlist Optimization Options" Made minor editorial updates
March 2009	9.0.0	<ul style="list-style-type: none"> Was chapter 11 in the 8.1.0 release. Updated the "Physical Synthesis for Registers—Register Retiming" and "Physical Synthesis Options for Fitting" Updated "Performing Physical Synthesis Optimizations" Deleted Gate-Level Register Retiming section. Updated the referenced documents
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"> Updated "Physical Synthesis Optimizations for Performance on page 11-9" Added Physical Synthesis Options for Fitting on page 11-16

Related Information

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.



7. Engineering Change Orders with the Chip Planner

Programmable logic can accommodate changes to a system specification late in the design cycle. In a typical engineering project development cycle, the specification of the programmable logic portion is likely to change after engineering begins or while integrating all system elements. Last-minute design changes, commonly referred to as engineering change orders (ECOs), are small targeted changes to the functionality of a design after the design has been fully compiled.

The Chip Planner supports ECOs by allowing quick and efficient changes to your logic late in the design cycle for supported devices. The Chip Planner provides a visual display of your post-place-and-route design mapped to the device architecture of your chosen FPGA and allows you to create, move, and delete logic cells and I/O atoms.

Note: The Intel Quartus Prime Standard Edition ECO feature does not support Intel Arria 10 devices.

In addition to making ECOs, the Chip Planner allows you to perform detailed analysis on routing congestion, relative resource usage, logic placement, Logic Lock (Standard) regions, fan-ins and fan-outs, paths between registers, and delay estimates for paths.

ECOs directly apply to atoms in the supported target device. As such, performing an ECO relies on your understanding of the device architecture of the target device.

Related Information

- [Analyzing and Optimizing the Design Floorplan](#) on page 103
For more information about using the Chip Planner for design analysis
- [Literature](#)
For more information about the architecture of your device

7.1. Engineering Change Orders

In the context of an FPGA design, you can apply an ECO directly to a physical resource on the device to modify its behavior. ECOs are typically made during the verification stage of a design cycle. When a small change is required on a design (such as modifying a PLL for a different clock frequency or routing a signal out to a pin for analysis) recompilation of the entire design can be time consuming, especially for larger designs.

Because several iterations of small design changes can occur during the verification cycle, recompilation times can quickly add up. Furthermore, a full recompilation due to a small design change can result in the loss of previous design optimizations. Making ECOs, instead of performing a full recompilation on your design, limits the change only to the affected portions of logic.



7.1.1. Performance Preservation

You can preserve the results of previous design optimizations when you make changes to an existing design with one of the following methods:

- Incremental compilation
- Rapid recompile
- ECOs

Choose the method to modify your design based on the scope of the change. The methods above are arranged from the larger scale change to the smallest targeted change to a compiled design.

The incremental compilation feature allows you to preserve compilation results at an RTL component or module level. After the initial compilation of your design, you can assign modules in your design hierarchy to partitions. Upon subsequent compilations, incremental compilation recompiles changed partitions based on the chosen preservation levels.

The rapid recompilation feature leverages results from the latest post-fit netlist to determine the changes required to honor modifications you have made to the source code. If you run a rapid recompilation, the Compiler refits only changed portion of the netlist.

ECOs provide a finer granularity of control compared to the incremental compilation and the rapid recompilation feature. All modifications are performed directly on the architectural elements of the device. You should use ECOs for targeted changes to the post-fit netlist.

Note: In the Intel Quartus Prime software versions 10.0 and later, the software does not preserve ECO modifications to the netlist when you recompile a design with the incremental compilation feature turned on. You can reapply ECO changes made during a previous compilation with the Change Manager.

Related Information

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design](#)

7.1.2. Compilation Time

In the traditional programmable logic design flow, a small change in the design requires a complete recompilation of the design. A complete recompilation of the design consists of synthesis and place-and-route. Making small changes to the design to reach the final implementation on a board can be a long process. Because the Chip Planner works only on the post-place-and-route database, you can implement your design changes in minutes without performing a full compilation.

7.1.3. Verification

After you make a design change, you can verify the impact on your design. To verify that your changes do not violate timing requirements, perform static timing analysis with the Intel Quartus Prime Timing Analyzer after you check and save your netlist changes in the Chip Planner.



Additionally, you can perform a gate-level or timing simulation of the ECO-modified design with the post-place-and-route netlist generated by the Intel Quartus Prime software.

Related Information

[Intel Quartus Prime Timing Analyzer](#)

7.1.4. Change Modification Record

All ECOs made with the Chip Planner are logged in the Change Manager to track all changes. With the Change Manager, you can easily revert to the original post-fit netlist or you can pick and choose which ECOs to apply.

Additionally, the Intel Quartus Prime software provides support for multiple compilation revisions of the same project. You can use ECOs made with the Chip Planner in conjunction with revision support to compare several different ECO changes and revert back to previous project revisions when required.

7.2. ECO Design Flow

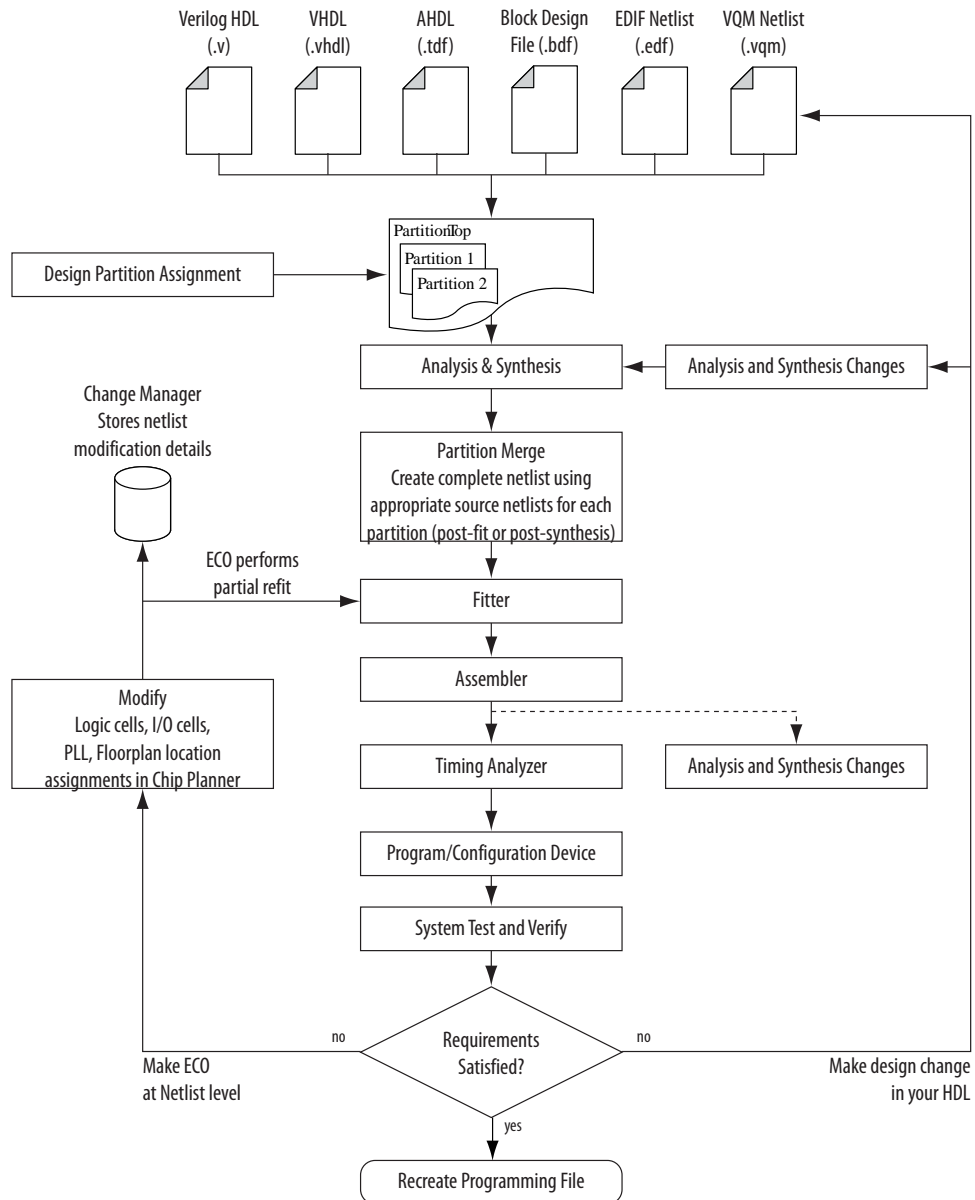
For iterative verification cycles, implementing small design changes at the netlist level can be faster than making an RTL code change. As such, making ECO changes are especially helpful when you debug the design on silicon and require a fast turnaround time to generate a programming file for debugging the design.

Note: The Intel Quartus Prime Standard Edition ECO feature does not support Intel Arria 10 devices.

The figure shows the design flow for making ECOs.



Figure 52. Design Flow to Support ECOs



A typical ECO application occurs when you uncover a problem on the board and isolate the problem to the appropriate nodes or I/O cells on the device. You must be able to correct the functionality quickly and generate a new programming file. By making small changes with the Chip Planner, you can modify the post-place-and-route netlist directly without having to perform synthesis and logic mapping, thus decreasing the turnaround time for programming file generation during the verification cycle. If the

change corrects the problem, no modification of the HDL source code is necessary. You can use the Chip Planner to perform the following ECO-related changes to your design:

- Document the changes made with the Change Manager
- Easily recreate the steps taken to produce design changes
- Generate EDA simulation netlists for design verification

Note: For more complex changes that require HDL source code modifications, the incremental compilation feature can help reduce recompilation time.

7.3. The Chip Planner Overview

The Chip Planner provides a visual display of device resources. It shows the arrangement and usage of the resource atoms in the device architecture that you are targeting. Resource atoms are the building blocks for your device, such as ALMs, LEs, PLLs, DSP blocks, memory blocks, or I/O elements.

The Chip Planner also provides an integrated platform for design analysis and for making ECOs to your design after place-and-route. The toolset consists of the Chip Planner (providing a device floorplan view of your mapped design) and two integrated subtools—the Resource Property Editor and the Change Manager.

For analysis, the Chip Planner can show logic placement, Logic Lock (Standard) regions, relative resource usage, detailed routing information, routing congestion, fan-ins and fan-outs, paths between registers, and delay estimates for paths. Additionally, the Chip Planner allows you to create location constraints or resource assignment changes, such as moving or deleting logic cells or I/O atoms with the device floorplan. For ECO changes, the Chip Planner enables you to create, move, or delete logic cells in the post-place-and-route netlist for fast programming file generation. Additionally, you can open the Resource Property Editor from the Chip Planner to edit the properties of resource atoms or to edit the connections between resource atoms. All changes to resource atoms and connections are logged automatically with the Change Manager.

7.3.1. Opening the Chip Planner

To open the Chip Planner, on the Tools menu, click **Chip Planner**. Alternatively, click the **Chip Planner** icon on the Intel Quartus Prime software toolbar.

Optionally, you can open the Chip Planner by cross-probing from the shortcut menu in the following tools:

- Design Partition Planner
- Compilation Report
- Logic Lock (Standard) Regions window
- Technology Map Viewer
- Project Navigator window
- RTL source code
- Node Finder



- Simulation Report
- RTL Viewer
- Report Timing panel of the Timing Analyzer

7.3.2. The Chip Planner Tasks and Layers

The Chip Planner allows you to set up tasks to quickly implement ECO changes or manipulate assignments for the floorplan of the device. Each task consists of an editing mode and a set of customized layer settings.

Related Information

- [Performing ECOs in the Resource Property Editor](#) on page 147
- [Analyzing and Optimizing the Design Floorplan](#) on page 103

7.4. Performing ECOs with the Chip Planner (Floorplan View)

You can manipulate resource atoms in the Chip Planner when you select the ECO editing mode.

The following ECO changes can be made with the Chip Planner Floorplan view:

- Create atoms
- Delete atoms
- Move existing atoms

Note: To configure the properties of atoms, such as managing the connections between different LEs/ALMs, use the Resource Property Editor.

To select the ECO editing mode in the Chip Planner, in the **Editing Mode** list at the top of the Chip Planner, select the ECO editing mode.

Related Information

[Performing ECOs in the Resource Property Editor](#) on page 147

7.4.1. Creating, Deleting, and Moving Atoms

You can use the Chip Planner to create, delete, and move atoms in the post-compilation design.

7.4.2. Check and Save Netlist Changes

After making all the ECOs, you can run the Fitter to incorporate the changes by clicking the **Check and Save Netlist Changes** icon in the Chip Planner toolbar. The Fitter compiles the ECO changes, performs design rule checks on the design, and generates a programming file.

7.5. Performing ECOs in the Resource Property Editor

You can view and edit the following resources with the Resource Property Editor.

7.5.1. Logic Elements

An Altera® LE contains a four-input LUT, which is a function generator that can implement any function of four variables. In addition, each LE contains a register fed by the output of the LUT or by an independent function generated in another LE.

You can use the Resource Property Editor to view and edit any LE in the FPGA. To open the Resource Property Editor for an LE, on the Project menu, point to **Locate**, and then click **Locate in Resource Property Editor** in one of the following views:

- RTL Viewer
- Technology Map Viewer
- Node Finder
- Chip Planner

For more information about LE architecture for a particular device family, refer to the device family handbook or data sheet.

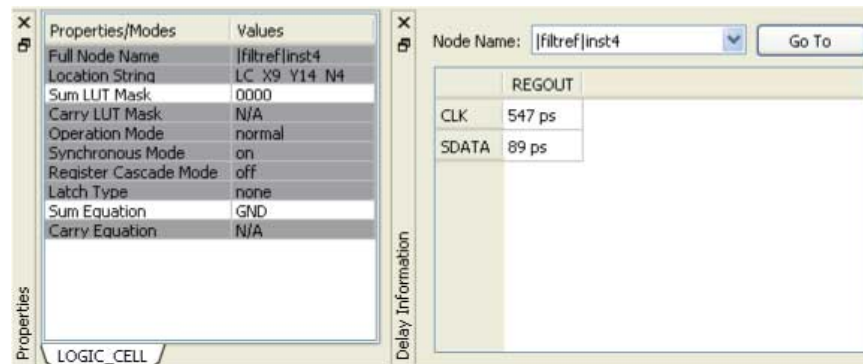
You can use the Resource Property Editor to change the following LE properties:

- Data input to the LUT
- LUT mask or LUT

7.5.1.1. Logic Element Properties

To view logic element properties, on the View menu, click **View Properties**.

Figure 53. LE Properties in the Resource Property Editor



7.5.1.2. Modes of Operation

LUTs in an LE can operate in either normal or arithmetic mode.

When an LE is configured in normal mode, the LUT in the LE can implement a function of four inputs.

When the LE is configured in arithmetic mode, the LUT in the LE is divided into two 3-input LUTs. The first LUT generates the signal that drives the output of the LUT, while the second LUT generates the carry-out signal. The carry-out signal can drive only a carry-in signal of another LE.

For more information about LE modes of operation, refer to volume 1 of the appropriate device handbook.



7.5.1.3. Sum and Carry Equations

You can change the logic function implemented by the LUT by changing the sum and carry equations. When the LE is configured in normal mode, you can change only the sum equation. When the LE is configured in arithmetic mode, you can change both the sum and the carry equations.

The LUT mask is the hexadecimal representation of the LUT equation output. When you change the LUT equation, the Intel Quartus Prime software automatically changes the LUT mask. Conversely, when you change the LUT mask, the Intel Quartus Prime software automatically computes the LUT equation.

7.5.1.4. sload and sclr Signals

Each LE register contains a synchronous load (`sload`) signal and a synchronous clear (`sclr`) signal. You can invert either the `sload` or `sclr` signal feeding into the LE.

If the design uses the `sload` signal in an LE, the signal and its inversion state must be the same for all other LEs in the same LAB. For example, if two LEs in a LAB have the `sload` signal connected, both LEs must have the `sload` signal set to the same value. This is also true for the `sclr` signal.

7.5.1.5. Register Cascade Mode

When register cascade mode is enabled, the cascade-in port feeds the input to the register. The register cascade mode is used most often when the design implements shift registers.

You can change the register cascade mode by connecting (or disconnecting) the cascade in the port. However, if you create this port, you must ensure that the source port LE is directly above the destination LE.

7.5.1.6. Cell Delay Table

The cell delay table describes the propagation delay from all inputs to all outputs for the selected LE.

7.5.1.7. Logic Element Connections

To view the connections that feed in and out of an LE, on the View menu, click **View Port Connections**.

Figure 54. View LE Connections in the Connectivity Window

Input Port Name	Signal Name	Inverted	Output Port Name	Signal Name
DATAA	<Disconnected>	False	COUT	<Disconnected>
DATAB	<Disconnected>	False	COMBOUT	<Disconnected>
SDATA	filter state_m:inst1 filter.tap4	False	REGOUT	filter inst4
DATAD	<Disconnected>	False		
CIN	<Disconnected>	False		
INVERTA	<Disconnected>	False		
REGCASCIN	<Disconnected>	False		
SLOAD	VCC	False		
SCLR	<Disconnected>	False		
IACLAR	VCC	False		
ALOAD	<Disconnected>	False		
CLK	filter clk	False		
ENA	<Disconnected>	False		

7.5.1.8. Deleting a Logic Element

To delete an LE, follow these steps:

1. Right-click the desired LE in the Chip Planner, point to **Locate**, and click **Locate in Resource Property Editor**.
2. You must remove all fan-out connections from an LE prior to deletion. To delete fan-out connections, right-click each connected output signal, point to **Remove**, and click **Fanouts**. Select all of the fan-out signals in the **Remove Fan-outs** dialog box and click **OK**.
3. To delete an atom after all fan-out connections are removed, right-click the atom in the Chip Planner and click **Delete Atom**.

7.5.2. Adaptive Logic Modules

Each ALM contains LUT-based resources that can be divided between two adaptive LUTs (ALUTs).

With up to eight inputs to the two ALUTs, each ALM can implement various combinations of two functions. This adaptability allows the ALM to be completely backward-compatible with four-input LUT architectures. One ALM can implement any function with up to six inputs and certain seven-input functions. In addition to the ALUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. The ALM can efficiently implement various arithmetic functions and shift registers with these dedicated resources.

You can implement the following types of functions in a single ALM:

- Two independent 4-input functions
- An independent 5-input function and an independent 3-input function
- A 5-input function and a 4-input function, if they share one input
- Two 5-input functions, if they share two inputs
- An independent 6-input function
- Two 6-input functions, if they share four inputs and share the same functions
- Certain 7-input functions

You can use the Resource Property Editor to change the following ALM properties:

- Data input to the LUT
- LUT mask or LUT equation

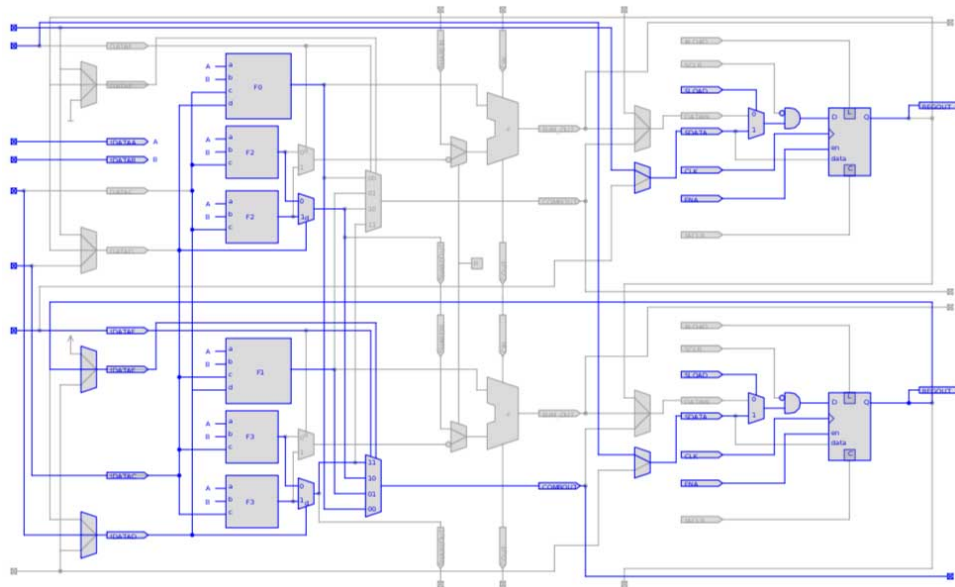
7.5.2.1. Adaptive Logic Module Schematic

You can view and edit any ALM atom with the Resource Property Editor by right-clicking the ALM in the RTL Viewer, the Node Finder, or the Chip Planner, and clicking **Locate in Resource Property Editor**.

For a detailed description of the ALM, refer to the device handbooks of devices based on an ALM architecture.

By default, the Intel Quartus Prime software displays the used resources in blue and the unused in gray. For the figure, the used resources are in blue and the unused resources are in gray.

Figure 55. Adaptive Logic Module



7.5.2.2. Adaptive Logic Module Properties

The properties that you can display for the ALM include an equations table that shows the name and location of each of the two combinational nodes and two register nodes in the ALM, the individual LUT equations for each of the combinational nodes, and the `combout`, `sumout`, `carryout`, and `shareout` equations for each combinational node.

7.5.2.3. Adaptive Logic Module Connections

Click **View > View Connectivity** to view the input and output connections for the ALM.

7.5.3. FPGA I/O Elements

Altera FPGAs that have high-performance I/O elements, including up to six registers, are equipped with support for a number of I/O standards that allow you to run your design at peak speeds. Use the Resource Property Editor to view, change connectivity, and edit the properties of the I/O elements. Use the Chip Planner (Floorplan view) to change placement, delete, and create new I/O elements.

For a detailed description of the device I/O elements, refer to the applicable device handbook.

You can change the following I/O properties:

- Delay chain
- Bus hold
- Weak pull up
- Slow slew rate

- I/O standard
- Current strength
- Extend OE disable
- PCI I/O
- Register reset mode
- Register synchronous reset mode
- Register power up
- Register mode

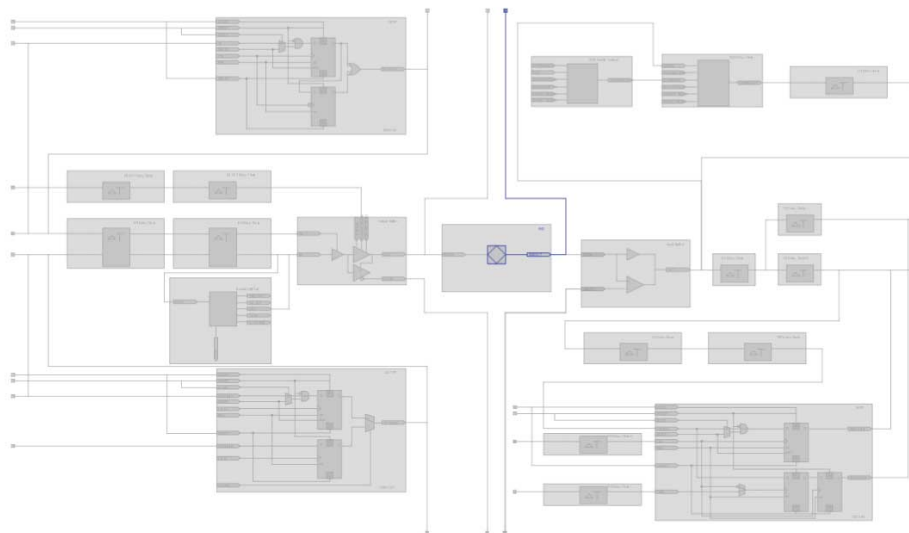
7.5.3.1. Stratix V I/O Elements

The I/O elements in Stratix® V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate (SDR) or double data rate (DDR) transfer.

I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal to the output buffer. These registers allow faster source-synchronous register-to-register transfers and resynchronization. The input path consists of the DDR input registers, alignment and synchronization registers, and half data rate blocks; you can bypass each block in the input path. The input path uses the deskew delay to adjust the input register clock delay across process, voltage, and temperature (PVT) variations.

By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 56. Stratix V Device I/O Element Structure



Related Information

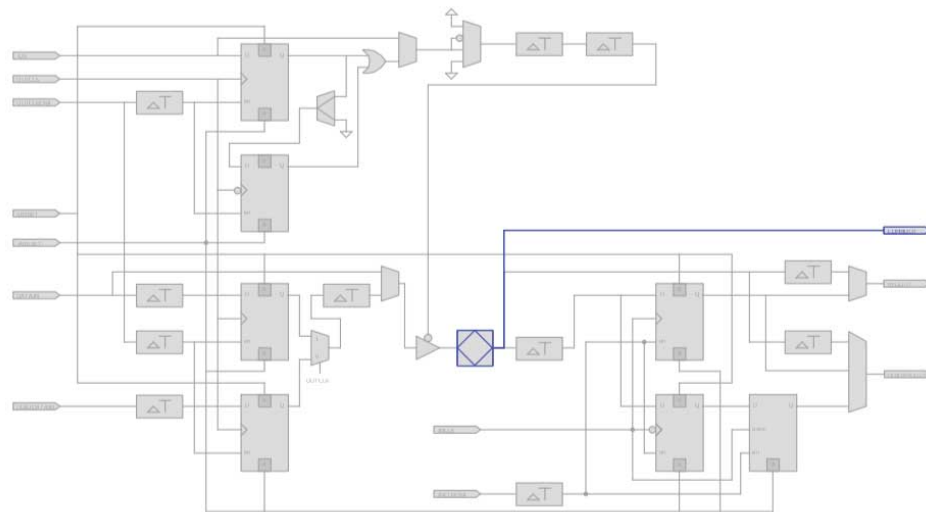
[Stratix V Device Handbook](#)

7.5.3.2. Stratix IV I/O Elements

The I/O elements in Stratix IV devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional SDR or DDR transfer.

The I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal for the output buffer. Each path consists of a set of delay elements that allow you to fine-tune the timing characteristics of each path for skew management. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 57. Stratix IV I/O Element and Structure



Related Information

Literature

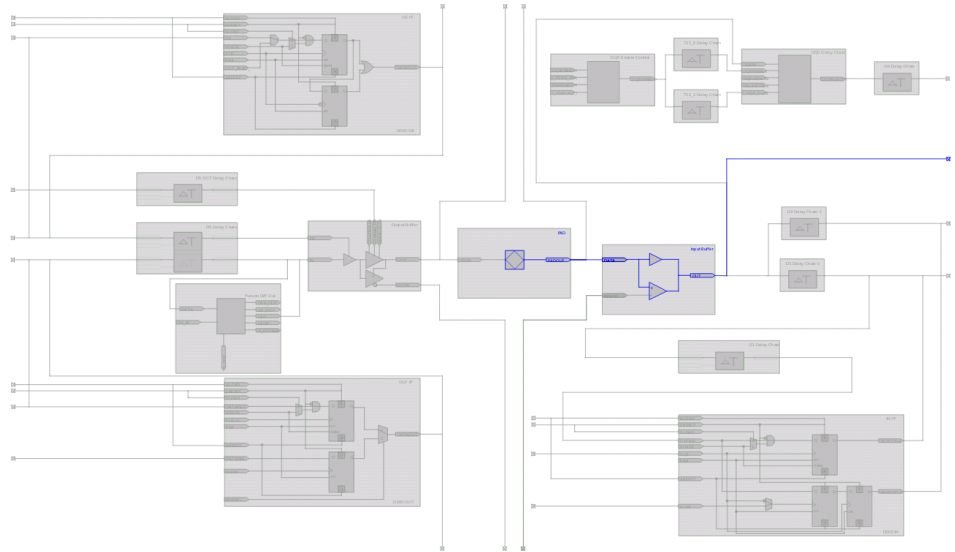
For more information about I/O elements in Stratix IV devices

7.5.3.3. Arria V I/O Elements

The I/O elements in Arria[®] V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional SDR or DDR transfer.

The I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal for the output buffer. Each path consists of a set of delay elements that allow you to fine-tune the timing characteristics of each path for skew management. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 58. Arria V Device I/O Element and Structure

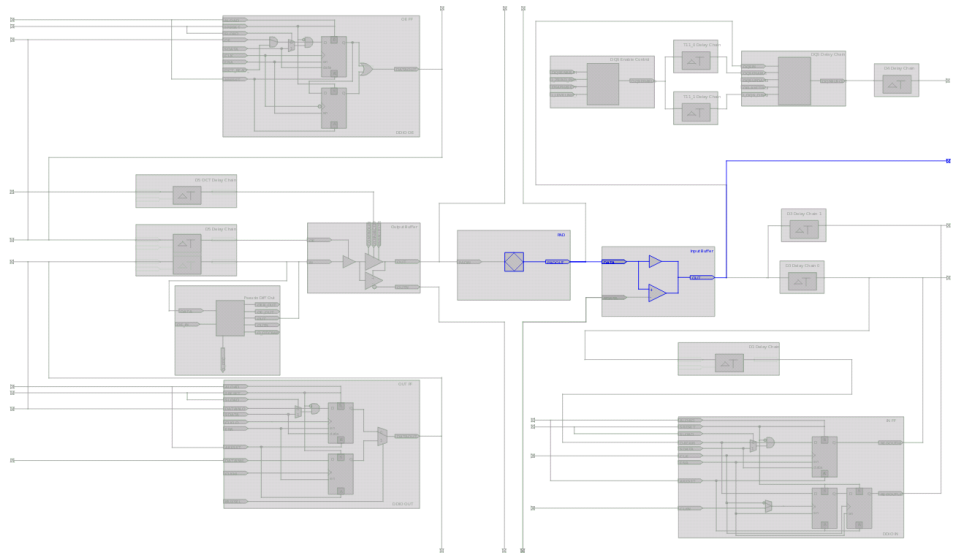


7.5.3.4. Cyclone V I/O Elements

The I/O elements in Cyclone V devices contain a bidirectional I/O buffer and registers for complete embedded bidirectional single data rate transfer. The I/O element contains three input registers, two output registers, and two output-enable registers. The two output registers and two output-enable registers are utilized for double-data rate (DDR) applications.

You can use the input registers for fast setup times and the output registers for fast clock-to-output times. Additionally, you can use the output-enable (OE) registers for fast clock-to-output enable timing. You can use I/O elements for input, output, or bidirectional data paths. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

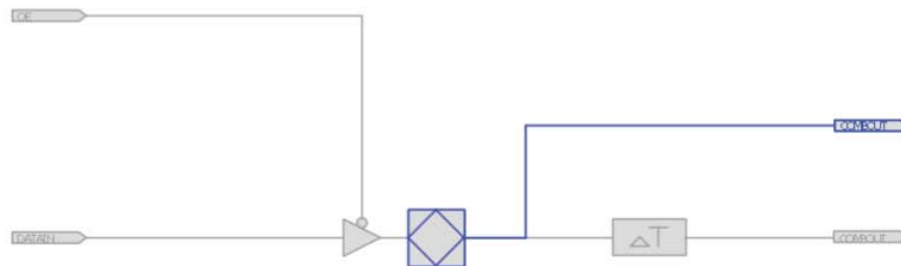
Figure 59. Cyclone V Device I/O Elements and Structure



7.5.3.5. MAX V I/O Elements

The I/O elements in MAX[®] V devices contain a bidirectional I/O buffer. You can drive registers from adjacent LABs to or from the bidirectional I/O buffer of the I/O element. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

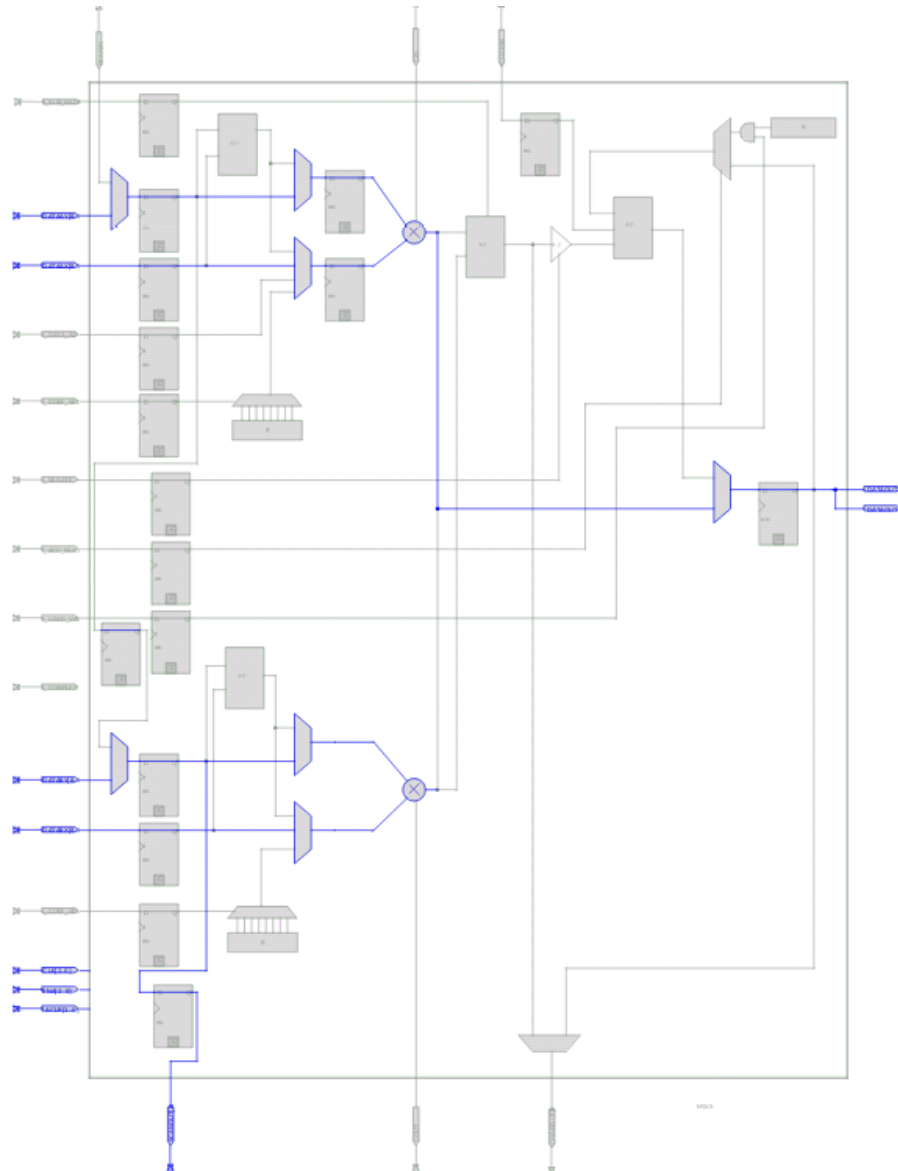
Figure 60. MAX V Device I/O Elements and Structure



7.5.4. FPGA RAM Blocks

With the Resource Property Editor, you can view the architecture of different RAM blocks in the device, modify the input and output registers to and from the RAM blocks, and modify the connectivity of the input and output ports. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 61. M9K RAM View in a Stratix V Device

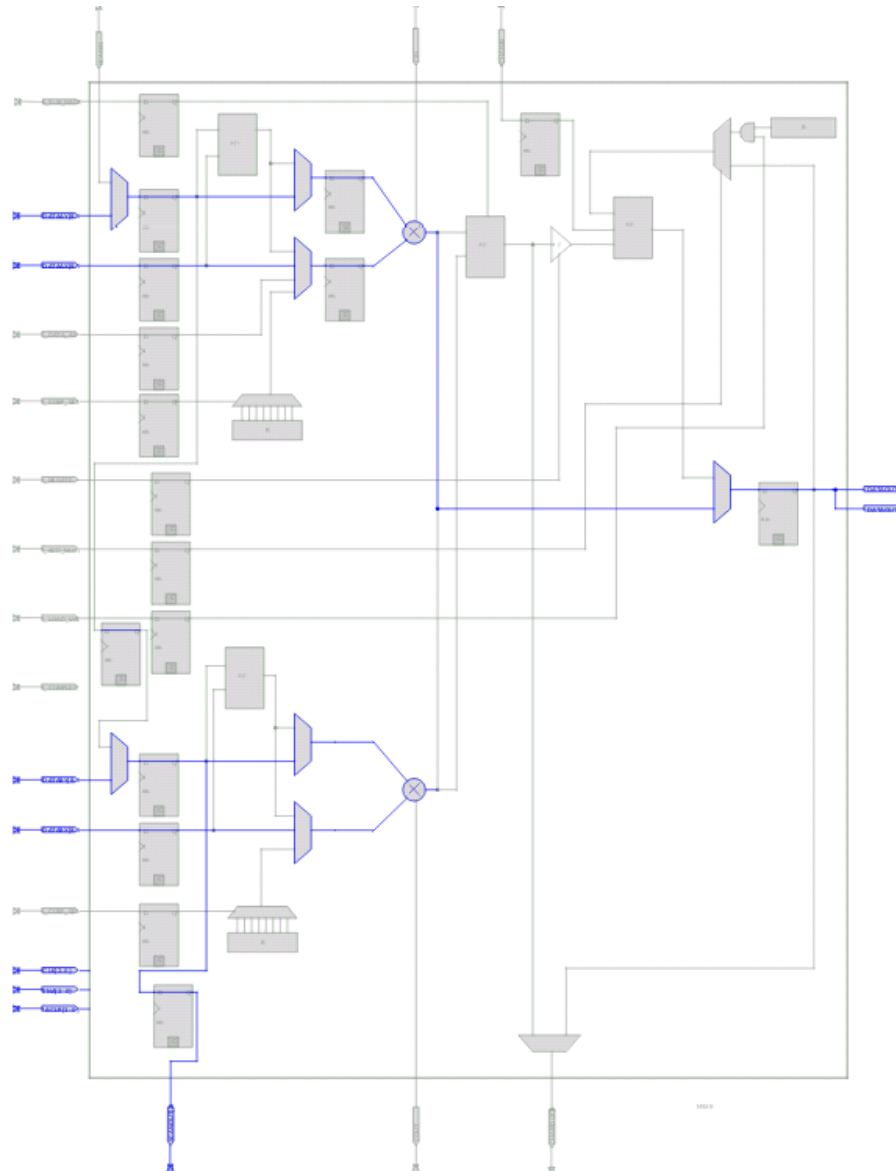


7.5.5. FPGA DSP Blocks

Dedicated hardware DSP circuit blocks in Altera devices provide performance benefits for the critical DSP functions in your design.

The Resource Property Editor allows you to view the architecture of DSP blocks in the Resource Property Editor for the Cyclone and Stratix series of devices. The Resource Property Editor also allows you to modify the signal connections to and from the DSP blocks and modify the input and output registers to and from the DSP blocks. By default, the [Intel Quartus Prime](#) software displays the used resources in blue and the unused resources in gray.

Figure 62. DSP Block View in a Stratix V Device



7.6. Change Manager

The Change Manager maintains a record of every change you perform with the Chip Planner, the Resource Property Editor, the Signal Probe feature, or a Tcl script. Each row of data in the Change Manager represents one ECO.

The Change Manager allows you to apply changes, roll back changes, delete changes, and export change records to a Text File (**.txt**), a Comma-Separated Value File (**.csv**), or a Tcl Script File (**.tcl**). The Change Manager tracks dependencies between changes, so that when you apply, roll back, or delete a change, any prerequisite or dependent changes are also applied, rolled back, or deleted.

7.6.1. Complex Changes in the Change Manager

Certain changes in the Change Manager (including creating or deleting atoms and changing connectivity) can appear to be self-contained, but are actually composed of multiple actions. The Change Manager marks such complex changes with a plus icon in the **Index** column.

You can click the plus icon to expand the change record and show all the component actions performed as part of that complex change.

Related Information

[Example of Managing Changes With the Change Manager](#)

7.6.2. Managing Signal Probe Signals

The Signal Probe pins that you create from the **Signal Probe Pins** dialog box are recorded in the Change Manager. After you have made a Signal Probe assignment, you can use the Change Manager to quickly disable Signal Probe assignments by selecting **Revert to Last Saved Netlist** on the shortcut menu in the Change Manager.

Related Information

[Quick Design Debugging Using Signal Probe](#)

7.6.3. Exporting Changes

You can export changes to a **.txt**, a **.csv**, or a **.tcl**. Tcl scripts allow you to reapply changes that were deleted during compilation.

Related Information

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design](#)

7.7. 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. The Tcl commands for controlling the Chip Planner are located in the `chip_planner` package of the `quartus_cdb` executable.

Related Information

- [About Intel Quartus Prime Scripting](#)
- [Tcl Scripting](#)
- [Intel Quartus Prime Settings File Manual](#)
- [Command Line Scripting](#)

7.8. Common ECO Applications

You can use an ECO to make a post-compilation change to your design.



To help build your system quickly, you can use Chip Planner functions to perform the following activities:

- Adjust the drive strength of an I/O with the Chip Planner
- Modify the PLL properties with the Resource Property Editor, see ["Modify the PLL Properties With the Chip Planner"](#)
- Modify the connectivity between new resource atoms with the Chip Planner and Resource Property Editor

Related Information

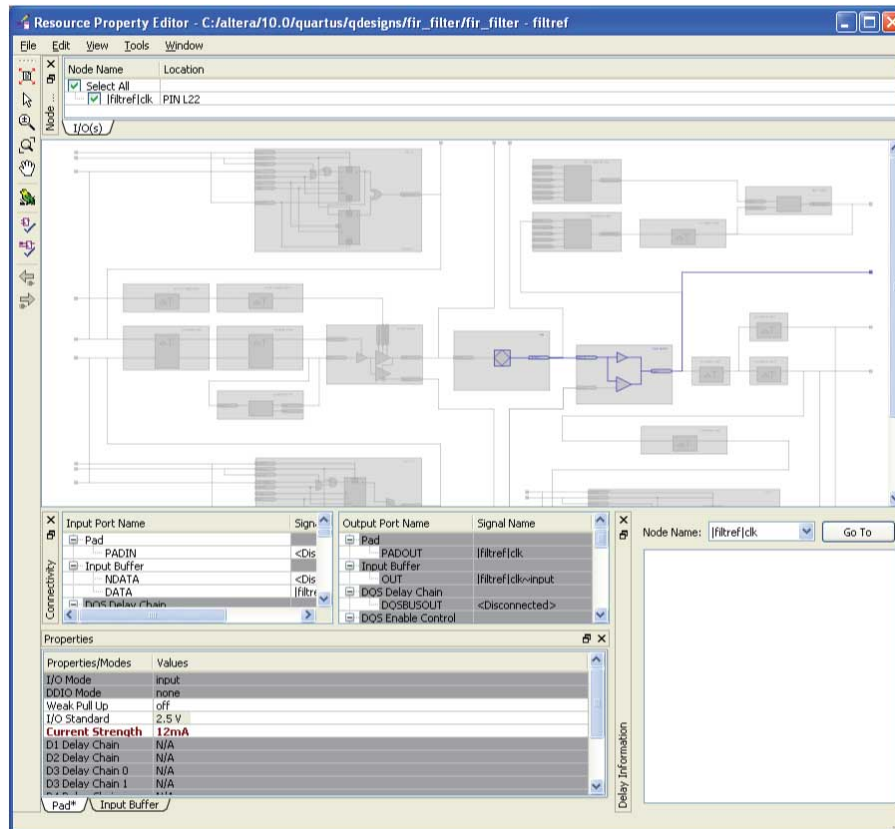
[Modify the PLL Properties With the Chip Planner](#) on page 160

7.8.1. Adjust the Drive Strength of an I/O with the Chip Planner

To adjust the drive strength of an I/O, follow these steps to incorporate the ECO changes into the netlist of the design.

1. In the **Editing Mode** list at the top of the Chip Planner, select the ECO editing mode.
2. Locate the I/O in the **Resource Property Editor**.
3. In the **Resource Property Editor**, point to the **Current Strength** option in the **Properties** pane and double-click the value to enable the drop-down list.
4. Change the value for the **Current Strength** option.
5. Right-click the ECO change in the Change Manager and click **Check & Save All Netlist Changes** to apply the ECO change.

Figure 63. I/O in the Resource Property Editor



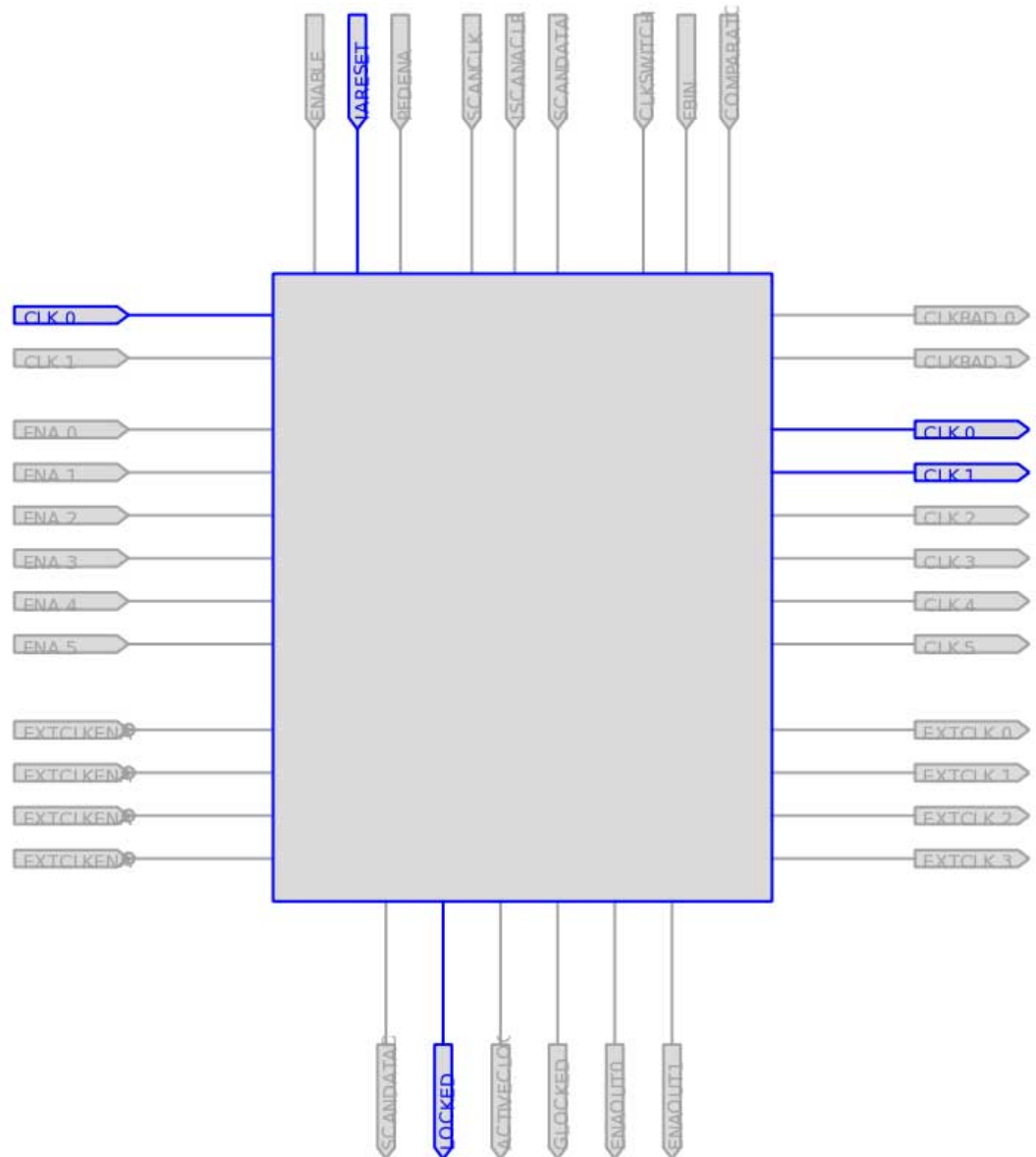
Note: You can change the pin locations of input or output ports with the ECO flow. You can drag and move the signal from an existing pin location to a new location while in the Post Compilation Editing (ECO) task in the Chip Planner. You can then click **Check & Save All Netlist Changes** to compile the ECO.

7.8.2. Modify the PLL Properties With the Chip Planner

You use PLLs to modify and generate clock signals to meet design requirements. Additionally, you can use PLLs to distribute clock signals to different devices in a design, reducing clock skew between devices, improving I/O timing, and generating internal clock signals.

The Resource Property Editor allows you to view and modify PLL properties to meet your design requirements.

Figure 64. PLL View in the Resource Property Editor of a Stratix Device



7.8.3. PLL Properties

The Resource Property Editor allows you to modify PLL options, such as phase shift, output clock frequency, and duty cycle.

You can also change the following PLL properties with the Resource Property Editor:

- Input frequency
- M V_{CO} tap
- M initial
- M value



- N value
- M counter delay
- N counter delay
- M2 value
- N2 value
- SS counter
- Charge pump current
- Loop filter resistance
- Loop filter capacitance
- Counter delay
- Counter high
- Counter low
- Counter mode
- Counter initial
- V_{CO} tap

You can also view post-compilation PLL properties in the Compilation Report. To do so, in the Compilation Report, select **Fitter** and then select **Resource Section**.

7.8.3.1. Adjusting the Duty Cycle

Use the equation to adjust the duty cycle of individual output clocks.

$$\text{High \%} = \frac{\text{Counter High}}{(\text{Counter High} + \text{Counter Low})}$$

7.8.3.2. Adjusting the Phase Shift

Use the equation to adjust the phase shift of an output clock of a PLL.

$$\text{Phase Shift} = (\text{Period } V_{CO} \times 0.125 \times \text{Tap } V_{CO}) + (\text{Initial } V_{CO} \times \text{Period } V_{CO})$$

For normal mode, Tap V_{CO}, Initial V_{CO}, and Period V_{CO} are governed by the following settings:

$$\text{Tap } V_{CO} = \text{Counter Delay} - M \text{ Tap } V_{CO}$$

$$\text{Initial } V_{CO} = \text{Counter Delay} - M \text{ Initial}$$

$$\text{Period } V_{CO} = \text{In Clock Period} \times N \div M$$

For external feedback mode, Tap V_{CO}, Initial V_{CO}, and Period V_{CO} are governed by the following settings:

$$\text{Tap } V_{CO} = \text{Counter Delay} - M \text{ Tap } V_{CO}$$

$$\text{Initial } V_{CO} = \text{Counter Delay} - M \text{ Initial}$$

$$\text{Period } V_{CO} = \underline{\text{In Clock Period} \times N}$$



(M+ Counter High+Counter Low)

Related Information

Stratix Device Handbook

7.8.3.3. Adjusting the Output Clock Frequency

Use the equation to adjust the PLL output clock in normal mode.

$$\text{Output Clock Frequency} = \text{Input Frequency} \cdot \frac{\text{M Value}}{\text{N Value} + \text{Counter High} + \text{Counter Low}}$$

Use the equation to adjust the PLL output clock in external feedback mode.

$$\text{OUTCLK} = \frac{\text{M Value} + \text{External Feedback Counter High} + \text{External Feedback Counter Low}}{\text{N Value} + \text{Counter High} + \text{Counter Low}}$$

7.8.3.4. Adjusting the Spread Spectrum

Use the equation to adjust the spread spectrum for your PLL.

$$\% \text{ Spread} = \frac{M_2 N_1}{M_1 N_2}$$

7.8.4. Modify the Connectivity between Resource Atoms

The Chip Planner and Resource Property Editor allow you to create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist. These features are useful for small changes when you are debugging a design, such as manually inserting pipeline registers into a combinational path that fails timing, or routing a signal to a spare I/O pin for analysis.

Use the following procedure to create a new register in a Cyclone V device and route register output to a spare I/O pin. This example illustrates how to create a new resource atom and modify the connections between resource atoms.

To create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist, follow these steps:

1. Create a new register in the Chip Planner.
2. Locate the atom in the Resource Property Editor.
3. To assign a clock signal to the register, right-click the clock input port for the register, point to **Edit connection**, and click **Other**. Use the Node Finder to assign a clock signal from your design.
4. To tie the SLOAD input port to V_{CC} , right-click the clock input port for the register, point to **Edit connection**, and click **VCC**.
5. Assign a data signal from your design to the SDATA port.
6. In the Connectivity window, under the output port names, copy the port name of the register.
7. In the Chip Planner, locate a free I/O resource and create an output buffer.
8. Locate the new I/O atom in the Resource Property Editor.



9. On the input port to the output buffer, right-click, point to **Edit connection**, and click **Other**.
10. In the **Edit Connection** dialog box, type the output port name of the register you have created.
11. Run the ECO Fitter to apply the changes by clicking **Check and Save Netlist Changes**.

Note: A successful ECO connection is subject to the available routing resources. You can view the relative routing utilization by selecting **Routing Utilization** as the Background Color Map in the **Layers Settings** dialog box of the Chip Planner. Also, you can view individual routing channel utilization from local, row, and column interconnects with the tooltips created when you position your mouse pointer over the appropriate resource. Refer to the device data sheet for more information about the architecture of the routing interconnects of your device.

7.9. Post ECO Steps

After you make an ECO change with the Chip Planner, you must perform static timing analysis of your design with the Timing Analyzer to ensure that your changes did not adversely affect the timing performance of your design.

For example, when you turn on one of the delay chain settings for a specific pin, you change the I/O timing. Therefore, to ensure that the design still meets all timing requirements, you should perform static timing analysis.

Related Information

[Intel Quartus Prime Timing Analyzer](#)

For more information about performing a static timing analysis of your design

7.10. Engineering Change Orders with the Chip Planner Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.
2018.05.07	18.0.0	Added statement indicating no Chip Planner ECO support for Intel Arria 10 devices.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
June 2014	14.0.0	<ul style="list-style-type: none"> • Updated formatting. • Removed references to Stratix, Stratix II, Stratix III, Arria GX, Arria II GX, Cyclone, Cyclone II, Cyclone III, and MAX II devices. • Added MAX V, Cyclone V, Arria V I/O elements
June 2012	12.0.0	Removed survey link.
November 2011	10.1.1	Template update.
December 2010	10.1.0	<ul style="list-style-type: none"> • Updated chapter to new template • Removed "The Chip Planner FloorPlan Views" section • Combined "Creating Atoms", "Deleting Atoms", and "Moving Atoms" sections, and linked to Help. • Added Stratix V I/O elements in "FPGA I/O Elements".
<i>continued...</i>		

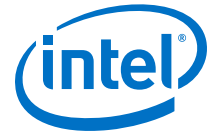


Document Version	Intel Quartus Prime Version	Changes
July 2010	10.0.0	<ul style="list-style-type: none"> Added information to page 17-1. Added information to "Engineering Change Orders" on page 17-2. Changed heading from "Performance" to "Performance Preservation" on page 7-2. Updated information in "Performance Preservation" on page 17-2. Changed heading from "Documentation" to "Change Modification Record" on page 17-3. Changed heading from "Resource Property Editor" to "Performing ECOs in the Resource Property Editor" on page 17-15. Removed "Using Incremental Compilation in the ECO Flow" section. Preservation support for ECOs with the incremental compilation flow has been removed in the Quartus II software version 10.0. Removed "Referenced Documents" section.
November 2009	9.1.0	<ul style="list-style-type: none"> Updated device support list Made minor editorial updates
March 2009	9.0.0	<ul style="list-style-type: none"> Updated Figure 17-17. Made minor editorial updates. Chapter 15 was previously Chapter 13 in the 8.1.0 release.
November 2008	8.1.0	<ul style="list-style-type: none"> Corrected preservation attributes for ECOs in the section "Using Incremental Compilation in the ECO Flow" on page 15-32. Minor editorial updates. Changed to 8½" x 11" page size.
May 2008	8.0.0	<ul style="list-style-type: none"> Updated device support list Modified description for ECO support for block RAMs and DSP blocks Corrected Stratix PLL ECO example Added an application example to show modifying the connectivity between resource atoms

Related Information

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.



A. Intel Quartus Prime Standard Edition User Guides

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Standard Edition FPGA design flow.

Related Information

- [Intel Quartus Prime Standard Edition User Guide: Getting Started](#)
Introduces the basic features, files, and design flow of the Intel Quartus Prime Standard Edition software, including managing Intel Quartus Prime Standard Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- [Intel Quartus Prime Standard Edition User Guide: Platform Designer](#)
Describes creating and optimizing systems using Platform Designer (Standard), a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer (Standard) automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- [Intel Quartus Prime Standard Edition User Guide: Design Recommendations](#)
Describes best design practices for designing FPGAs with the Intel Quartus Prime Standard Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Standard Edition synthesis optimally implements your design in hardware.
- [Intel Quartus Prime Standard Edition User Guide: Design Compilation](#)
Describes set up, running, and optimization for all stages of the Intel Quartus Prime Standard Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- [Intel Quartus Prime Standard Edition User Guide: Design Optimization](#)
Describes Intel Quartus Prime Standard 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, and optimization of device resource usage.
- [Intel Quartus Prime Standard Edition User Guide: Programmer](#)
Describes operation of the Intel Quartus Prime Standard 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 Standard 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 Standard 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 Standard 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 Standard Edition User Guide: Debug Tools](#)
Describes a portfolio of Intel Quartus Prime Standard 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 Standard Edition User Guide: Timing Analyzer](#)
Explains basic static timing analysis principals and use of the Intel Quartus Prime Standard 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 Standard Edition User Guide: Power Analysis and Optimization](#)
Describes the Intel Quartus Prime Standard 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 Standard 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 Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- [Intel Quartus Prime Standard 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 Standard Edition User Guide: Scripting](#)
Describes use of Tcl and command line scripts to control the Intel Quartus Prime Standard Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.