## 1. Design Optimization Overview

1.1. Device Considerations

1.1.1. Device Migration Considerations

1.2. Required Settings for Initial Compilation

1.2.1. Guidelines for I/O Assignments

1.2.2. Guidelines for Time Constraints

1.3. Trade-Offs and Limitations

1.3.1. Reducing Area

1.3.2. Reducing Critical Path Delay

1.3.3. Reducing Power Consumption

1.3.4. Reducing Runtime

1.4. Intel Quartus Prime Software Tools for Design Optimization

1.4.1. Design Visualization Tools

1.4.2. Advisors

1.4.3. Design Exploration

1.5. Design Space Explorer II

1.5.1. How DSE II Works

1.5.2. Performing a Design Exploration with the DSE II Utility

1.6. Design Optimization Overview Revision History

## 2. Optimizing the Design Netlist

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

2.2. Intel Quartus Prime Design Flow with the Netlist Viewers

2.3. RTL Viewer Overview

2.3.1. Maximizing Readability in RTL Viewer

2.3.2. Running the RTL Viewer

2.4. Technology Map Viewer Overview

2.5. Netlist Viewer User Interface

2.5.1. Netlist Navigator Pane

2.5.2. Properties Pane

2.5.3. Netlist Viewers Find Pane

2.6. Schematic View

2.6.1. Display Schematics in Multiple Tabbed View

2.6.2. Schematic Symbols

2.6.3. Select Items in the Schematic View

2.6.4. Shortcut Menu Commands in the Schematic View

2.6.5. Filtering in the Schematic View

2.6.6. View Contents of Nodes in the Schematic View

2.6.7. Moving Nodes in the Schematic View

2.6.8. View LUT Representations in the Technology Map Viewer

2.6.9. Zoom Controls

2.6.10. Navigating with the Bird’s Eye View

2.6.11. Partition the Schematic into Pages

2.6.12. Follow Nets Across Schematic Pages

2.6.13. Maintaining Selection in the Resource Property Viewer

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

2.8. Cross-Probing to the Netlist Viewers from Other Intel Quartus Prime Windows
5.6.8. Intel Stratix 10 Timing Closure Recommendations ........................................ 100
5.7. Periphery to Core Register Placement and Routing Optimization .......................... 102
\hspace{1cm} 5.7.1. Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box .......................................................... 103
\hspace{1cm} 5.7.2. Setting Periphery to Core Optimizations in the Assignment Editor .......... 104
\hspace{1cm} 5.7.3. Viewing Periphery to Core Optimizations in the Fitter Report ................. 104
5.8. Scripting Support ............................................................................................... 105
\hspace{1cm} 5.8.1. Initial Compilation Settings .................................................................. 106
\hspace{1cm} 5.8.2. I/O Timing Optimization Techniques .................................................. 106
\hspace{1cm} 5.8.3. Register-to-Register Timing Optimization Techniques .......................... 107
5.9. Timing Closure and Optimization Revision History ............................................. 107

6. Analyzing and Optimizing the Design Floorplan .................................................. 111
\hspace{1cm} 6.1. Design Floorplan Analysis in the Chip Planner .......................................... 111
\hspace{2cm} 6.1.1. Starting the Chip Planner ........................................................................ 112
\hspace{2cm} 6.1.2. Chip Planner GUI Components ............................................................. 112
\hspace{2cm} 6.1.3. Viewing Design Elements in the Chip Planner ........................................ 113
\hspace{2cm} 6.1.4. Exploring Paths in the Chip Planner ...................................................... 121
\hspace{2cm} 6.1.5. Viewing Assignments in the Chip Planner ............................................. 124
\hspace{2cm} 6.1.6. Viewing High-Speed and Low-Power Tiles in the Chip Planner ............... 125
\hspace{1cm} 6.2. Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner ............................................................ 125
\hspace{3cm} 6.2.1. Viewing Design Connectivity and Hierarchy ........................................... 126
\hspace{1cm} 6.3. Using Logic Lock Regions in the Chip Planner ........................................ 128
\hspace{2cm} 6.3.1. Viewing Connections Between Logic Lock Regions in the Chip Planner .... 128
\hspace{2cm} 6.3.3. Logic Lock Regions ................................................................................ 128
\hspace{2cm} 6.3.4. Attributes of a Logic Lock Region ........................................................... 129
\hspace{2cm} 6.3.5. Migrating Assignments between Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition ..................................................... 129
\hspace{2cm} 6.3.6. Creating Logic Lock Regions .................................................................. 129
\hspace{2cm} 6.3.7. Customizing the Shape of Logic Lock Regions ........................................ 132
\hspace{2cm} 6.3.8. Placing Device Resources into Logic Lock Regions ............................... 134
\hspace{2cm} 6.3.9. Hierarchical Regions ............................................................................ 138
\hspace{2cm} 6.3.10. Additional Intel Quartus Prime Logic Lock Design Features .................. 138
\hspace{2cm} 6.3.11. Logic Lock Regions Window .................................................................. 138
\hspace{2cm} 6.3.12. Snapping to a Region .......................................................................... 139
\hspace{1cm} 6.4. Using User-Defined Clock Regions in the Chip Planner ................................ 140
\hspace{2cm} 6.4.1. Creating Clock Assignments with the Chip Planner ................................ 141
\hspace{2cm} 6.4.2. Resizing a Clock Assignment .................................................................. 142
\hspace{2cm} 6.4.3. Moving a Clock Assignment ................................................................. 142
\hspace{2cm} 6.4.4. Deleting a Clock Region Assignment ..................................................... 142
\hspace{2cm} 6.4.5. Assigning a Clock Signal to a Clock Region ........................................... 143
\hspace{2cm} 6.4.6. Clock Assignment Properties .................................................................. 143
\hspace{1cm} 6.5. Scripting Support ...................................................................................... 144
\hspace{2cm} 6.5.1. Creating Logic Lock Assignments with Tcl commands ........................... 144
\hspace{2cm} 6.5.2. Assigning Virtual Pins with a Tcl command ............................................ 145
\hspace{2cm} 6.5.3. Logic Lock Region Assignment Examples ............................................. 145
\hspace{1cm} 6.6. Analyzing and Optimizing the Design Floorplan Revision History ............... 146

7. Implementing Engineering Change Orders .......................................................... 149
\hspace{1cm} 7.1. Engineering Change Order Flow .................................................................. 149
\hspace{1cm} 7.2. ECO Tcl Script Example ............................................................................. 150
7.3. Viewing ECO Compilation Reports ................................................................. 151
7.4. ECO Commands ............................................................................................. 151
   7.4.1. ECO Command Quick Reference ........................................................... 152
   7.4.2. make_connection .................................................................................. 152
   7.4.3. remove_connection .............................................................................. 153
   7.4.4. modify_lutmask .................................................................................... 154
   7.4.5. adjust_pll_refclk ............................................................................... 154
   7.4.6. modify_io_slew_rate .......................................................................... 155
   7.4.7. modify_io_current_strength ............................................................... 155
   7.4.8. modify_io_delay_chain ...................................................................... 156
   7.4.9. create_new_node ............................................................................... 156
   7.4.10. place_node ...................................................................................... 157
   7.4.11. create_wirelut .................................................................................... 157
7.5. ECO Command Limitations ............................................................................. 158
7.6. Interactive ECO Fitting .................................................................................. 159
   7.6.1. eco_load_design and eco_commit_design Commands ......................... 159
7.7. Implementing Engineering Change Orders Revision History ....................... 160


A. Intel Quartus Prime Pro Edition User Guides .................................................. 162
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
- Compilation Flows
  In Intel Quartus Prime Pro Edition User Guide: Compiler
- 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 Pro 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.

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 60
- Using the Intel Quartus Prime Timing Analyzer
  In Intel Quartus Prime Pro Edition User Guide: Timing Analyzer
- Intel Quartus Prime Timing Analyzer Cookbook
- Advanced Settings (Fitter)
  In Intel Quartus Prime Help
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

<table>
<thead>
<tr>
<th>Trade-off</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource usage and critical path timing.</td>
<td>Certain techniques (such as logic duplication) can improve timing performance at the cost of increased area.</td>
</tr>
<tr>
<td>Power requirements can result in area and timing trade-offs.</td>
<td>For example, reducing the number of available high-speed tiles, or attempting to shorten high-power nets at the expense of critical path nets.</td>
</tr>
<tr>
<td>System cost and time-to-market considerations can affect the choice of device.</td>
<td>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.</td>
</tr>
</tbody>
</table>

Finally, constrains 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. 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 48
- Netlist Optimizations and Physical Synthesis on page 42

1.3.2. 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.

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 60

1.3.3. 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 Pro Edition User Guide: Power Analysis and Optimization

1.3.4. 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%.

Related Information
Reducing Compilation Time
In Intel Quartus Prime Pro 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

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTL Viewer</td>
<td>Provides a schematic representation of the design before synthesis and place-and-route.</td>
</tr>
<tr>
<td>Technology Map Viewer</td>
<td>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.</td>
</tr>
<tr>
<td>Design Partition Planner</td>
<td>Displays designs at partition and entity levels, and can display connectivity between entities</td>
</tr>
<tr>
<td>Design Partition Planner and Chip Planner</td>
<td>Allow you to partition and layout the design at a higher level.</td>
</tr>
<tr>
<td>Chip Planner</td>
<td>Allow you to make floorplan assignments, perform power analysis, and visualize critical paths and routing congestion.</td>
</tr>
</tbody>
</table>

**Related Information**

- Design Floorplan Analysis in the Chip Planner on page 111
- Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner on page 125
- Optimizing the Design Netlist on page 16
- RTL Viewer Overview on page 18

### 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 stages presented by complexity.

The advisors are:

- Timing Optimization Advisor
- Power Optimization Advisor
- Compilation Time Advisor

**Related Information**

- Compilation Time Advisor
  In Intel Quartus Prime Pro Edition User Guide: Compiler
- Advisors in the Intel Quartus Prime Software
  In Intel Quartus Prime Help
- Timing Optimization Advisor on page 82
- Power Optimization Advisor
  In Intel Quartus Prime Pro 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 11

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**

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.
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**
- Fitter Seed on page 98
- Design Space Explorer II for Power-Driven Optimization
  In Intel Quartus Prime Pro 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:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2018.05.07       | 18.0.0                      | • General topic reorganization.  
                   |                             | • Added how DSE II works, and the main steps to follow when performing a design exploration. |
| 2017.11.06       | 17.1.0                      | • Added mention to the Design Partition Planner in Design Analysis topic. |
| 2016.10.31       | 16.1.0                      | • Implemented Intel rebranding. |
| 2016.05.03       | 16.0.0                      | Removed statements about serial equivalence when using multiple processors. |
| 2015.11.02       | 15.1.0                      | Changed instances of Quartus II to Quartus Prime. |
| 2014.12.15       | 14.1.0                      | • 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 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 17
- RTL Viewer Overview on page 18
- Technology Map Viewer Overview on page 19
- Filtering in the Schematic View on page 31
- Viewing a Timing Path on page 39

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

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

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

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

You can use the Technology Map Viewer to look at the results at the end of Analysis and Synthesis. If you have compiled your design through the Fitter stage, you can view your post-mapping netlist in the Technology Map Viewer (Post-Mapping) and your post-fitting netlist in the Technology Map Viewer. If you perform only Analysis and Synthesis, both the Netlist Viewers display the same post-mapping netlist.
In addition, you can use the RTL Viewer or Technology Map Viewer to locate the source of a particular signal, which can help you debug your design. Use the navigation techniques described in this chapter to search easily through your design. You can trace back from a point of interest to find the source of the signal and ensure the connections are as expected.

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

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

Related Information
- Intel Quartus Prime Design Flow with the Netlist Viewers on page 17
- RTL Viewer Overview on page 18
- Technology Map Viewer Overview on page 19

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

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

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

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

2.3. RTL Viewer Overview

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

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

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

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

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

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

To open the RTL Viewer, click **Tools ➤ Netlist Viewers ➤ RTL Viewer**.
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 VCC and GND.
- Groups pins, nets, wires, module ports, and certain logic into buses where appropriate.
- Groups constant bus connections.
- Displays values in hexadecimal format.
- Converts NOT gates into bubble inversion symbols in the schematic.
- Merges chains of equivalent combinational gates into a single gate; for example, a 2-input AND gate feeding a 2-input AND gate is converted to a single 3-input AND gate.

2.3.2. Running the RTL Viewer

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

1. Analyze the design to generate an RTL netlist by clicking Processing ➤ Start ➤ Start Analysis & Elaboration. You can also perform a full compilation on any process that includes the initial Analysis and Elaboration stage of the Intel Quartus Prime compilation flow.

2. Open the RTL Viewer by clicking Tools ➤ Netlist Viewers ➤ RTL Viewer.

2.4. Technology Map Viewer Overview

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

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

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

You can view Intel Quartus Prime technology-mapped results after synthesis, fitting, or timing analysis. To run the Technology Map Viewer for an Intel Quartus Prime project, on the Processing menu, point to Start and click Start Analysis & Synthesis to synthesize and map the design to the target technology. At this stage,
the Technology Map Viewer shows the same post-mapping netlist as the Technology
Map Viewer (Post-Mapping). You can also perform a full compilation, or any process
that includes the synthesis stage in the compilation flow.

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

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

**Related Information**
- Viewing a Timing Path on page 39
- View Contents of Nodes in the Schematic View on page 31
- Netlist Viewer User Interface on page 20

### 2.5. Netlist Viewer User Interface

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

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

- The **Netlist Navigator** pane—displays a representation of the project hierarchy.
- The **Find** pane—allows you to find and locate specific design elements in the
  schematic view.
- The **Properties** pane displays the properties of the selected block when you select
  **Properties** from the shortcut menu.
- The schematic view—displays a graphical representation of the internal structure
  of the design.
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), and one Technology Map Viewer (Post-Mapping) window open at the same time, although each window can show multiple pages, each with multiple tabs. For example, you cannot have two RTL Viewer windows open at the same time.
2.5.1. Netlist Navigator Pane

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

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

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

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

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instances</td>
<td>Modules or instances in the design that can be expanded to lower hierarchy levels.</td>
</tr>
<tr>
<td>Primitives</td>
<td>Low-level nodes that cannot be expanded to any lower hierarchy level. These primitives include:</td>
</tr>
<tr>
<td></td>
<td>• Registers and gates that you can view in the RTL Viewer when using Intel Quartus Prime Pro Edition synthesis.</td>
</tr>
<tr>
<td></td>
<td>• Logic cell atoms in the Technology Map Viewer or in the RTL Viewer when using a VQM or EDIF from third-party synthesis software.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Ports</td>
<td>The I/O ports in the current level of hierarchy.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• When a pin represents a bus or an array of pins, expand the pin entry in the list view to see individual pin names.</td>
</tr>
</tbody>
</table>

2.5.2. 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, VCC or GND). The following table lists the possible values of a port:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>The port is not connected and has VCC value (tied to VCC)</td>
</tr>
<tr>
<td>GND</td>
<td>The port is not connected and has GND value (tied to GND)</td>
</tr>
<tr>
<td>--</td>
<td>The port is connected and has value (other than VCC or GND)</td>
</tr>
<tr>
<td>Unconnected</td>
<td>The port is not connected and has no value (hanging)</td>
</tr>
</tbody>
</table>

If the selected node is an atom primitive, the **Properties** pane displays a schematic of the internal logic.
2.5.3. Netlist Viewers Find Pane

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

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

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

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

2.6. Schematic View

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

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

2.6.1. Display Schematics in Multiple Tabbed View

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

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

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

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

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

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

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

#### Table 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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I/O Ports</strong></td>
<td>An input, output, or bidirectional port in the current level of hierarchy. A device input, output, or bidirectional pin when viewing the top-level hierarchy. The symbol can also represent a bus. Only one wire is shown connected to the bidirectional symbol, representing the input and output paths. Input symbols appear on the left-most side of the schematic. Output and bidirectional symbols appear on the right-most side of the schematic.</td>
</tr>
<tr>
<td>![CLK_SEL(1:0)]</td>
<td><img src="image" alt="RESET_N" /></td>
</tr>
<tr>
<td><strong>I/O Connectors</strong></td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="MEM_OE_N" /></td>
<td><img src="image" alt="MEM_OE_N" /></td>
</tr>
<tr>
<td><img src="image" alt="MEM_OE_N" /></td>
<td><img src="image" alt="MEM_OE_N" /></td>
</tr>
<tr>
<td><strong>OR, AND, XOR Gates</strong></td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="always1" /></td>
<td><img src="image" alt="always1" /></td>
</tr>
<tr>
<td><img src="image" alt="always0" /></td>
<td><img src="image" alt="always0" /></td>
</tr>
<tr>
<td><img src="image" alt="C" /></td>
<td><img src="image" alt="C" /></td>
</tr>
<tr>
<td><strong>MULTIPLEXER</strong></td>
<td>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.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><img src="image" alt="Mux5 Symbol" /></td>
<td><strong>Mux5</strong>&lt;br&gt;Sel[2:0]&lt;br&gt;Data[7:0]&lt;br&gt;Out&lt;br&gt;<strong>Description</strong>&lt;br&gt;A buffer primitive. The figure shows the tri-state buffer, with an inverted output enable port. Other buffers without an enable port include LCELL, SOFT, and GLOBAL. The NOT gate and EXP expander buffers use this symbol without an enable port and with an inverted output port.</td>
</tr>
<tr>
<td><img src="image" alt="Buffer Symbol" /></td>
<td><strong>BUFFER</strong>&lt;br&gt;OE&lt;br&gt;DataIn → Out0&lt;br&gt;<strong>Description</strong>&lt;br&gt;A buffer primitive. The figure shows the tri-state buffer, with an inverted output enable port. Other buffers without an enable port include LCELL, SOFT, and GLOBAL. The NOT gate and EXP expander buffers use this symbol without an enable port and with an inverted output port.</td>
</tr>
</tbody>
</table>
| ![Latch Symbol](image) | **LATCH**<br>Pre<br>D<br>Q<br>Ena<br>Clr<br>**Description**<br>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:  
  - DFFEA (data flipflop with enable and asynchronous load) primitive with additional ALOAD asynchronous load and ADATA data signals  
  - DFFEAS (data flipflop with enable and synchronous and asynchronous load), which has ASDATA as the secondary data port |
| ![Atom Primitive](image) | **Atom Primitive**<br>DataA<br>DataB<br>CombOut<br>DataC<br>**Description**<br>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. |
| ![Other Primitive](image) | **Other Primitive**<br>Cpu_D[10]<br>PadIn<br>PadOut<br>Padio<br>Bidir<br>**Description**<br>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. |
| ![Instance](image) | **Instance**<br>Sspeed_ch:speed<br>Get_ticket<br>Accel_in<br>Clk<br>Reset<br>**Description**<br>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. |
| ![Encrypted Instance](image) | **Encrypted Instance**<br>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. |

*continued...*
### 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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[3:0]</td>
<td>An adder operator: ( \text{OUT} = A + B )</td>
</tr>
</tbody>
</table>

---

**Table 6.** Operator Symbols in the RTL Viewer Schematic View

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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>A constant signal value that is highlighted in gray and displayed in hexadecimal format by default throughout the schematic.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Mult0</td>
<td>A multiplier operator: ( OUT = A \times B )</td>
</tr>
<tr>
<td>Div0</td>
<td>A divider operator: ( OUT = A / B )</td>
</tr>
<tr>
<td>Equal3</td>
<td>Equals</td>
</tr>
<tr>
<td>ShiftLeft0</td>
<td>A left shift operator: ( OUT = (A &lt;&lt; COUNT) )</td>
</tr>
<tr>
<td>ShiftRight0</td>
<td>A right shift operator: ( OUT = (A &gt;&gt; COUNT) )</td>
</tr>
<tr>
<td>Mod0</td>
<td>A modulo operator: ( OUT = (A % B) )</td>
</tr>
<tr>
<td>LessThan0</td>
<td>A less than comparator: ( OUT = (A &lt; B : A &gt; B) )</td>
</tr>
<tr>
<td>Mux5</td>
<td>A multiplexer: ( OUT = DATA[SEL] ) The data range size is ( 2^{sel\ range\ size} )</td>
</tr>
</tbody>
</table>

*continued...*
### Related Information

- [Partition the Schematic into Pages](#) on page 35
- [Follow Nets Across Schematic Pages](#) on page 36

#### 2.6.3. Select Items in the Schematic View

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

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

Items selected in the schematic view are automatically selected in the **Netlist Navigator** pane. The folder then expands automatically if it is required to show the selected entry; however, the folder does not collapse automatically when you deselect 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 23

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

When you right-click a selected instance or primitive in the schematic view, the Netlist Viewer displays a shortcut menu.
If the selected item is a node, you see the following options:

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

### 2.6.5. Filtering in the Schematic View

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

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

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

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

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

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

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

![Diagram](image1.png)

*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.

![Display Nets Across Hierarchies](image)

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.

![Show Connectivity Details](image)

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

### 2.6.7. Moving Nodes in the Schematic View

Rearrange items in the schematic view by dragging to destination.

To move a node from one area of the netlist to another, select the node and hold down the Shift key. Legal placements appear as shaded areas within the hierarchy. Click to drop the selected node.
2.6.8. View LUT Representations in the Technology Map Viewer

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

You can view the LUT representations in the following three tabs in the Properties dialog box:
- The Schematic tab—the equivalent gate representations of the LUT.
- The Truth Table tab—the truth table representations.

Related Information
Properties Pane on page 23

2.6.9. Zoom Controls

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

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

You can use the Zoom Tool on the Netlist Viewer toolbar to control magnification in the schematic view. When you select the Zoom Tool in the toolbar, clicking in the schematic zooms in and centers the view on the location you clicked. Right-click in the schematic to zoom out and center the view on the location you clicked. When you
select the Zoom Tool, you can also zoom into a certain portion of the schematic by selecting a rectangular box area with your mouse cursor. The schematic is enlarged to show the selected area.

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

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

**Related Information**
Filtering in the Schematic View on page 31

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

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

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

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

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

### 2.6.11. Partition the Schematic into Pages

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

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

**Related Information**
Netlist Viewer User Interface on page 20
2.6.12. Follow Nets Across Schematic Pages

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

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

**Related Information**

Schematic Symbols on page 26

2.6.13. Maintaining Selection in the Resource Property Viewer

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

From 19.1 release, when you click on a port (in the schematic) or the signal name (in the Connectivity pane), the path is selected. Each node keeps only a dependency path. This dependency path is cleared when you recompile, close the project, select another port, run a new locate, or left-click on a non-port space.
Note:  
- Every double-click on a node in the Chip Planner to open the Resource Property Viewer is considered as a new locate.
- For the path-tracing purpose, using the **Go To Destination Node** or **Go To Source Node** option adds more nodes to the Node Selection table, and does not clear the path-dependency selection (as opposed to every new locate, for example, double-clicking on nodes in the Chip Planner).
- The dependency path selection or highlight is not applicable to Node Properties in the Chip Planner.

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

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

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

You can select one or more hierarchy boxes, nodes, state nodes, or state transition arcs that interest you in the Netlist Viewer and locate the corresponding items in another applicable Intel Quartus Prime software window. You can then view and make changes or assignments in the appropriate editor or floorplan.
To locate an item from the Netlist Viewer in another window, right-click the items of interest in the schematic or state diagram, point to Locate, and click the appropriate command. The following commands are available:

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

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

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

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

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

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

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

To locate elements in the Netlist Viewer from another Intel Quartus Prime window, select the node or nodes in the appropriate window; for example, select an entity in the Entity list on the Hierarchy tab in the Project Navigator, or select nodes in the Timing Closure Floorplan, or select node names in the From or To column in the Assignment Editor. Next, right-click the selected object, point to Locate, and click
Locate in RTL Viewer or Locate in Technology Map Viewer. After you click this command, the Netlist Viewer opens, or is brought to the foreground if the Netlist Viewer is open.

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

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

2.9. Viewing a Timing Path

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

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

1. To open the report from the Compilation Report Table of Contents, click Timing Analyzer GUI ➤ Report Timing, and double-click the timing corner.

2. To open the report from the Timing Analyzer, open the Report Timing folder in the Report pane, and double-click the timing corner.

3. In the Summary of Paths tab, right-click a row in the table and select Locate Path ➤ Locate in Technology Map Viewer. In the Technology Map Viewer, the schematic page displays the nodes along the timing path with a summary of the total delay.

Related Information

Report Timing (Dialog Box)
In Intel Quartus Prime Pro Edition User Guide: Timing Analyzer

2.10. Optimizing the Design Netlist Revision History

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.07.01</td>
<td>19.1</td>
<td>Added Maintaining Selection in the Resource Property Viewer topic explaining how the iterm-oterm dependency is maintained in the schematic view.</td>
</tr>
</tbody>
</table>
| 2018.09.24       | 18.1.0                      | • Added link to Viewing a Timing Path.  
• Removed reference to unsupported CARRY buffer from "Schematic Symbols" topic. |
| 2016.10.31       | 16.1.0                      | • Implemented Intel rebranding. |
| 2016.05.03       | 16.0.0                      | Removed Schematic Viewer topic. |

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2015.11.02       | 15.1.0                      | Added Schematic Viewer topic for viewing stage snapshots. Added information for the following new features and feature updates:  
  - 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) |
| 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                      |  
  - Updated screenshots  
  - Updated chapter for the Intel Quartus Prime software version 10.0, including major user interface changes |
| November 2009    | 9.1.0                       |  
  - Updated devices  
  - Minor text edits |
| March 2009       | 9.0.0                       |  
  - 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                       |  
  - 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  
  - 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. Netlist Optimizations and Physical Synthesis

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

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

**Table 7. Synthesis Netlist Optimization and Physical Synthesis Options**

<table>
<thead>
<tr>
<th>Options</th>
<th>Location/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable synthesis netlist</td>
<td>Enable synthesis optimization options (for example, Synthesis Effort) in the</td>
</tr>
<tr>
<td>optimization settings</td>
<td>Advanced Synthesis Settings dialog box. Click Assignments ➤ Settings ➤ Compiler</td>
</tr>
<tr>
<td></td>
<td>Settings ➤ Advanced Settings (Synthesis) to access these options.</td>
</tr>
<tr>
<td>Enable physical synthesis</td>
<td>Enable physical synthesis options (for example, Advanced Physical Synthesis) in the</td>
</tr>
<tr>
<td>options</td>
<td>Advanced Fitter Settings dialog box. Click Assignments ➤ Settings ➤ Compiler Settings</td>
</tr>
<tr>
<td></td>
<td>➤ Advanced Settings (Fitter) to access these settings.</td>
</tr>
</tbody>
</table>

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

3.1. Physical Synthesis Optimizations

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

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

**Related Information**

Compiler Settings Page (Settings Dialog Box)
In Intel Quartus Prime Help
3.1.1. Enabling Physical Synthesis Optimization

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

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

3.1.2. Physical Synthesis Options

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

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

Table 8. Physical Synthesis Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Physical Synthesis</td>
<td>Uses the physical synthesis engine to perform combinational and sequential optimization during fitting to improve circuit performance.</td>
</tr>
<tr>
<td>Netlist Optimizations</td>
<td>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.</td>
</tr>
<tr>
<td>Allow Register Duplication</td>
<td>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 &amp; Synthesis and the Fitter.</td>
</tr>
<tr>
<td>Allow Register Merging</td>
<td>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 &amp; Synthesis and the Fitter.</td>
</tr>
</tbody>
</table>

3.2. Applying Netlist Optimizations

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

You may have to experiment with available options to see which combination of settings works best for a particular design. Refer to the messages in the compilation report to see the magnitude of improvement with each option, and to help you decide whether you should turn on a given option or specific effort level.
Turning on more netlist optimization options can result in more changes to the node names in the design; bear this in mind if you are using a verification flow, such as the Signal Tap Logic Analyzer or formal verification that requires fixed or known node names.

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

**Related Information**

Design Space Explorer II on page 11

### 3.2.1. WYSIWYG Primitive Resynthesis

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

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

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

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

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

**Note:**

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

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

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

```
quartus_sh --qhelp
```

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

Use the following Tcl command to make a global assignment:

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

Use the following Tcl command to make an instance assignment:

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

Related Information

- Command Line Scripting
- Tcl Scripting
- API Functions for Tcl
  In Intel Quartus Prime Help
  For information about all settings and constraints in the Intel Quartus Prime software.

3.3.1. Synthesis Netlist Optimizations

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

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

#### Table 9. Synthesis Netlist Optimizations and Associated Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>Intel Quartus Prime Settings File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform WYSIWYG Primitive Resynthesis</td>
<td>ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Optimization Mode</td>
<td>OPTIMIZATION_MODE</td>
<td>BALANCEDHIGH PERFORMANCE EFFORT AGGRESSIVE PERFORMANCE</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Power-Up Don’t Care</td>
<td>ALLOW_POWER_UP_DONT_CARE</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
</tbody>
</table>

#### Table 10. Physical Synthesis Optimizations and Associated Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>Intel Quartus Prime Settings File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Physical Synthesis</td>
<td>ADVANCED_PHYSICAL_SYNTHESIS</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
</tbody>
</table>

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

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.04.24</td>
<td>18.1.0</td>
<td>Updated example in &quot;Netlist Optimizations and Physical Synthesis&quot; topic.</td>
</tr>
<tr>
<td>2019.04.18</td>
<td>18.1.0</td>
<td>Clarified wording in &quot;Netlist Optimizations and Physical Synthesis&quot; topic.</td>
</tr>
<tr>
<td>2018.09.24</td>
<td>18.1.0</td>
<td>Removed reference to unsupported CASCADE buffer from &quot;Optimize IOC Register Placement for Timing Logic Option&quot; topic.</td>
</tr>
<tr>
<td>2018.05.07</td>
<td>18.0.0</td>
<td>Removed topic: Isolating a Partition Netlist.</td>
</tr>
<tr>
<td>2017.11.06</td>
<td>17.1.0</td>
<td>• Removed reference to .vqm files</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added topic: Isolating a Partition Netlist.</td>
</tr>
<tr>
<td>2016.10.31</td>
<td>16.1.0</td>
<td>• Implemented Intel rebranding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated physical synthesis options and procedure.</td>
</tr>
<tr>
<td>2016.05.02</td>
<td>16.0.0</td>
<td>• Removed information about deprecated physical synthesis options.</td>
</tr>
<tr>
<td>2015.11.02</td>
<td>15.1.0</td>
<td>• Changed instances of Quartus II to Intel Quartus Prime.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Physical Synthesis.</td>
</tr>
<tr>
<td>2014.12.15</td>
<td>14.1.0</td>
<td>• Updated location of Fitter Settings, Analysis &amp; Synthesis Settings, and Physical Synthesis Optimizations Settings to Compiler Settings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated DSE II content.</td>
</tr>
<tr>
<td>June 2014</td>
<td>14.0.0</td>
<td>Updated format.</td>
</tr>
<tr>
<td>November 2013</td>
<td>13.1.0</td>
<td>Removed HardCopy device information.</td>
</tr>
</tbody>
</table>

*continued...*
# 3. Netlist Optimizations and Physical Synthesis

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2012</td>
<td>12.0.0</td>
<td>Removed survey link.</td>
</tr>
<tr>
<td>November 2011</td>
<td>10.0.2</td>
<td>Template update.</td>
</tr>
<tr>
<td>December 2010</td>
<td>10.0.1</td>
<td>Template update.</td>
</tr>
</tbody>
</table>
| July 2010        | 10.0.0                      | - Added links to Intel Quartus Prime Help in several sections.  
|                  |                             | - Removed Referenced Documents section.  
|                  |                             | - Reformatted Document Revision History |
| November 2009    | 9.1.0                       | - 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                       | - 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                       | - 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.
4. Area Optimization

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

4.1. Resource Utilization Information

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

The Compilation Report provides information about resource usage.

**Related Information**

Viewing Routing Congestion on page 116

4.1.1. Flow Summary Report

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

**Figure 12. Flow Summary Report**

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.
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

4.2.1. Resource Utilization Issues Overview

Resource utilization issues can be divided into three categories:

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

4.2.2. I/O Pin Utilization or Placement

Resolve I/O resource problems with these guidelines.

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

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

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

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

Related Information

Managing Device I/O Pins

4.2.3. Logic Utilization or Placement

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

4.2.3.1. Guideline: Optimize Source Code

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

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

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

Related Information
- AN 584: Timing Closure Methodology for Advanced FPGA Designs
- Recommended HDL Coding Styles
  In Intel Quartus Prime Pro Edition User Guide: Design Recommendations

4.2.3.2. Guideline: Optimize Synthesis for Area, Not Speed

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

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

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

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

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

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

**Related Information**
Optimization Mode

### 4.2.3.3. Guideline: Restructure Multiplexers

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

**Related Information**

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

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

The Perform WYSIWYG Primitive Resynthesis logic option specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the Optimization Technique logic option. The Perform WYSIWYG Primitive Resynthesis logic option is useful for resynthesizing some or all of the WYSIWYG primitives in your design for better area or performance. However, WYSIWYG primitive resynthesis can be done only when you use third-party synthesis tools.
The **Balanced** setting typically produces utilization results that are very similar to the **Area** setting with better performance results. The **Area** setting can give better results in some cases. Performing WYSIWYG resynthesis for area in this way typically reduces register-to-register timing performance.

**Related Information**
Perform WYSIWYG Primitive Resynthesis logic option
For information about this logic option

### 4.2.3.5. Guideline: Use Register Packing

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

**Related Information**
Auto Packed Registers logic option
For more information about the Auto Packed Registers logic option

### 4.2.3.6. Guideline: Remove Fitter Constraints

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

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

**Related Information**
Analyzing and Optimizing the Design Floorplan on page 111

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

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

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

If the Fitter cannot resolve a design due to memory resource limitations, the design may require a type of memory that the device does not have.
For memory blocks created with the Parameter Editor, edit the RAM block type to target a new memory block size.

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

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

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

### Related Information
- Inferring Shift Registers in HDL Code
- Fitter Resource Utilization by Entity Report

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

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

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

### Related Information
- Advanced Fitter Settings Dialog Box

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

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

If the DSP function was created with the parameter editor, open the parameter editor and edit the function so it targets logic cells instead of DSP blocks. The Intel Quartus Prime software uses the `DEDICATED_MULTIPLIER_CIRCUITRY` IP core parameter to control the implementation.
DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. You can turn off this inference in your synthesis tool. When you are using Intel Quartus Prime synthesis, you can disable inference by turning off the **Auto DSP Block Replacement** logic option for your entire project. Click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**. Turn off **Auto DSP Block Replacement**. Alternatively, you can disable the option for a specific block with the Assignment Editor.

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

### 4.2.3.11. Guideline: Use a Larger Device

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

### 4.2.4. Routing

Resolve routing resource problems with these guidelines.

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

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

**Related Information**

Auto Packed Registers logic option

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

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

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

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

**Related Information**

Fitter Aggressive Routability Optimizations logic option

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

The Router Effort Multiplier controls how quickly the router tries to find a valid solution. The default value is 1.0 and legal values must be greater than 0.
• Numbers higher than 1 help designs that are difficult to route by increasing the routing effort.
• Numbers closer to 0 (for example, 0.1) can reduce router runtime, but usually reduce routing quality slightly.

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

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

### 4.2.4.4. Guideline: Remove Fitter Constraints

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

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

Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

**Related Information**

Analyzing and Optimizing the Design Floorplan on page 111

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

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

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

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

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

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

**Related Information**

Optimization Mode

4.2.4.6. **Guideline: Optimize Source Code**

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

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

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

4.3. **Scripting Support**

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

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

   ```bash
   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:

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

3. Use the following Tcl command to make an instance assignment:

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

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

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

### 4.3.1. Initial Compilation Settings

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

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>.qsf File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Placement Effort Multiplier</td>
<td>PLACEMENT_EFFORT_MULTIPLIER</td>
<td>Any positive, non-zero value</td>
<td>Global</td>
</tr>
<tr>
<td>Router Effort Multiplier</td>
<td>ROUTER_EFFORT_MULTIPLIER</td>
<td>Any positive, non-zero value</td>
<td>Global</td>
</tr>
<tr>
<td>Router Timing Optimization level</td>
<td>ROUTER_TIMING_OPTIMIZATION_LEVEL</td>
<td>NORMAL, MINIMUM, MAXIMUM</td>
<td>Global</td>
</tr>
<tr>
<td>Final Placement Optimization</td>
<td>FINAL_PLACEMENT_OPTIMIZATION</td>
<td>ALWAYS, AUTOMATICALLY, NEVER</td>
<td>Global</td>
</tr>
</tbody>
</table>

### 4.3.2. Resource Utilization Optimization Techniques

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

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

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

### 4.4. Area Optimization Revision History

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018.10.18</td>
<td>18.1.0</td>
<td>• Corrected broken link to Optimization Modes Help topic.</td>
</tr>
<tr>
<td>2018.07.03</td>
<td>18.0.0</td>
<td>• Fixed typo and added links in topic Guideline: Retarget Memory Blocks.</td>
</tr>
<tr>
<td>2017.05.08</td>
<td>17.0.0</td>
<td>• Removed information about deprecated Integrated Synthesis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Revised topics: Resolving Resource Utilization Issues, Guideline: Optimize Synthesis for Area, Not Speed</td>
</tr>
<tr>
<td>2016.10.31</td>
<td>16.1.0</td>
<td>• Implemented Intel rebranding.</td>
</tr>
<tr>
<td>2016.05.02</td>
<td>16.0.0</td>
<td>• Removed information about deprecated physical synthesis options.</td>
</tr>
<tr>
<td>2015.11.02</td>
<td>15.1.0</td>
<td>• Changed instances of Quartus II to Intel Quartus Prime.</td>
</tr>
<tr>
<td>2014.12.15</td>
<td>14.1.0</td>
<td>• Updated location of Fitter Settings, Analysis &amp; Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.</td>
</tr>
<tr>
<td>June 2014</td>
<td>14.0.0</td>
<td>• Removed Cyclone III and Stratix III devices references.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed Macrocell-Based CPLDs related information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated template.</td>
</tr>
<tr>
<td>May 2013</td>
<td>13.0.0</td>
<td>• Initial release.</td>
</tr>
</tbody>
</table>

### Related Information

**Documentation Archive**

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.
5. 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.

Note: Some techniques are family specific.

5.1. Optimize Multi Corner Timing

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

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

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

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

5.2. 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 8
- Displaying Path Reports with the Timing Analyzer on page 77

### 5.2.1. Viewing Critical Paths

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

### 5.3. Critical Chains

The Intel Stratix® 10 device family uses the Hyper-Aware design flow to shorten design cycles and optimize performance. The Hyper-Aware design flow maximizes use of Hyper-Registers by combining automated register retiming with implementation of targeted timing closure recommendations (Fast Forward compilation). This sum of techniques drive the highest performance for Intel Stratix 10 designs.

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

For more information about improving design performance using the Hyper-Retimer critical chain reports, refer to the *Interpreting Critical Chain Reports* topic in the *Intel Stratix 10 High-Performance Design Handbook*.

**Related Information**
- Running the Hyper-Aware Design Flow
  - In *Intel Stratix 10 High-Performance Design Handbook*

### 5.3.1. Viewing Critical Chains

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

The critical chain is available at two different stages in the Hyper Aware Design Flow:
In the Retiming Limit Details Report:
This report is associated with the retiming stage in the Hyper Aware Design Flow, and is enabled by default.

In the Fast Forward Compilation Report:
The Fast Forward Compilation stage is optional, and disabled by default. You enable this stage from the Compilation Dashboard. Alternatively, start the task directly by clicking the **Fast Forward Timing Closure Recommendations** in the Compilations tasks.

You can also graphically visualize the critical chains in the Technology Map Viewer. For more details, refer to **Locate Critical Chains** in the *Intel Stratix 10 High-Performance Design Handbook*.

**Related Information**
- **Locate Critical Chains**
  * Intel Stratix 10 High-Performance Design Handbook
- **Intel Stratix 10 HyperFlex Design: Analyzing Critical Chains (OS10CRCHNS)**
  Online Course

**5.4. 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.

**5.4.1. Review Compilation Results**

**5.4.1.1. 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.

**5.4.1.2. 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.
5.4.1.3. Evaluate Optimization Results

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

5.4.1.4. Evaluate Resource Usage

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

5.4.1.4.1. 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**

<table>
<thead>
<tr>
<th>Location</th>
<th>Fan-Out</th>
<th>Global Resource Used</th>
<th>Global Line Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRACIALPLL_X98_Y2_N0</td>
<td>1</td>
<td>Global Clock</td>
<td>--</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y2_N1</td>
<td>29044</td>
<td>Global Clock</td>
<td>GLCK7</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y13_N1</td>
<td>253103</td>
<td>Global Clock</td>
<td>GLCK6</td>
</tr>
<tr>
<td>FF_X185_Y66_N13</td>
<td>260349</td>
<td>Global Clock</td>
<td>GLCK8</td>
</tr>
<tr>
<td>PN_AES1</td>
<td>4897</td>
<td>Global Clock</td>
<td>GLCK4</td>
</tr>
<tr>
<td>FRACIALPLL_X98_Y11_N0</td>
<td>1</td>
<td>Global Clock</td>
<td>--</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y3_N1</td>
<td>1691</td>
<td>Regional Clock</td>
<td>RCLK29</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y8_N1</td>
<td>302</td>
<td>Regional Clock</td>
<td>RCLK23</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y11_N1</td>
<td>141</td>
<td>Regional Clock</td>
<td>RCLK25</td>
</tr>
<tr>
<td>PLLOUTPUTCOUNTER_X98_Y10_N1</td>
<td>17</td>
<td>Regional Clock</td>
<td>RCLK22</td>
</tr>
</tbody>
</table>

If you assign these resources to a Regional Clock, the Global Clock becomes available for another signal. You can ignore signals with an empty value in the Global Line Name column as the signal uses dedicated routing, and not a clock buffer.

The Non-Global High Fan-Out Signals report lists the highest fan-out nodes not routed on global signals.

Reset and enable signals appear at the top of the list.

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

5.4.1.4.2. Routing Usage

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

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

![Fitter Resource Usage Summary Report](image)

**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. Routing Usage Summary Report**

![Routing Usage Summary Report](image)
5.4.1.4.3. 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 multicycle transfers between multi-rate clocks, and between different clock networks.

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

**Figure 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**

![Clock Delay Diagram]

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.

5.4.1.5. Evaluate Other Reports and Adjust Settings Accordingly

5.4.1.5.1. Difficulty Packing Design

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

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

5.4.1.5.2. 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.
5.4.1.5.3. 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.

5.4.1.5.4. 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.

5.4.1.5.5. 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.

5.4.1.5.6. 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.
5.4.1.5.7. 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)** mode, which reduces Fitter effort and compilation time as soon as timing requirements are met. You can optionally select another **Optimization mode** to target performance, area, routability, power, or compile time.

To increase Fitter effort further, you can also enable the **Assignments ➤ Settings ➤ Compiler Settings ➤ Advanced Settings (Fitter) ➤ Fitter Effort** option. The default **Auto Fit** setting reduces Fitter effort once timing requirements are met. **Standard Fit (highest effort)** setting uses maximum effort regardless of the design’s requirements, leading to higher compilation time and more timing margin.
5.4.1.5.8. 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.

5.4.1.6. Evaluate Clustering Difficulty

You can evaluate clustering difficulty to help reach timing closure.

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

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

5.4.2. Review Details of Timing Paths

5.4.2.1. 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. The Extra Fitter Information tab is not available for Intel Stratix 10 devices.

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 121

5.4.2.2. 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.

5.4.2.2.1. Source Location

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

5.4.2.2.2. 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).
5. Timing Closure and Optimization

5.4.2.2.3. 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.

5.4.2.2.4. 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.

5.4.2.3. 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.

5.4.2.4. Suspicious Setup

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

One typical cause is math precision error. For example, 10Mhz/3 = 33.33 ns per period. In three cycles, the time is 99.999 ns vs 100.000 ns. Setting a maximum delay can provide an appropriate setup relationship.
Another cause of failure are paths that must be false by design intent, such as:

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

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

5.4.2.5. 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.

5.4.2.6. 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.

5.4.2.7. 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 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. The Extra Fitter Information tab is not available for Intel Stratix 10 devices.

Related Information

Viewing Available Clock Networks in the Device on page 114
5.4.2.8. Timing Closure Recommendations

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

5.4.3. 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 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.

5.4.3.1. 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:

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 on page 126
- 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

5.5. 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.

5.5.1. 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

5.5.2. 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**

### 5.5.3. 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.

### 5.5.3.1. 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 **Extra Fitter Information** tab is not available for Intel Stratix 10 devices.

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 Technology Map Viewer provides schematic, 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 Path**, and select **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** on page 16
- **Generating Timing Reports**

### 5.5.3.2. 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 121
- Design Evaluation for Timing Closure on page 62

5.5.3.3. 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

- 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

5.5.3.4. 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.
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.

### 5.5.3.5. 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:

```bash
report_timing -setup -npaths 50 -detail path_only \
    -to "main_system: main_system_inst|app_cpu:cpu|*" \n    -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:

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

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

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

**5.5.3.6. 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.

**5.6. Timing Optimization**

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

**5.6.1. 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. For Intel Stratix 10 devices, use the Fast Forward Timing Closure Recommendations report.
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. The **Extra Fitter Information** tab is not available for Intel Stratix 10 devices.

**Related Information**
- Fast Forward Timing Closure Recommendations on page 101
- Report Timing Closure Recommendations Dialog Box In Intel Quartus Prime Help

**5.6.2. 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.
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 indicate recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icons indicate general suggestions. For these entries, the advisor does not report whether these recommendations were followed, but instead explains how you can achieve better performance. For a legend that provides more information for each icon, refer to the “How to use” page in the Timing Optimization Advisor.

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

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

### 5.6.3. Optional Fitter Settings

This section focuses only on the optional timing-optimization Fitter settings, which are **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](https://www.intel.com) in Intel Quartus Prime Help
5.6.3.1. Optimize Hold Timing

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

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

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

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

![Optimize Hold Timing Option Fixing an Internal Hold Time Violation](image)

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**


5.6.3.2. 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 13. Fitter Aggressive Routability Optimizations Logic Option Settings

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always</td>
<td>The Fitter always performs aggressive routability optimizations. If you set the Fitter Aggressive Routability Optimizations logic option to <strong>Always</strong>, reducing wire utilization may affect the performance of your design.</td>
</tr>
<tr>
<td>Never</td>
<td>The Fitter never performs aggressive routability optimizations. If improving timing is more important than reducing wire usage, then set this option to <strong>Automatically</strong> or <strong>Never</strong>.</td>
</tr>
<tr>
<td>Automatically</td>
<td>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 <strong>Automatically</strong> or <strong>Never</strong>.</td>
</tr>
</tbody>
</table>

5.6.4. 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.

*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 14. Improving Setup and Clock-to-Output Times

<table>
<thead>
<tr>
<th>Order</th>
<th>Technique</th>
<th>Affects $t_{SU}$</th>
<th>Affects $t_{CO}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Verify that the appropriate constraints are set for the failing I/Os (refer to Initial Compilation: Required Settings)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Use timing-driven compilation for I/O (refer to Fast Input, Output, and Output Enable Registers)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Use fast input register (refer to Programmable Delays)</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>Use fast output register, fast output enable register, and fast OCT register (refer to Programmable Delays)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Decrease the value of Input Delay from Pin to Input Register or set Decrease Input Delay to Input Register = ON</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>Decrease the value of Input Delay from Pin to Internal Cells or set Decrease Input Delay to Internal Cells = ON</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>7</td>
<td>Decrease the value of Delay from Output Register to Output Pin or set Increase Delay to Output Pin = OFF (refer to Fast Input, Output, and Output Enable Registers)</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*continued...*
Optimize IOC Register Placement for Timing Logic Option on page 86
Fast Input, Output, and Output Enable Registers on page 86
Programmable Delays on page 87
Use PLLs to Shift Clock Edges on page 88
Use Fast Regional Clock Networks and Regional Clocks Networks on page 88
Spine Clock Limitations on page 88

Related Information
Required Settings for Initial Compilation on page 7

5.6.4.1. 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.

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:

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

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

Related Information
Optimize IOC Register Placement for Timing Logic Option
In Intel Quartus Prime Help

5.6.4.2. 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.
If the fast I/O setting is on, the register is always placed in the I/O element. If the fast I/O setting is off, the register is never placed in the I/O element. This is true even if the Optimize IOC Register Placement for Timing option is turned on. If there is no fast I/O assignment, the Intel Quartus Prime software determines whether to place registers in I/O elements if the Optimize IOC Register Placement for Timing option is turned on.

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

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

Related Information

- Fast Input Register logic option
- Fast Output Register logic option
- Fast Output Enable Register logic option
- Fast OCT Register logic option

5.6.4.3. 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.

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*

### 5.6.4.4. 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 $t_{SU}$ at the expense of $t_{H}$, 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 $t_{H}$

![Original vs. With PLL](image)

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

### 5.6.4.5. 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.

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

**Related Information**
- Viewing Available Clock Networks in the Device on page 114
- Layers Settings on page 112
- Report Spine Clock Utilization dialog box (Chip Planner) in *Intel Quartus Prime Help*

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

The next stage of design optimization seeks to improve register-to-register \( (f_{\text{MAX}}) \) 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**

### 5.6.5.1. 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 or location assignments.

Be aware of the number of logic levels needed to implement your logic while you are coding. Too many levels of logic between registers might result in critical paths failing timing. Try restructuring the design to use pipelining or more efficient coding.
techniques. Also, try limiting high fan-out signals in the source code. When possible, duplicate and pipeline control signals. Make sure the duplicate registers are protected by a preserve attribute, to avoid merging during synthesis.

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

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

Related Information
AN 584: Timing Closure Methodology for Advanced FPGA Designs

5.6.5.2. 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
   — Prevent Shift Register Inference
   — Use Other Synthesis Options Available in Your Synthesis Tool

5. To optimize for performance, turn on Advanced Physical Optimization

6. Try different Fitter seeds. If only a small number of paths are failing by small negative slack, then you can try with a different seed to find a fit that meets constraints in the Fitter seed noise.

   Note: Omit this step if a large number of critical paths are failing, or if the paths are failing by a long margin.
7. To control placement, make Logic Lock assignments.
8. Modify your design source code to fix areas of the design that are still failing timing requirements by significant amounts.
9. Make location assignments, or as a last resort, perform manual placement by back-annotating the design.

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

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

Related Information
- Design Space Explorer II on page 11
- Required Settings for Initial Compilation on page 7

5.6.5.3. 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.

Related Information
- Perform WYSIWYG Primitive Resynthesis Logic Option
- Optimization Technique Logic Option

In Intel Quartus Prime Help
5.6.5.4. 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

5.6.5.5. 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

5.6.5.6. 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.

5.6.5.7. 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.
5.6.5.8. Duplicate Registers for Fan-Out Control

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

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

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

You can manually create a register tree and group the endpoints in the RTL by leveraging your system-level knowledge about how best to disperse the signal throughout your design, but it can be time consuming and have a widespread impact. For more information about manually creating a register tree, refer to Manual Register Duplication on page 93.

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

- Estimated Physical Proximity
- Hierarchical Proximity

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

5.6.5.8.1. Manual Register Duplication

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

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

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

If you are using Maximum Fan-Out assignments, benchmark your design with and without these assignments to evaluate whether they give the expected improvement in timing performance. Use the assignments only when you get improved results.
You can manually duplicate registers in the Intel Quartus Prime software regardless of the synthesis tool used. To duplicate a register, apply the **Manual Logic Duplication** logic option to the register with the Assignment Editor.

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

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

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

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

where,

- `register_name` is the register to duplicate. To create a register tree from a chain, create a unique assignment for each register in the chain. `DUPLICATE_REGISTER` assignments are processed in the appropriate order if they apply to registers that drive each other in a chain.

- `num_duplicates` is the number of duplicates of the register to create (including the original). If the original signal has `M` fan-out, the average fan-outs of the duplicates are `M/N` but any individual duplicate may have more or fewer, at the discretion of the algorithm.

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

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

**Important:**

- Setting `PHYSICAL_SYNTHESIS` to OFF disables `DUPLICATE_REGISTER`.

- Unlike other physical synthesis optimizations, the `DUPLICATE_REGISTER` assignment does allow duplication of registers that feed asynchronous clears and registers having location assignments.

- The `DUPLICATE_REGISTER` assignment does not process registers that have any of the following conditions:
  - Registers drive global signals or clock signals.
  - Registers have timing assignments or exceptions applied to them.
  - Registers have a `preserve` attribute or a `PRESERVE_REGISTER` assignment.
  - Registers are marked as `don't touch`.
  - Registers drive or are driven by other partitions.
5.6.5.8.3. Automatic Register Duplication: Hierarchical Proximity

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

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

where,

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

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

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

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

Consider the following example illustration of a netlist with a register chain and hierarchical organization of the endpoints it drives. The `DUPLICATE_HIERARCHY_DEPTH` assignment duplicates the pipeline registers across hierarchies, as shown in Figure 35 on page 97.
In this case, `regZ` is the appropriate assignment target as it is the endpoint in a chain of four registers. There is a maximum of three duplication candidates in this example (`regZ`, `regY`, and `regX`), so the assignment value can be anywhere between 1 and 3. `regA` is not pulled into the hierarchy to preserve the timing and optimization of paths that precede it. The `DUPLICATE_HIERARCHY_DEPTH` assignment is best used when a signal must be duplicated to more than 100 duplicates and the sub-hierarchy below the chain is deep and meaningful enough to guide the structure of the tree required.

When `num_levels` is set to 1, only `regZ` is pulled out of the chain and pushed down one hierarchy level into its fan-out tree.
Figure 34. **Netlist After Duplicating regZ to Hierarchy Level Two**

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

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

Figure 35. **Registers Duplicated Across Hierarchies**

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

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

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

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

5.6.5.10. Use Other Synthesis Options Available in Your Synthesis Tool

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

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

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

5.6.5.11. Fitter Seed

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

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

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

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

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

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

To specify a Fitter seed use the following Tcl command:

```
set_global_assignment -name SEED <value>
```
Related Information
Design Space Explorer II on page 11

5.6.5.12. Set Maximum Router Timing Optimization Level

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

Related Information
Router Timing Optimization Level Logic Option
In **Intel Quartus Prime Help**

5.6.6. 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 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.

5.6.7. 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
5.6.8. Intel Stratix 10 Timing Closure Recommendations

Note: This section applies only to designs targeting the Intel Stratix 10 family of devices. Other families do not have the capabilities described in this section.

In traditional FPGA timing closure flows, the starting point for most design analysis is the critical path. Due to the nature of Intel Stratix 10 devices and the availability of the Hyper Retimer, it is best to start your timing closure activities from the Retiming Limit Report. You want to give the tool as many optimization opportunities as possible before having to look into more time intensive and potentially manual timing closure techniques.

Related Information
- Interpreting Critical Chain Reports
  In Intel Stratix 10 High-Performance Design Handbook

5.6.8.1. Retiming Limit Details Report

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

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

5.6.8.1.1. Using the Retiming Limit Details Report

To access the Retiming Limit Details report:

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

The Technology Map Viewer displays a schematic representation of the complete critical chain after place, route and register retiming.
5.6.8.2. Fast Forward Timing Closure Recommendations

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

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

The Fast Forward Details Report provides the following information:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Step                        | Displays the various Fast Forward optimization steps, starting from the pre-optimization base compilation.  
|                             | • Each step comes with its associated critical chain.  
|                             | • Each step corresponds to a new optimization cumulative to the previous step.                                                              |
| Fast Forward Optimization    | Analyzed Summary of the optimizations necessary to implement each step.                                                                        |
| Estimated $f_{\text{MAX}}$  | Estimated $f_{\text{MAX}}$ performance after you implement the recommendations for this step in your design. This is cumulative, and step $n$ represents the potential $f_{\text{MAX}}$ after implementing all previous steps. |
| Optimization Analyzed       | (cumulative) List of all the consecutive optimization steps applied.                                                                           |
| Recommendation for Critical Chain | Lists recommended changes to your designs. These recommendations are geared towards removing retiming limitations, and allowing register movement. |

Related Information

- Interpreting Fast Forward Compile Reports  
  In *Intel Stratix 10 High-Performance Design Handbook*
- Accepting or Rejecting Fast Forward Optimizations  
  In *Hyper-Optimization for Intel Stratix 10 Designs*
5.6.8.2.1. Generating Fast Forward Timing Closure Recommendations

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

The Intel Quartus Prime Pro Edition software allows you to automate or refine Fast Forward analysis:
- To run Fast Forward compilation during each full compilation, click **Assignments ➤ Settings ➤ Compiler Settings ➤ HyperFlex**, and turn on **Run Fast Forward Timing Closure Recommendations during compilation**.
- To modify how Fast Forward compilation interprets specific I/O and block types, click **Assignments ➤ Settings ➤ Compiler Settings ➤ HyperFlex Advanced Settings**.

5.6.8.2.2. Implementing Fast Forward Recommendations

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

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

For more information about implementing Fast Forward timing closure recommendations in your design, refer to the Implement Fast Forward Recommendations section of the Intel Stratix 10 High-Performance Design Handbook

**Related Information**

Implement Fast Forward Recommendations
In *Intel Stratix 10 High-Performance Design Handbook*

5.7. 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 37. 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.

**5.7.1. Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box**

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

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

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings ➤ Compiler Settings ➤ Advanced Settings (Fitter)**.
2. In the *Advanced Fitter Settings* dialog box, for the *Periphery to Core Placement and Routing Optimization* option, select one of the following options depending on how you want to direct periphery to core optimizations in your design:
a. Select **Auto** to direct the software to automatically identify transfers with tight timing windows, place the core registers, and route all connections to or from the periphery.

b. Select **On** to direct the software to globally optimize all transfers between the periphery and core registers, regardless of timing requirements.

*Note*: Setting this option to **On** in the **Advanced Fitter Settings** is not recommended. The intended use for this setting is in the Assignment Editor to force optimization for a targeted set of nodes or instance.

c. Select **Off** to disable periphery to core path optimization in your design.

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

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

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

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

For paths to appear in the Assignments Editor, you must first run Analysis & Synthesis on your design.

### 5.7.3. 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 16. Fitter Report - Periphery to Core Transfer Optimization (P2C) Summary

<table>
<thead>
<tr>
<th>From Path</th>
<th>To Path</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>Node 2</td>
<td>Placed and Routed—Core register is locked. Periphery to core/core to periphery routing is committed.</td>
</tr>
<tr>
<td>Node 3</td>
<td>Node 4</td>
<td>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.</td>
</tr>
<tr>
<td>Node 5</td>
<td>Node 6</td>
<td>Not Optimized—This occurs when P2C is set to Auto and the path is not optimized due to one of the following issues: 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.</td>
</tr>
</tbody>
</table>

### 5.8. 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

  For information about all settings and constraints in the Intel Quartus Prime software.
- Tcl Scripting
  In Intel Quartus Prime Pro Edition User Guide: Scripting
- Command Line Scripting
  In Intel Quartus Prime Pro Edition User Guide: Scripting

5.8.1. 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 17. Initial Compilation Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>.qsf File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimize IOC Register Placement For Timing</td>
<td>OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Optimize Hold Timing</td>
<td>OPTIMIZE_HOLD_TIMING</td>
<td>OFF, IO PATHS AND MINIMUM TPD PATHS, ALL PATHS</td>
<td>Global</td>
</tr>
</tbody>
</table>

Table 18. Advanced Compilation Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>.qsf File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router Timing Optimization level</td>
<td>ROUTER_TIMING_OPTIMIZATION_LEVEL</td>
<td>NORMAL, MINIMUM, MAXIMUM</td>
<td>Global</td>
</tr>
</tbody>
</table>

Related Information

Design Optimization Overview on page 6

5.8.2. I/O Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the I/O timing optimization settings.
Table 19. I/O Timing Optimization Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>.qsf File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimize IOC Register Placement For Timing</td>
<td>OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Fast Input Register</td>
<td>FAST_INPUT_REGISTER</td>
<td>ON, OFF</td>
<td>Instance</td>
</tr>
<tr>
<td>Fast Output Register</td>
<td>FAST_OUTPUT_REGISTER</td>
<td>ON, OFF</td>
<td>Instance</td>
</tr>
<tr>
<td>Fast Output Enable Register</td>
<td>FAST_OUTPUT_ENABLE_REGISTER</td>
<td>ON, OFF</td>
<td>Instance</td>
</tr>
<tr>
<td>Fast OCT Register</td>
<td>FAST_OCT_REGISTER</td>
<td>ON, OFF</td>
<td>Instance</td>
</tr>
</tbody>
</table>

5.8.3. 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 20. Register-to-Register Timing Optimization Settings

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>.qsf File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform WYSIWYG Primitive Resynthesis</td>
<td>ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Fitter Seed</td>
<td>SEED</td>
<td>&lt;integer&gt;</td>
<td>Global</td>
</tr>
<tr>
<td>Maximum Fan-Out</td>
<td>MAX_FANOUT</td>
<td>&lt;integer&gt;</td>
<td>Instance</td>
</tr>
<tr>
<td>Manual Logic Duplication</td>
<td>DUPLICATE_ATOM</td>
<td>&lt;node name&gt;</td>
<td>Instance</td>
</tr>
<tr>
<td>Optimize Power during Synthesis</td>
<td>OPTIMIZE_POWER_DURING_SYNTHESIS</td>
<td>NORMAL, OFF</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTRA_EFFORT</td>
<td></td>
</tr>
<tr>
<td>Optimize Power during Fitting</td>
<td>OPTIMIZE_POWER_DURING_FITTING</td>
<td>NORMAL, OFF</td>
<td>Global</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTRA_EFFORT</td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Register-to-Register Timing Optimization Techniques on page 89

5.9. Timing Closure and Optimization Revision History

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.07.01</td>
<td>19.1</td>
<td>Added important notes to Automatic Register Duplication: Estimated Physical Proximity and Automatic Register Duplication: Hierarchical Proximity topics.</td>
</tr>
</tbody>
</table>
| 2019.04.01       | 19.1                        | • Added more information about register duplication methods in Duplicate Logic for Fan-out Control topic.  
• Moved content related to manual register duplication from Duplicate Logic for Fan-out Control topic to a newly created sub-topic Manually Adding Duplicate Registers.  
• Added Automatic Register Duplication: Estimated Physical Proximity and Automatic Register Duplication: Hierarchical Proximity as new sub-topics under Duplicate Logic for Fan-out Control to describe automatic register duplication process. |

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2018.11.12       | 18.1.0                     | • 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                     | • Removed duplicated topic: Resource Utilization Optimization Techniques. The topic is now in the Area Optimization chapter.  
|                  |                            | • Removed reference to unsupported CARRY and CASCADE buffers from "Optimize I/OC Register Placement for Timing Logic Option" topic. |
| 2017.11.06       | 17.1.0                     | • Added support for Intel Stratix 10 Hyper-Retiming, Fast Forward compilation, and Fast Forward Viewer.  
|                  |                            | • Added topic about using partitions to achieve timing closure.  
|                  |                            | • Removed statement about applying physical synthesis optimizations in a portion of a design.  
|                  |                            | • Removed references to optimizing hold timing for selected paths.  
|                  |                            | • Updated logic options about resource utilization optimization settings. |
| 2017.05.08       | 17.0.0                     | • Added topic: Critical Paths.  
|                  |                            | • Updated Register-to-Register Timing and renamed to Register-to-Register Timing Analysis.  
|                  |                            | • Renamed topic: Timing Analysis with the Timing Analyzer to Displaying Path Reports with the Timing Analyzer.  
|                  |                            | • Removed (LUT-Based Devices) remark from topic titles.  
|                  |                            | • Renamed topic: Optimizing Timing (LUT-Based Devices) to Timing Optimization.  
|                  |                            | • Renamed topic: Improving Register-to-Register Timing Summary to Improving Register-to-Register Timing .  
|                  |                            | • Removed topics: Tips for Locating Multiple Paths to the Chip Planner, LogicLock Assignments and Hierarchy Assignments, .  
|                  |                            | • Removed reference to deprecated Fitter Effort Logic Option.  
|                  |                            | • Removed information about Pin Advisor and Resource Optimization Advisor.  
|                  |                            | • Removed figure: Clock Regions |
| 2016.10.31       | 16.1.0                     | • Implemented Intel rebranding. |
| 2016.05.02       | 16.0.0                     | • Removed information about deprecated physical synthesis options.  
|                  |                            | • Added information about monitoring clustering difficulty. |
| 2015.11.02       | 15.1.0                     | • Added: Periphery to Core Register Placement and Routing Optimization.  
|                  |                            | • Changed instances of Quartus II to Quartus Prime. |
| 2014.12.15       | 14.1.0                     | • Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.  
|                  |                            | • Updated DSE II content. |

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2014</td>
<td>14.0.0</td>
<td>• Dita conversion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Replaced Megafunction content with IP core content.</td>
</tr>
<tr>
<td>November 2013</td>
<td>13.1.0</td>
<td>• Added Design Evaluation for Timing Closure section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed Optimizing Timing (Macrocell-Based CPLDs) section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Optimize Multi-Corner Timing and Fitter Aggressive Routability Optimization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Timing Analysis with the Timing Analyzer to show how to access the Report All Summaries command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Ignored Timing Constraints to include a help link to Fitter Summary Reports with the Ignored Assignment Report information.</td>
</tr>
<tr>
<td>May 2013</td>
<td>13.0.0</td>
<td>• Renamed chapter title from Area and Timing Optimization to Timing Closure and Optimization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed design and area/resources optimization information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fitter Aggressive Routability Optimization.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tips for Analyzing Paths from/to the Source and Destination of Critical Path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tips for Locating Multiple Paths to the Chip Planner.</td>
</tr>
<tr>
<td>November 2012</td>
<td>12.1.0</td>
<td>• Updated &quot;Initial Compilation: Optional Fitter Settings&quot; on page 13–2, &quot;I/O Assignments&quot; on page 13–2, &quot;Initial Compilation: Optional Fitter Settings&quot; on page 13–2, &quot;Resource Utilization&quot; on page 13–9, &quot;Routing&quot; on page 13–21, and &quot;Resolving Resource Utilization Problems&quot; on page 13–43.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minor text edits throughout the chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Table 13–6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the &quot;Spine Clock Limitations&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed the Change State Machine Encoding section from page 19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed Figure 13-5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minor text edits throughout the chapter.</td>
</tr>
<tr>
<td>May 2011</td>
<td>11.0.0</td>
<td>• Reorganized sections in &quot;Initial Compilation: Optional Fitter Settings&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added new information to &quot;Resource Utilization&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added new information to &quot;Duplicate Logic for Fan-Out Control&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added links to Help</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Additional edits and updates throughout chapter</td>
</tr>
</tbody>
</table>

continued...
5. Timing Closure and Optimization

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| December 2010    | 10.1.0                     | • 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                     | • Moved Compilation Time Optimization Techniques section to new Reducing Compilation Time chapter  
                   |                     | • Removed references to Timing Closure Floorplan  
                   |                     | • Moved Smart Compilation Setting and Early Timing Estimation sections to new Reducing Compilation Time 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.
6. Analyzing and Optimizing the Design Floorplan

As FPGA designs grow larger in density, the ability to analyze the design for performance, routing congestion, and logic placement is critical to meet the design requirements. This chapter discusses how the Chip Planner and Logic Lock regions help you improve your design's floorplan.

Design floorplan analysis helps to close timing, and ensures optimal performance in highly complex designs. With analysis capability, the Intel Quartus Prime Chip Planner helps you close timing quickly on your designs. You can use the Chip Planner together with Logic Lock regions to compile your designs hierarchically and assist with floorplanning. Additionally, use partitions to preserve placement and routing results from individual compilation runs.

You can perform design analysis, as well as create and optimize the design floorplan with the Chip Planner. To make I/O assignments, use the Pin Planner.

Note: As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.


Related Information
- Early Place Flow
  In Intel Quartus Prime Pro Edition User Guide: Compiler
- Floorplanning a Partial Reconfiguration Design
  In Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration
- Managing Device I/O Pins

6.1. Design Floorplan Analysis in the Chip Planner

The Chip Planner simplifies floorplan analysis by providing visual display of chip resources. With the Chip Planner, you can view post- compilation placement, connections, and routing paths. You can also make assignment changes, such as creating and deleting resource assignments.
The Chip Planner showcases:

- Logic Lock regions
- Relative resource usage
- Detailed routing information
- Fan-in and fan-out connections between nodes
- Timing paths between registers
- Delay estimates for paths
- Routing congestion information

### 6.1.1. Starting the Chip Planner

To start the Chip Planner, select **Tools ➤ Chip Planner**. You can also start the Chip Planner by the following methods:

- Click the Chip Planner icon on the Intel Quartus Prime software toolbar.
- In the following tools, right-click any chip resource and select **Locate ➤ Locate in Chip Planner**:  
  - Compilation Report  
  - **Logic Lock Regions Window**  
  - Technology Map Viewer  
  - **Project Navigator** window  
  - Node Finder  
  - Simulation Report  
  - Report Timing panel of the Timing Analyzer

### 6.1.2. Chip Planner GUI Components

#### 6.1.2.1. Chip Planner Toolbar

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

#### 6.1.2.2. Layers Settings

The Chip Planner allows you to control the display of resources.

**Layers Settings Pane**

With the **Layers Settings** pane, you can manage the graphic elements that the Chip Planner displays.

You open the **Layers Settings** pane by clicking **View ➤ Layers Settings**. The **Layers Settings** pane offers layer presets, which group resources that are often used together. The **Basic**, **Detailed**, and **Floorplan Editing** default presets are useful for general assignment-related activities. You can also create custom presets tailored to your needs.

**Related Information**

- **Viewing Architecture-Specific Design Information** on page 114
• Layers Settings Dialog Box
  In Intel Quartus Prime Help

6.1.2.3. Locate History Window

As you optimize your design floorplan, you might have to locate a path or node in the Chip Planner more than once. The Locate History window lists all the nodes and paths you have displayed using a Locate in Chip Planner command, providing easy access to the nodes and paths of interest to you.

If you locate a required path from the Timing Analyzer Report Timing pane, the Locate History window displays the required clock path. If you locate an arrival path from the Timing Analyzer Report Timing pane, the Locate History window displays the path from the arrival clock to the arrival data. Double-clicking a node or path in the Locate History window displays the selected node or path in the Chip Planner.

6.1.2.4. Chip Planner Floorplan Views

The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the targeted Intel device. As you zoom in, the level of abstraction decreases, revealing more details about your design.

Bird’s Eye View

The Bird’s Eye View displays a high-level picture of resource usage for the entire chip and provides a fast and efficient way to navigate between areas of interest in the Chip Planner.

The Bird’s Eye View is particularly useful when the parts of your design that you want to view are at opposite ends of the chip, and you want to quickly navigate between resource elements without losing your frame of reference.

Properties Window

The Properties window displays detailed properties of the objects (such as atoms, paths, Logic Lock regions, or routing elements) currently selected in the Chip Planner. To display the Properties window, right-click the object and select View ➤ Properties.

Related Information

Bird's Eye View Window
  In Intel Quartus Prime Help

6.1.3. Viewing Design Elements in the Chip Planner

The Chip Planner allows you to locate and report details on various elements of your design, such as viewing available clock networks, routing congestion, I/O banks, and high-speed serial interfaces in the floorplan.

The following section described how to view various design elements in the Chip Planner.
6.1.3.1. Viewing Architecture-Specific Design Information

The Chip Planner allows you to view architecture-specific information related to your design. By enabling the options in the Layers Settings pane, you can view:

- **Device routing resources used by your design**—View how blocks are connected, as well as the signal routing that connects the blocks.
- **LE configuration**—View logic element (LE) configuration in your design. For example, you can view which LE inputs are used; whether the LE utilizes the register, the look-up table (LUT), or both; as well as the signal flow through the LE.
- **ALM configuration**—View ALM configuration in your design. For example, you can view which ALM inputs are used; whether the ALM utilizes the registers, the upper LUT, the lower LUT, or all of them. You can also view the signal flow through the ALM.
- **I/O configuration**—View device I/O resource usage. For example, you can view which components of the I/O resources are used, whether the delay chain settings are enabled, which I/O standards are set, and the signal flow through the I/O.
- **PLL configuration**—View phase-locked loop (PLL) configuration in your design. For example, you can view which control signals of the PLL are used with the settings for your PLL.
- **Timing**—View the delay between the inputs and outputs of FPGA elements. For example, you can analyze the timing of the DATAB input to the COMBOUT output.

**Related Information**

- Layers Settings on page 112
- Layers Settings Dialog Box
  In Intel Quartus Prime Help

6.1.3.2. Viewing Available Clock Networks in the Device

When you enable a clock region layer in the Layers Settings pane, you display the areas of the chip that are driven by global and regional clock networks. When the selected device does not contain a given clock region, the option for that category is unavailable in the dialog box.
Figure 38. 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 88
- Layers Settings on page 112
- Report Spine Clock Utilization dialog box (Chip Planner)

6.1.3.3. Viewing Clock Sector Utilization

The Chip Planner provides a visual representation of a design's clock sector utilization.

To generate the report in the Chip Planner:
1. In the Tasks pane, double-click Report Clock Sector Utilization to open the Report Clock Sector Utilization dialog box.
2. If you want the report to include the source nodes, turn on Report source nodes. The equivalent TCL command appears at the bottom of the Dialog Box.
3. Click OK.
The report output shows the most used clock sectors.

The Report pane displays a list of clock sectors, with colors according to utilization. The clock sector with the highest utilization appears in red, and the sector with least utilization appears in blue.

You can turn on or off the sector visibility from the Report pane. You can also highlight nodes, if applicable.

![Clock Sector Utilization Report](Figure 39)

**Related Information**

Report Clock Sector Utilization dialog box (Chip Planner)

*In Intel Quartus Prime Help*

### 6.1.3.4. Viewing Routing Congestion

The Report Routing Utilization task allows you to determine the percentage of routing resources in use following a compilation. This feature can identify zones with lack of routing resources, helping you to make design changes to meet routing congestion design requirements.
To view the routing congestion in the Chip Planner:

1. In the Tasks pane, double-click the Report Routing Utilization command to launch the Report Routing Utilization dialog box.
2. Click Preview in the Report Routing Utilization dialog box to preview the default congestion display.
3. Change the Routing Utilization Type to display congestion for specific resources. The default display uses dark blue for 0% congestion (blue indicates zero utilization) and red for 100%. You can adjust the slider for Threshold percentage to change the congestion threshold level.

The congestion map helps you determine whether you can modify the floorplan, or modify the RTL to reduce routing congestion. Consider:

- The routing congestion map uses the color and shading of logic resources to indicate relative resource utilization; darker shading represents a greater utilization of routing resources. Areas where routing utilization exceeds the threshold value that you specify in the Report Routing Utilization dialog box appear in red.
- To identify a lack of routing resources, you must investigate each routing interconnect type separately by selecting each interconnect type in turn in the Routing Utilization Settings dialog box.
- The Compiler's messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, can indicate difficulties fitting your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, show increased chances of not getting a valid fit.

**Related Information**

Viewing Routing Resources on page 123

### 6.1.3.5. Viewing I/O Banks

To view the I/O bank map of the device in the Chip Planner, double-click Report All I/O Banks in the Tasks pane.

### 6.1.3.6. Viewing High-Speed Serial Interfaces (HSSI)

The Chip Planner displays a detailed block view of the receiver and transmitter channels of the high-speed serial interfaces. To display the HSSI block view, double-click Report HSSI Block Connectivity in the Tasks pane.
6.1.3.7. Viewing the Source and Destination of Placed Nodes

The Chip Planner allows you to view the registered fan-in or fan-outs of nodes in compiled designs with the Report Registered Connections task. This report is different from the Generate Fanin/Fanout connections report in that the source and destination nodes appear without connection lines, which may obscure the view.

1. In the Chip Planner, select one or more nodes.
2. In the Task pane, double-click Report Registered Connections.
3. Select the options from the dialog box, and click OK.

The Reports pane displays the registered source and destination nodes. Turn on or off to switch visibility in the graphic view.
**Figure 41. Report Registered Connections**

![Image of the Chip Planner view showing connections]

### Related Information
- Viewing Fan-In and Fan-Out Connections of Placed Resources on page 119
- Expand Connections Command (View Menu)

### 6.1.3.8. Viewing Fan-In and Fan-Out Connections of Placed Resources

Displays the atoms that fan-in to or fan-out from a resource, including connectivity lines.

To display the fan-in or fan-out connections from a resource you selected,

1. In the Chip Planner toolbar, click the **Generate Fan-In Connections** icon or the **Generate Fan-Out Connections** icon.

2. To remove other connections that appear on the Chip Planner view, click the **Clear Unselected Connections** icon.

You can also perform these actions from the Chip Planner View menu.

### Related Information
- Viewing the Source and Destination of Placed Nodes on page 118
- Expand Connections Command (View Menu)

In **Intel Quartus Prime Help**
6.1.3.9. Viewing Immediate Fan-In and Fan-Out Connections

Displays the immediate fan-in or fan-out connection for the selected atom.

For example, when you view the immediate fan-in for a logic resource, you see the routing resource that drives the logic resource. You can generate immediate fan-ins and fan-outs for all logic resources and routing resources.

- To display the immediate fan-in or fan-out connections, click View ➤ Generate Immediate Fan-In Connections or View ➤ Generate Immediate Fan-Out Connections.
- To remove the connections displayed, use the Clear Unselected Connections icon in the Chip Planner toolbar.

6.1.3.10. Viewing Selected Contents

You can view a detailed report of the contents of any area that you select in the Chip Planner. When you view the contents of a selected area, Chip Planner generates a hierarchical, color coded list of the design elements in the selection. This functionality allows you to quickly determine where the Compiler places specific modules of the design.

Follow these steps to view selected contents in the Chip Planner:

1. In the Tasks pane, double-click Report Selection Contents. The Report Selection Contents dialog box appears.
2. Under Report design instances in selection, turn on or off Show registers names and Show combinational names to display names of those type in the report.

Figure 42. Report Selection Contents Dialog Box

3. Click OK. The report generates and displays the list of selected elements in the Reports pane.
4. To customize the color coding of report folders, right-click any report, and then click Properties. You can customize the Report Name, Report Color, and the Highlighted Area Minimum Size for the report.

6.1.4. Exploring Paths in the Chip Planner

Use the Chip Planner to explore paths between logic elements. The following examples use the Chip Planner to traverse paths from the Timing Analysis report.

6.1.4.1. Analyzing Connections for a Path

To determine the elements forming a selected path or connection in the Chip Planner, click the Expand Connections icon in the Chip Planner toolbar.
6.1.4.2. Locate Path from the Timing Analysis Report to the Chip Planner

To locate a path from the Timing Analysis report to the Chip Planner, perform the following steps:

1. Select the path you want to locate in the Timing Analysis report.
2. Right-click the path and point to **Locate Path ▶ Locate in Chip Planner**. The path appears in the **Locate History** window of the Chip Planner.

![Path List in the Locate History Window](image)

**Related Information**

Displaying Path Reports with the Timing Analyzer on page 77

6.1.4.3. Show Delays

With the **Show Delays** feature, you can view timing delays for paths appearing in Timing Analyzer reports. To access this feature, click **View ▶ Show Delays** in the main menu. Alternatively click the Show Delays icon 📉 in the Chip Planner toolbar. To see the partial delays on the selected path, click the “+” sign next to the path delay displayed in the **Locate History** window.

For example, you can view the delay between two logic resources or between a logic resource and a routing resource.
6.1.4.4. Viewing Routing Resources

With the Chip Planner and the Locate History window, you can view the routing resources that a path or connection uses. You can also select and display the Arrival Data path and the Arrival Clock path.

Figure 47. 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 48.  **Highlight Routing**

To see the rows and columns where the Fitter routed the path, right-click a path and select **Highlight Routing**.

![Highlight Routing Diagram](image)

**Related Information**

- Viewing Routing Congestion on page 116

### 6.1.5. Viewing Assignments in the Chip Planner

You can view location assignments in the Chip Planner by selecting the appropriate layer, or any custom preset that displays block utilization in the **Layers Settings** pane.

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

Figure 49.  **Viewing Assignments in the Chip Planner**

![Viewing Assignments Diagram](image)

To create or move an assignment, or to make node and pin location assignments to Logic Lock regions, drag the selected resource to a new location. The Fitter applies the assignments that you create during the next place-and-route operation.
6.1.6. Viewing High-Speed and Low-Power Tiles in the Chip Planner

Some Intel devices have ALMs that can operate in either high-speed mode or low-power mode. The power mode is set during the fitting process in the Intel Quartus Prime software. These ALMs are grouped together to form larger blocks, called “tiles”.

To view a power map, double-click **Tasks ➤ Core Reports ➤ Report High-Speed/Low-Power Tiles** after running the Fitter. The Chip Planner displays low-power and high-speed tiles in contrasting colors; yellow tiles operate in a high-speed mode, while blue tiles operate in a low-power mode.

![Figure 50. High-Speed and Low Power Tiles in an Intel Arria 10 Device](image)

6.2. Creating Partitions and Logic Lock Regions with the Design Partition Planner and the Chip Planner

Using Logic Lock regions with design partitions allows you to preserve the location of a block while the Fitter works in other portions of the design. When you use the Design Partition Planner with the Chip Planner, you can create partitions and Logic Lock regions in a way that benefits both the connectivity and physical locations of entities.

To use this technique in an Intel Quartus Prime Pro Edition design:

1. Compile the design.
2. Open the Chip Planner and the Design Partition Planner.
   - Click **Tools ➤ Chip Planner**
   - Click **Tools ➤ Design Partition Planner**
3. In the **Chip Planner** window, go to the **Tasks** pane, and double-click **Report Design Partitions**.
The Report Design Partitions task causes the Chip Planner to display the physical locations of design entities using the same colors that the entities displayed in the Design Partition Planner.

4. In the Chip Planner, click **View ➤ Bird’s Eye View**
   The **Bird’s Eye** View opens.

5. In the Design Partition Planner, drag all the larger entities out from their parents.
   Alternatively, you can right-click the entity and click **Extract from Parent**.
   The Chip Planner displays the physical placement of the entities shown in the Design Partition Planner, with consistent colors between the two tools. You can view physical placement in the Chip Planner and connectivity in the Design Partition Planner.

6. Identify entities that are unsuitable to place in Logic Lock regions:
   — The Chip Planner shows an entity to be physically dispersed over noncontiguous areas of the device
   — The Design Partition Planner shows an entity to have a large number of connections to other entities.

7. Return entities unsuitable to place in Logic Lock regions to their parent, by dragging into the parent's entities.
   Alternatively, right-click the entity and click **Collapse to Parent**

8. Create a partition for each remaining entity by right-clicking the entity, and then clicking **Create Design Partition**.

9. Create a Logic Lock region for each partition by right-clicking the partition, and then clicking **Create Logic Lock Region**.

**Related Information**

Planning Design Partitions
   In *Intel Quartus Prime Pro Edition User Guide: Block-Based Design*

6.2.1. **Viewing Design Connectivity and Hierarchy**

By default, when you open a compiled design, the Design Partition Planner displays the design as a single top-level entity, containing lower-level entities. If the Design Partition Planner has opened the design previously, the design appears in its last state.
Figure 51. Top-Level Entity in the Design Partition Planner

- To show connectivity between entities, extract entities from the top-level entity by dragging them into the surrounding white space, or by right-clicking an entity and clicking **Extract from Parent** on the shortcut menu. When you extract entities, Design Partition Planner draws the connection bundles between entities, showing the number of connections between pairs of entities.

Figure 52. Partitioned Design with Connection Bundles

- To customize the appearance of connection bundles or to set thresholds for connection counts, click **View ➤ Bundle Configuration**, and set the necessary options in the **Bundle Configuration** dialog box.

- To see bundles containing failing paths, open the Timing Analyzer, and then click **View ➤ Show Timing Data** in the Design Partition Planner. Bundles containing failing paths are displayed in red, as are entities having nodes that reside on failing paths.

- To see detailed information about the connections in a bundle, right-click the bundle, and then click **Bundle Properties** to open the **Bundle Properties** dialog box.

- To switch between connectivity display mode and hierarchical display mode, click **View ➤ Hierarchy Display**. Alternatively, click and hold the hierarchy icon in the top-left corner of any entity to switch temporarily to a hierarchy display.
6.3. Using Logic Lock Regions in the Chip Planner

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

6.3.1. Viewing Connections Between Logic Lock Regions in the Chip Planner

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

The Chip Planner shows the connections between Logic Lock regions. By default, you can view each connection as an individual line. You can choose to display connections between two Logic Lock regions as a single bundled connection rather than as individual connection lines. To use this option, open the Chip Planner and on the View menu, click Inter-region Bundles.

Related Information
Inter-region Bundles Dialog Box
For more information about the Inter-region Bundles dialog box, refer to Intel Quartus Prime Help.

6.3.3. Logic Lock Regions

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

Note:
As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.

Logic Lock regions do not have preservation attributes, just boundaries and reservation of logic resources. You can use Intel Quartus Prime Pro Edition software to implement fully hierarchical Logic Lock region assignments.

A Logic Lock region is composed of two elements:

- Placement Region: Constrains logic to a specific area of the device; the Fitter places the logic in the region you specify. If you designate a region as Reserved, the Fitter cannot place other logic in the region.
- Routing Region: Constrains routing to a specific area. By default, routing regions are unconstrained. The routing region must encompass the placement region. A routing region cannot be reserved. For more details, refer to Defining Routing Regions.

Related Information
- Early Place Flow
  In Intel Quartus Prime Pro Edition User Guide: Compiler
- Creating Logic Lock Regions on page 129
### 6.3.4. Attributes of a Logic Lock Region

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

**Table 21. Attributes of Logic Lock Regions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>Number of columns</td>
<td>Specifies the width of the Logic Lock region. If Size/State is set to Auto/Floating, the attribute is set to Undetermined.</td>
</tr>
<tr>
<td>Height</td>
<td>Number of rows</td>
<td>Specifies the height of the Logic Lock region. If Size/State is set to Auto/Floating, the attribute is set to Undetermined.</td>
</tr>
<tr>
<td>Origin</td>
<td>Any Floorplan Location</td>
<td>Specifies the location of the Logic Lock region on the floorplan. The origin is at the lower left corner of the Logic Lock region.</td>
</tr>
<tr>
<td>Reserved</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Core-Only</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Size/State</td>
<td>Fixed/Locked</td>
<td>Auto/Floating</td>
</tr>
<tr>
<td></td>
<td>• If set to Fixed/Locked, the default value, you define the Logic Lock region's size and placement.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If set to Auto/Floating, the Fitter determines the size and placement of the Logic Lock region.</td>
<td></td>
</tr>
<tr>
<td>Routing Region</td>
<td>Unconstrained</td>
<td>Whole Chip</td>
</tr>
</tbody>
</table>

**Related Information**

Logic Lock Regions Window on page 138

### 6.3.5. Migrating Assignments between Intel Quartus Prime Standard Edition and Intel Quartus Prime Pro Edition

The Intel Quartus Prime Pro Edition software does not support the Intel Quartus Prime Standard Edition Logic Lock (Standard) assignments. Therefore, if you are migrating a design from Intel Quartus Prime Standard Edition to Intel Quartus Prime Pro Edition, you must convert the Logic Lock (Standard) assignments into Logic Lock assignments.

**Related Information**

Replace Logic Lock Regions  
In Intel Quartus Prime Pro Edition User Guide: Getting Started

### 6.3.6. Creating Logic Lock Regions
6.3.6.1. Creating Logic Lock Regions with the Chip Planner

1. Click **View ➤ Logic Lock Regions ➤ Create Logic Lock Region**
2. Click and drag on the Chip Planner floorplan to create a region of your preferred location and size

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape does not matter.

6.3.6.2. Creating Logic Lock Regions with the Project Navigator

1. Perform either a full compilation or analysis and elaboration on the design.
2. If the Project Navigator is not already open, click **View ➤ Utility Windows ➤ Project Navigator**. The Project Navigator displays the hierarchy of the design.
3. With the design hierarchy fully expanded, right-click any design entity, and click **Create New Logic Lock Region**.
4. Assign the entity to the new region.

The new region has the same name as the entity.

6.3.6.3. Creating Logic Lock Regions with the Logic Lock Regions Window

1. Click **Assignments ➤ Logic Lock Regions Window**.
2. In the **Logic Lock Regions** window, click **<<new>>**.

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape does not matter.

**Related Information**
Logic Lock Regions Window on page 138

6.3.6.4. Defining Routing Regions

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

Table 22. Routing Region Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained (default)</td>
<td>Allows the fitter to use any available routes on the device.</td>
</tr>
<tr>
<td>Whole Chip</td>
<td>Same as Unconstrained, but writes the constraint in the Intel Quartus Prime settings file (.qsf).</td>
</tr>
<tr>
<td>Fixed with Expansion</td>
<td>Follows the outline of the placement region. The routing region scales by a number of rows/cols larger than the placement region.</td>
</tr>
<tr>
<td>Custom</td>
<td>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.</td>
</tr>
</tbody>
</table>

Figure 53. Routing Regions

Figure 54. Noncontiguous Logic Lock Region

6.3.6.5. Noncontiguous Logic Lock Regions

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

Related Information
Merging Logic Lock Regions on page 133
6.3.6.6. Considerations on Using Auto Sized Regions

If you use Auto/Floating Size/State Logic Lock regions, take into account:

- **Auto/Floating** regions cannot be reserved.
- Verify that your Logic Lock region is not empty. If you do not assign any instance to the region, the Fitter reduces the size to 0 by 0, making the region invalid.
- The region may or may not be associated with a partition. When you combine partitions with Auto/Floating Size/State Logic Lock regions, you get flexibility to solve your particular fitting challenges. However, every constraint that you add reduces the solutions available, and too many constraints can result in the Fitter not finding a solution. Some cases are:
  - If a partition is preserved at synthesis or not preserved, the Logic Lock region confines the logic to a specific area, allowing the Fitter to optimize the logic within the partition, and optimize the placement within the Logic Lock region.
  - If a partition is preserved at placement, routed, or final; a Logic Lock region is not an effective placement boundary, because the location of the partition’s logic is fixed.
  - However, if the Logic Lock region is reserved, the Fitter avoids placing other logic in the area, which can help you reduce resource congestion.
- Once the outcome of the Logic Lock region meets your specification, you can:
  - Convert the Logic Lock region to Fixed/Locked Size/State.
  - Leave the Logic Lock region with Auto/Floating Size/State attribute and use the region as a "keep together" type of function.
  - If the Logic Lock region is also a partition, you can preserve the place and route through the partition and remove the Logic Lock region entirely.

6.3.7. Customizing the Shape of Logic Lock Regions

To create custom shaped Logic Lock regions, you can perform logic operations. Non-rectangular Logic Lock regions can help you exclude certain resources, or place parts of your design around specific device resources to improve performance.

**Attention:** There is no undo feature for the Logic Lock shapes for 17.1.

6.3.7.1. Adding a New Shape to a Logic Lock Region

To add a new shape to an existing Logic Lock region, perform the following steps in the Chip Planner:

1. Select the Logic Lock region.
2. In the Navigation toolbar, click the Add Logic Lock Region icon 📚.
3. Click and drag to generate the shape you want to add. The new shape merges automatically with the selected Logic Lock region.

**Attention:** If you selected more than one region, the operation appends the new shape to all them.
6.3.7.2. Subtracting Shape from Logic Lock Region

To subtract a shape from an existing Logic Lock region, perform the following steps in the Chip Planner:

1. Select the Logic Lock region.
2. In the Navigation toolbar, click the Subtract Logic Lock Region icon.
3. Click and drag the shape you want to subtract. The modified region displays automatically.

The operation performs in all selected regions.

6.3.7.3. Merging Logic Lock Regions

To merge two or more Logic Lock regions, perform the following steps:

1. Ensure that no more than one of the regions that you intend to merge has logic assignments.
2. Arrange the regions into the locations where you want the resultant region.
3. Select all the individual regions that you want to merge by clicking each of them while pressing the Shift key.
4. Right-click the title bar of any of the selected Logic Lock regions and select Logic Lock Regions ➤ Merge Logic Lock Region. The individual regions that you select merge to create a single new region.

If you select multiple named regions, the Merge Logic Lock Region option is deactivated.
6.3.7.4. Noncontiguous Logic Lock Regions

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

Related Information
Creating Logic Lock Regions on page 129

6.3.8. Placing Device Resources into Logic Lock Regions

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

If a Logic Lock region boundary includes part of a device resource, the Intel Quartus Prime software allocates the entire resource to that Logic Lock region.

To add an instance using the Logic Lock Region window, right-click the region and select Logic Lock Properties ➤ Add. Alternatively, in the Intel Quartus Prime software you can drag entities from the Hierarchy viewer into a Logic Lock region’s name field in the Logic Lock Regions Window.
6.3.8.1. Empty Logic Lock Regions

Intel Quartus Prime allows you to have Logic Lock regions with no members. Empty regions are a tool to manage space in the FPGA for future logic. This technique only works when you set the regions to **Reserved**

Some reasons to use empty Logic Lock regions are:
- Preliminary floorplanning.
- Complex incremental builds.
- Team based design and interconnect logic.
- Confining logic placements.

Since Logic Lock regions do not reserve any routing resources, the Fitter may use the area for routing purposes.

Use the **Core Only** attribute for empty Logic Lock regions. When you include periphery resources in empty regions, you restrict the periphery component placement, which can result in a no fit design. After you name the empty region, you can perform the same manipulations as with any populated Logic Lock Region.

**Figure 59. Logic Placed Outside of an Empty region**

The figure shows an empty Logic Lock region and the logic around it. However, some IOs, HSSIO, and PLLs are in the empty region. This placement happens because the output port connects to the IO, and the IO is always part of the root_partition (top-level partition).

6.3.8.2. Pin Assignment

A Logic Lock region incorporates all device resources within its boundaries, including memory and pins. The Intel Quartus Prime Pro Edition software does not include pins automatically when you assign an entity to a region, unless the **Core Only** attribute is off.

You can manually assign pins to Logic Lock regions; however, this placement puts location constraints on the region. The software only obeys pin assignments to locked regions that border the periphery of the device. The locked regions must include the I/O pins as resources.

6.3.8.3. Reserved Logic Lock Regions

The **Reserved** attribute instructs the Fitter to only place the entities and nodes that you specifically assigned to the Logic Lock region in the Logic Lock region.
The Intel Quartus Prime software honors all entity and node assignments to Logic Lock regions. Occasionally entities and nodes do not occupy an entire region, which leaves some of the region’s resources unoccupied.

To increase the region’s resource utilization and performance, Intel Quartus Prime software by default fills the unoccupied resources with other nodes and entities that have not been assigned to another region. To prevent this behavior, turn on **Reserved** in the **Logic Lock Regions** window.

### 6.3.8.4. Virtual Pins

A virtual pin is an I/O element that the Compiler temporarily maps to a logic element, and not to a pin during compilation. The software implements virtual pins as LUTs. To assign a Virtual Pin, use the Assignment Editor. You can create virtual pins by assigning the **Virtual Pin** logic option to an I/O element.

When you apply the **Virtual Pin** assignment to an input pin, the pin no longer appears as an FPGA pin; the Compiler fixes the virtual pin to GND in the design. The virtual pin is not a floating node.

Use virtual pins only for I/O elements in lower-level design entities that become nodes after you import the entity to the top-level design; for example, when compiling a partial design.

**Note:** The **Virtual Pin** logic option must be assigned to an input or output pin. If you assign this option to a bidirectional pin, tri-state pin, or registered I/O element, Synthesis ignores the assignment. If you assign this option to a tri-state pin, the Fitter inserts an I/O buffer to account for the tri-state logic; therefore, the pin cannot be a virtual pin. You can use multiplexer logic instead of a tri-state pin if you want to continue to use the assigned pin as a virtual pin. Do not use tri-state logic except for signals that connect directly to device I/O pins.

In the top-level design, you connect these virtual pins to an internal node of another module. By making assignments to virtual pins, you can place those pins in the same location or region on the device as that of the corresponding internal nodes in the top-level module. You can use the **Virtual Pin** option when compiling a Logic Lock module with more pins than the target device allows. The **Virtual Pin** option can enable timing analysis of a design module that more closely matches the performance of the module after you integrate it into the top-level design.

To display all assigned virtual pins in the design with the Node Finder, you can set **Filter Type** to **Pins: Virtual**. To access the Node Finder from the Assignment Editor, double-click the **To** field; when the arrow appears on the right side of the field, click and select **Node Finder**.

**Related Information**

- **Assigning Virtual Pins with a Tcl command** on page 145
- **Node Finder Command (View Menu)**
  - In [Intel Quartus Prime Help](https://www.intel.com/support)

### 6.3.8.5. Example: Placement Best Practices for Intel Arria 10 FPGAs

Logic Lock regions must take into account the device topology.
Note: As a best practice, define resource placement with iterative design flows. Use techniques like the Early Place Flow to guide your floorplanning decisions before setting hard placement constraints.

This example describes how I/O Columns constrain locations in Logic Lock regions in designs targeting Intel Arria 10 FPGAs.

**Figure 60. I/O Columns in Intel Arria 10 FPGAs**

Intel Arria 10 FPGAs have I/O columns located in the middle of the device. Signals can only enter or exit these columns from the side that faces the device edge.

**Figure 61. Signals Crossing I/O Columns in Intel Arria 10 FPGAs**

Routing a signal to cross the I/O column increases the routing delay, and can reduce design performance.

**Figure 62. Strategic Placement for Logic Lock Regions in Intel Arria 10 FPGAs**

- If a Logic Lock region contains a register that interface with the I/O column, place the Logic Lock region so that the region covers the I/O column and the core logic, for better access to the I/O column adjacent to the outer column edge.
- For high speed signal, you can get best results if you place the Logic Lock region on the outside of the I/O column, because the fitter is less likely to cross the column and incur delay.
6.3.9. Hierarchical Regions

Logic Lock regions are fully hierarchical. Parent regions must completely contain all child regions. The Reserved and Core-Only assignments also apply hierarchically.

Logic Lock assignments follow the same precedence as other constraints and assignments.

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

6.3.10. Additional Intel Quartus Prime Logic Lock Design Features

To complement the Logic Lock Regions Window, the Intel Quartus Prime software has additional features to help you design with Logic Lock regions.

6.3.10.1. Intel Quartus Prime Revisions Feature

When you evaluate different Logic Lock regions in your design, you might want to experiment with different configurations to achieve your desired results. The Intel Quartus Prime Revisions feature allows you to organize the same project with different settings until you find an optimum configuration.

To use the Revisions feature, choose Project ➤ Revisions. You can create a revision from the current design or any previously created revisions. Each revision can have an associated description. You can use revisions to organize the placement constraints created for your Logic Lock regions.

6.3.11. Logic Lock Regions Window

The Logic Lock Regions Window provides a summary of all Logic Lock regions defined in your design. Use the Logic Lock Regions Window to create, assign elements, and modify properties of a Logic Lock region.

Open the Logic Lock Regions Window in the Chip Planner by clicking View ➤ Logic Lock Window, and in Intel Quartus Prime by clicking Assignments ➤ Logic Lock Window.
You can customize the Logic Lock Regions Window by dragging and dropping the columns to change their order; you can also show and hide optional columns by right-clicking any column heading and then selecting the appropriate columns in the shortcut menu.

**Logic Lock Regions Properties Dialog Box**

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

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

**Related Information**

- Attributes of a Logic Lock Region on page 129
- Creating Logic Lock Regions with the Logic Lock Regions Window on page 130
- Logic Lock Regions Window
  In Intel Quartus Prime Help

### 6.3.12. Snapping to a Region

Chip Planner supports snap-to-lab for the Logic Lock region, where the Logic Lock region created is always snapped to a lab. It is applicable to clock regions in Intel Arria 10 and Intel Agilex® FPGAs, and clock sector in Intel Stratix 10 FPGA.

By default, Logic Lock regions are always snapped to the lab. You can change the default by clicking View > Logic Lock Regions > Snap Logic Lock Region to.

**Note:**

As observed in the following image, clock regions or sectors are visible (with orange boundary) when you perform an operation in Logic Lock region (for example, create, resize, or move the region) and snap-to-clock-region. It is not visible for snap-to-lab.
When you snap to a region, the Logic Lock region boundaries are displayed in the interactive mode. You can observe following behaviors:

- **Creating Region**: Left-click on the mouse to create the Logic Lock region. Upon releasing the mouse, the created Logic Lock region snaps to the containing clock region or sector.
- **Resize region (and resize diagonal)**: Left-click on the mouse and drag the Logic Lock region handle. Upon releasing the mouse, the Logic Lock region resizes and snaps to the containing clock region or sector.
- **Move region**: Select and drag the Logic Lock region to highlight the clock region boundaries. Upon releasing the mouse button, the Logic Lock region moves to the new position and snaps to the containing clock region or sector.
  - **Same place and route regions are moved**: Both Logic Lock regions move and snap to the containing clock sectors.
  - **Only place | route region is moved**: The selected region moves and snaps to the clock sector, and prompts warning if the new location or size of the region does not adhere to 'place bboxes contained within route bboxes' rule.
- **Subtract or make a hole**: When performing subtract in the snap-to-clock-region mode, you create a region where the region is snapped to a clock region or a sector, and then subtract away.

### 6.4. Using User-Defined Clock Regions in the Chip Planner

You can easily create and manipulate clock regions in the Chip Planner and make clock assignments to the regions.

You can create a user-defined clock region assignment to ensure that a given global clock signal is available to resources in a certain area of the device throughout all future design iterations. In instances of congestion involving global signal resources, you may specify a smaller clock region assignment to prevent a signal from using congested clock resources in other sectors.
If you create user-defined clock regions and subsequently compile the design, those user-defined clock regions are then shown as Fitter-defined clock regions, and can no longer be edited.

**Summary of User-Defined Clock Region Feature Support**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Clock Region Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shapes of clock regions.</td>
<td>Limited to rectangular regions that snap to clock sector grids.</td>
</tr>
<tr>
<td>Peripheral element assignments.</td>
<td>Limited to clocking design elements.</td>
</tr>
<tr>
<td>Clock region name.</td>
<td>Identified by the source clocking design element.</td>
</tr>
<tr>
<td>Support for multiple instances in the same regions.</td>
<td>Create one region per clock design element, and then specify the same definition for multiple clock design elements to assign to the same clock region.</td>
</tr>
</tbody>
</table>

**Using Clock Region Assignments in Intel Stratix 10 and Intel Agilex Devices**

You can constrain clock regions to a rectangle whose dimensions are defined by the sector grid, as seen in the Clock Sector Region layer of the Chip Planner. The rectangle is defined by the coordinates of its bottom-left and top-right corners. For example, $\text{SX0, SY0, SX1, SY1}$ constrains the clock to a $2 \times 2$ region, from the bottom left of sector 0,0 to the top right of sector 1,1.

The bounding rectangle can also be specified in chip coordinates, for example $X37 Y181 X273 Y324$; however, such a constraint should be sector-aligned. The Fitter automatically snaps to the smallest sector-aligned rectangle that encompasses the original assignment.

**6.4.1. Creating Clock Assignments with the Chip Planner**

1. Select the Create Clock Assignment icon, OR click View ➤ Clock Assignments ➤ Create Clock Assignment.

2. Click and drag on the Chip Planner floorplan to create a clock region of your preferred location and size. The region that you draw snaps to the smallest clock sector capable of containing the region. During interactive clock assignment operations such as creating, moving, or resizing, clock sector grids are shown in orange to facilitate positioning of the clock region relative to the clock sectors.

A clock symbol in the region title bar identifies it as a clock region. By default, the newly created clock region has the name *unassigned*, until you assign a clock signal from the context menu.
6.4.2. Resizing a Clock Assignment

1. Select the clock assignment. Handles appear on each side of the region and at the corners.
2. Position the crosshairs over the handle of your choice and the resize mouse cursor appears.
3. Hold the left mouse button and drag the resize cursor to expand or shrink the boundary of the clock assignment. When you release the mouse button, the clock assignment boundaries snap to the nearest containing clock sector grid.

6.4.3. Moving a Clock Assignment

1. Select the clock assignment.
2. Position the crosshairs over the clock assignment title bar and the move cursor appears.
3. Hold the left mouse button and drag the clock assignment to the desired new location.

6.4.4. Deleting a Clock Region Assignment
1. Select the clock assignment that you want to delete.
2. Right-click the clock assignment title bar to display the context menu, OR select View from the main menu bar.
3. Click Clock Assignments ➤ Delete Clock Assignment.
4. You are prompted to confirm that you want to delete the selected clock assignment. Click Yes to confirm the deletion.

The specified clock region assignment is deleted from the system.

6.4.5. Assigning a Clock Signal to a Clock Region

1. Right-click the clock assignment title bar to display the context menu, OR select View from the main menu bar.
2. Click Clock Assignments ➤ Set Clock Signal Name.
3. In the Set Clock Signal Name dialog box, browse to or type the desired clock signal name.
4. Click Ok.

The system renames the clock assignment according to the name of the specified clock signal.

6.4.6. Clock Assignment Properties

The Clock Assignment Properties pane displays properties of the selected clock assignment.

By default, the Clock Assignment Properties pane appears on a tab at the right side of the Chip Planner.

Figure 66. Clock Assignment Properties Pane
6.5. Scripting Support

You can run procedures and specify the settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt.

Related Information
- Tcl Scripting
  In Intel Quartus Prime Pro Edition User Guide: Scripting
- Command Line Scripting
  In Intel Quartus Prime Pro Edition User Guide: Scripting

6.5.1. Creating Logic Lock Assignments with Tcl commands

The Intel Quartus Prime software supports Tcl commands to create or modify Logic Lock assignments.

Note: Specify node names by using the full hierarchy path to the node.

Create or Modify a Placement Region

You can create the Logic Lock region from the GUI, or add the region directly to the QSF. The QSF entry contains the X/Y coordinates of the vertices and the Placement Region name.

The following assignment creates a new placement region with bounding box coordinates X46 Y36 X65 Y49:

```
set_instance_assignment -name PLACE_REGION "X46 Y36 X65 Y49" -to <node names>
```

- You can use the same command format to modify an existing assignment.
- To specify a non-rectangular or disjoint region, use a semicolon (;) as the delimiter between two or more bounding boxes.
- Assign multiple instances to the same region with multiple PLACE_REGION instance assignments.

Create or Modify a Routing Region

The following assignment creates a routing region with bounding box coordinates X5 Y5 X30 Y30:

```
set_instance_assignment -name ROUTE_REGION -to <node names> "X5 Y5 X30 Y30"
```

- You can use the same command format to modify an existing assignment.
- All instances with a routing region assignment must have a respective placement region; the routing region must fully contain the placement region.

Specify a Region as Reserved

The following assignment reserves an existing region:

```
set_instance_assignment -name <instance name> RESERVE_PLACE_REGION -to <node names> ON
```

- You can only reserve placement regions.
Specify a Region as Core Only

By default, the Intel Quartus Prime Pro Edition software includes pins in Logic Lock assignments. To specify a region as core only (that is, periphery logic in the instance that is not constrained), use the following assignment:

```
set_instance_assignment -name <instance name> CORE_ONLY_PLACE_REGION -to <node names> ON
```

Related Information

Creating Logic Lock Regions on page 129

6.5.2. Assigning Virtual Pins with a Tcl command

Use the following Tcl command to turn on the virtual pin setting for a pin called my_pin:

```
set_instance_assignment -name VIRTUAL_PIN ON -to my_pin
```

Related Information

- Virtual Pins on page 136
- Node Finder Command (View Menu) in Intel Quartus Prime Help

6.5.3. Logic Lock Region Assignment Examples

These examples show the syntax of Logic Lock region assignments in the .qsf file. Optionally, enter these assignments in the Assignment Editor, the Logic Lock Regions Window, or the Chip Planner.

Example 1. Assign Rectangular Logic Lock Region

Assigns a rectangular Logic Lock region to a lower right corner location of (10,10), and an upper right corner of (20,20) inclusive.

```
set_instance_assignment -name PLACE_REGION -to a\b\c "X10 Y10 X20 Y20"
```

Example 2. Assign Non-Rectangular Logic Lock Region

Assigns instance with full hierarchical path "x\y\z" to non-rectangular L-shaped Logic Lock region. The software treats each set of four numbers as a new box.

```
set_instance_assignment -name PLACE_REGION -to x\y\z "X10 Y10 X20 Y50; X20 Y10 X50 Y20"
```

Example 3. Assign Subordinate Logic Lock Instances

By default, the Intel Quartus Prime software constrains every child instance to the Logic Lock region of its parent. Any constraint to a child instance intersects with the constraint of its ancestors. For example, in the following example, all logic beneath "a\b\c\d" constrains to box (10,10), (15,15), and not (0,0), (15,15). This result occurs because the child constraint intersects with the parent constraint.

```
set_instance_assignment -name PLACE_REGION -to a\b\c "X10 Y10 X20 Y20"
set_instance_assignment -name PLACE_REGION -to a\b\c\d "X0 Y0 X15 Y15"
```
Example 4. Assign Multiple Logic Lock Instances

By default, a Logic Lock region constraint allows logic from other instances to share the same region. These assignments place instance c and instance g in the same location. This strategy is useful if instance c and instance g are heavily interacting.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X20 Y20"
```

Example 5. Assigned Reserved Logic Lock Regions

Optionally reserve an entire Logic Lock region for one instance and any of its subordinate instances.

```
set_instance_assignment -name PLACE_REGION -to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment -name RESERVE_PLACE_REGION -to a|b|c ON

# The following assignment causes an error. The logic in e|f|g is not legally placeable anywhere:
# set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X20 Y20"

# The following assignment does *not* cause an error, but is effectively constrained to the box (20,10), (30,20), since the (10,10),(20,20) box is reserved
# for a|b|c
set_instance_assignment -name PLACE_REGION -to e|f|g "X10 Y10 X30 Y20"
```

6.6. Analyzing and Optimizing the Design Floorplan Revision History

The following revision history applies to this chapter:

Table 23. Document Revision History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.07.30</td>
<td>19.3.0</td>
<td>Added new Using User-Defined Clock Regions in the Chip Planner section.</td>
</tr>
<tr>
<td>2019.07.01</td>
<td>19.1.0</td>
<td>Added new &quot;Snapping to a Region&quot; topic that describes the Snap Logic Lock Region to option.</td>
</tr>
<tr>
<td>2019.04.01</td>
<td>19.1.0</td>
<td>• Added new &quot;Viewing Selected Contents&quot; topic that describes a new report listing selected design elements.</td>
</tr>
<tr>
<td>2018.09.24</td>
<td>18.1.0</td>
<td>• Added topic: Viewing Clock Sector Utilization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added topic: Viewing the Source and Destination of Placed Nodes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Renamed topic: Generating Fan-In and Fan-Out Connections to Viewing Fan-In and Fan-Out Connections of Placed Resources.</td>
</tr>
<tr>
<td>2018.05.07</td>
<td>18.0.0</td>
<td>• Added recommendations for using iterative methods for floorplanning.</td>
</tr>
<tr>
<td>2017.11.06</td>
<td>17.1.0</td>
<td>• Changed instances of LogicLock Plus to Logic Lock.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for auto-sized Logic Lock regions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for empty Logic Lock regions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added topics: Considerations on Using Auto Sized Regions, Creating Partitions and Logic Lock Regions with the Design Partition Planner and Chip Planner.</td>
</tr>
</tbody>
</table>

continued...
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2017.05.08       | 17.0.0                     | • Chapter reorganization and content update.  
|                  |                            | • Added figures: Clock Regions, Path List in the Locate History Window, Show Physical Routing, Using the Add Rectangle Feature, Using the Subtract Rectangle Feature, Creating a Hole in a LogicLock Region, Noncontiguous LogicLock Region, Routing Regions, Logic Placed Outside of an Empty Region.  
|                  |                            | • Updated figures: HSSI Channel Blocks, Highlight Routing, High-Speed and Low Power Tiles in an Arria 10 Device, Show Delays Highlight Routing, Viewing Assignments in the Chip Planner, LogicLock Plus Regions Window, Using the Merge LogicLock Plus Region Command.  
|                  |                            | • Created topics: Adding Rectangle to a LogicLock Plus Region, Subtracting Rectangle from a LogicLock Plus Region.  
|                  |                            | • Moved topic: Viewing Critical Paths to Timing Closure and Optimization chapter and renamed to Critical Paths.  
|                  |                            | • Renamed topic: Creating Non-Rectangular LogicLock Plus Regions to Merging LogicLock Plus Regions.  
|                  |                            | • Renamed topic: Chip Planner Overview to Design Floorplan Analysis in the Chip Planner.  
|                  |                            | • Renamed chapter from Analyzing and Optimizing the Design Floorplan with the Chip Planner to Analyzing and Optimizing the Design Floorplan.  
| 2016.10.31       | 16.1.0                     | • Implemented Intel rebranding.  
|                  |                            | • Added topic describing how to create a hole in a LogicLock Plus region.  
| 2016.05.02       | 16.0.0                     | Updated information on creating LogicLock Plus regions.  
| 2015.11.02       | 15.1.0                     | • Changed instances of Quartus II to Quartus Prime.  
|                  |                            | • Added information on how to use LogicLock regions.  
| 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                     | • 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                     | • Updated for the 10.1 release.  
| July 2010        | 10.0.0                     | • 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  

*continued...*
6. Analyzing and Optimizing the Design Floorplan

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Updated &quot;Viewing Critical Paths&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated several graphics</td>
</tr>
<tr>
<td>November 2009</td>
<td>9.1.0</td>
<td>• Updated format of Document revision History table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated supported device information throughout</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 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.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated &quot;Creating Nonrectangular LogicLock Regions&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added &quot;Selected Elements Window&quot; section</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated table 12-1</td>
</tr>
<tr>
<td>May 2008</td>
<td>8.0.0</td>
<td>• Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Chip Planner Tasks and Layers&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LogicLock Regions&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Back-Annotating LogicLock Regions&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LogicLock Regions in the Timing Closure Floorplan&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Reserve LogicLock Region&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Creating Nonrectangular LogicLock Regions&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Viewing Available Clock Networks in the Device&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Table 10–1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reserve LogicLock Region Design Analysis Using the Timing Closure Floorplan</td>
</tr>
</tbody>
</table>

Related Information

Documentation Archive

For previous versions of the Intel Quartus Prime Handbook, search the documentation archives.
7. Implementing Engineering Change Orders

In a typical FPGA project development cycle, the specification of the programmable logic portion of the design often changes during the design process. The Intel Quartus Prime software supports last-minute, targeted design changes (also known as engineering change orders (ECOs)), even after you fully compile the design.

ECOs typically occur during the design verification stage. For example, during verification you determine that the design requires a small change, such as changing a netlist connection, correcting a logic error in a LUT, or adjusting a PLL clock frequency. Implementing an ECO, rather than performing a full recompilation on your design, requires less time and only changes affected logic.

You specify the ECO commands in a Tcl script using the `::quartus::eco` package.

**Note:** The Intel Quartus Prime Pro Edition software supports ECOs for Intel Stratix 10 and Intel Agilex devices only.

7.1. Engineering Change Order Flow

1. Identify an ECO modification you want to make in a compiled design.
2. Determine if ECO commands support the change, by reviewing ECO Commands on page 151 and ECO Command Limitations on page 158.
3. Create a Tcl script, as ECO Tcl Script Example on page 150 shows.
4. Before running ECO compilation, click **Project ➤ Archive Project** and archive the compilation database and output file set.
5. Click **Processing ➤ Start ➤ Start ECO Compilation**.

![Perform ECO Compilation](image)

6. Specify the **ECO Tcl Script** file, and click **OK**. The Fitter processes the ECO commands and updates the finalized netlist. The Fitter generates an error if you specify any commands incorrectly. The changes apply when the Fitter processing completes.
By default, the Assembler then runs automatically following successful ECO compilation to generate programming files. To disable running the Assembler automatically during the flow, turn off the Assembler stage checkbox on the Compilation Dashboard.

7. View the ECO results in post-fit analysis tools, such as the Timing Analyzer, Netlist Viewer, or Chip Planner.

You can use the following command-line alternatives to the GUI:

- Invoke the ECO flow from the Tcl Console:
  ```
  execute_flow -eco <script>.tcl
  ```

  By default, the Assembler runs automatically following successful Tcl Console ECO compilation. To disable running the Assembler automatically, add the following to the project .qsf file:
  ```
  set_global_assignment -name FLOW_DISABLE_ASSEMBLER ON
  ```

- Run following command from the command-line:
  ```
  quartus_fit <project> -c <revision> --eco <script>.tcl
  ```

  When running from the command line, any active Intel Quartus Prime GUI application does not refresh automatically. You must close and reopen the project to refresh the GUI.

**Note:** If you rerun the Fitter on a design after implementing an ECO, the Fitter overwrites the ECO changes. If you want to permanently implement the ECO changes, you must also update RTL and IP parameters and then recompile the design.

**Related Information**

Interactive ECO Fitting on page 159

### 7.2. ECO Tcl Script Example

The following shows an example ECO Tcl script that modifies the lutmask and reroutes connections:

**Figure 68. ECO Tcl Script Example**

```bash
# Modify the lutmask
post_message "About to modify lutmask"
modify_lutmask -to inst -eqn "A|B"

# Reroute connections
post_message "About to reroute connections"
remove_connection -from d1|0 -to d1|ff0 -port D
make_connection -from one_and_inst[out] -to d1|ff0 -port D
```
7.3. Viewing ECO Compilation Reports

The Compiler generates a report showing the details of each ECO compilation that you run successfully. You can view the report contents in the **ECO Changes** report under **Fitter** in the Compilation Report. Alternatively, you can view this data in the generated `fit.eco` file.

The Compiler organizes the report output according to the category of ECO change, such as Placement Changes. The table specifies the "Changes in Previous ECO Runs" and "Changes in Current ECO Run".

**Figure 69. ECO Report Example**

```
+ ; Placement Changes +
+ ; Name ; Location ; Iteration +
+ ; GND-ECO_INSERTED ; X78_Y72 ; Previous +
+ ; new_node ; X78_Y27 ; Previous +
+ ; new_wirelut ; X78_Y27 ; Previous +
+ ; new_wirelut ; X24_Y24 ; Current +
```

7.4. ECO Commands

The Intel Quartus Prime Pro Edition software supports the following ECO commands:

- ECO Command Quick Reference on page 152
- `make_connection` on page 152
- `remove_connection` on page 153
- `modify_lutmask` on page 154
- `adjust_pll_refclk` on page 154
- `modify_io_slew_rate` on page 155
- `modify_io_current_strength` on page 155
- `modify_io_delay_chain` on page 155
- `create_new_node` on page 156
- `place_node` on page 157
- `create_wirelut` on page 157
### 7.4.1. ECO Command Quick Reference

<table>
<thead>
<tr>
<th>ECO Change</th>
<th>ECO Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Route</strong></td>
<td>make_connection -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt;</td>
</tr>
<tr>
<td></td>
<td>remove_connection -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt;</td>
</tr>
<tr>
<td><strong>Tie-Off</strong></td>
<td>make_connection -tieoff &lt;VCC/GND&gt; -to &lt;node&gt; -port &lt;port&gt;</td>
</tr>
<tr>
<td><strong>Lutmask</strong></td>
<td>modify_lutmask -to &lt;node&gt; [-eqn &lt;lut equation&gt;] [-mask 0x00]</td>
</tr>
<tr>
<td><strong>Slew Rate</strong></td>
<td>modify_io_slew_rate &lt;val&gt; -to &lt;pin_name&gt;</td>
</tr>
<tr>
<td><strong>Current Strength</strong></td>
<td>modify_io_current_strength &lt;val&gt; -to &lt;pin_name&gt;</td>
</tr>
<tr>
<td><strong>Delay Chains</strong></td>
<td>modify_io_delay_chain &lt;val&gt; -type &lt;io_type&gt; -to &lt;pin_name&gt;</td>
</tr>
<tr>
<td><strong>Update MIF</strong></td>
<td>update_mif_files</td>
</tr>
<tr>
<td><strong>IOPLL Ref Clock</strong></td>
<td>adjust_pll_refclk -to &lt;pll name&gt; -refclk &lt;freq&gt;</td>
</tr>
<tr>
<td><strong>Create New Node</strong></td>
<td>create_new_node -type LUT -name &lt;name&gt;</td>
</tr>
<tr>
<td><strong>Place Node</strong></td>
<td>place_node -name &lt;name&gt; [-location &lt;location&gt;]</td>
</tr>
<tr>
<td><strong>Create Wirelut</strong></td>
<td>create_wirelut -from &lt;src&gt; -to &lt;dst&gt; -port &lt;port&gt; [-location &lt;location&gt;]</td>
</tr>
</tbody>
</table>

### 7.4.2. make_connection

**Description**

Connects the source signal to the destination block port. If the port has an existing connection, the command removes the previous connection and connects it to the signal you specify. The actual routing change occurs implicitly when appropriate. You can locate node names by right-clicking a node in Netlist Viewer, and then clicking Properties.

`make_connection` also supports adding connections from and to Hyper-Registers. In the case of Hyper-Registers, the command first disconnects the destination port, before making a new connection. You can then run the `make_connection` command to specify a replacement signal source or destination.

If a port is shared among multiple atoms (for example, RAM), then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

**Usage**

The following example connects `top|a_out` to the D input port of node `top|x`.

```
make_connection -from top|a_out -to top|x -port D
```

(1) Support only for Intel Stratix 10 devices.
Arguments

from  Output net of the source block of the new connection.

to Name of the destination block.

port The input port name of the destination block.

Options

tieoff To explicitly tie off an input port to VCC or GND.

VCC or GND

Example:

```
make_connection -tieoff VCC -to {node1} -port DATAA
```

to Name of the destination block.

port The input port name of the destination block.

7.4.3. remove_connection

Description
Disconnects the src signal from the destination block port. The actual routing change occurs implicitly. You can locate node names by right-clicking a node in Netlist Viewer, and then clicking Properties.

If a port is shared among multiple atoms (for example, RAM), then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

Usage
The following example disconnects top|a_out from the D input port of node top|x, and set top|x:D to a disconnected state.

```
remove_connection -from top|a_out -to top|x -port D
```

Arguments

from  Output net of the source block of the current connection.

to Name of the destination block.

port The input port name of the destination block.
### 7.4.4. modify_lutmask

**Description**
Modifies the lutmask of the matching destination node, with the lutmask value (-mask) in binary or hexadecimal, or with the equivalent lutmask value (-eqn) computed from specified logical equation.

**Usage**
The following example disconnects `top|a_out` from the D input port of node `top|x`, and set `top|x:D` to a disconnected state.

```plaintext
modify_lutmask -to top|lut_c -eqn {a&b&c}
modify_lutmask -to top|lut_a -mask 0xFF00FF00
modify_lutmask -to top|lut_b -mask 0b111111111001010
```

**Arguments**

- **eqn**: The logical equation of the inputs (A, B, C, D, E, F). The supported lexical tokens include `AND('&')`, `OR('|')`, `XOR('^')`, `NOT('!')`, `OPEN_BRACE('(')`, `CLOSE_BRACE(')')`. Specify `-mask` or `-eqn` to `Destination atom name`.

- **mask**: The lutmask value to be modified in binary or hexadecimal format. Specify `-mask` or `-eqn` to `Destination atom name`.

**Note:**
When you view the lutmask equations in the Resource Property Viewer, the equations display in terms of F0/F1/F2/F3 LUTs for A, B, C and D inputs. For LUTs also using E or F inputs, you must combine these sub-functions using the connectivity that the ALM diagram shows for the E and F muxes.

### 7.4.5. adjust_pll_refclk

**Description**
Changes the IOPLL frequencies by modifying the input reference clock frequency. The following stipulations apply:

- Maintain the original refclk and outclk ratios.
- The IOPLLs you change cannot generate IP clocks.
- Cascaded IOPLLs must connect directly (no clock gates in between them).
- IOPLLs cannot be in 'nondedicated' compensation modes.
- For all IOPLLs, outclks duty cycle equals 50 and phase shift equals 0.
- No support for Intel Agilex devices.
7.4.6. modify_io_slew_rate

**Description**
Implements the I/O pin slew setting rate that you specify for the I/O pin.

**Usage**
modify_io_slew_rate 1 -to top|ipin

**Arguments**

*to*  Instance name of destination pin that you want to modify.

7.4.7. modify_io_current_strength

**Description**
Implements the change to the I/O pin current strength setting that you specify for the I/O pin.

**Usage**
modify_io_current_strength 3mA -to top|ipin

**Arguments**

*to*  Instance name of the destination pin that you want to modify.

7.4.8. modify_io_delay_chain

**Description**
Implements the change to the delay chain settings that you specify for the I/O pin.

**Usage**
modify_io_delay_chain 3 -to top|ipin -type input
Arguments

**type**  Specifies one of the following I/O types: input, output, oe, io_12_lane_input, io_12_lane_input_strobe

to  Instance name of I/O pin that you want to modify the delay chain settings.

7.4.9. create_new_node

**Description**

Creates a simple LUT cell in the design netlist. The following use cases apply:

- Creating a VCC/GND lcell for input port tieoff, when the LUT does not already exist in the design.
- Adding a gate to fix a logic bug.
- Adding a wire LUT to help add hold delay.

The name of the new node is hierarchical. Therefore, when creating node $a|b|c|d$, you must ensure that hierarchy $a|b|c$ exists in the netlist. If the source or destination node lies under a partition, the new LUT inserts under that partition.

**Note:**

This command does not support extended or arithmetic LUTs.

After creating the new node, you can run the following commands to connect, modify the lutmask, or place the new node:

1. Run `make_connection` to connect to the new LUT’s DATA inputs and output port.
2. Run `modify_lutmask` to change the lutmask for the new LUT.
3. Run `place_node` to place (and subsequently route) the new LUT.

This flow ensures that all routing requirements are analyzed when determining a legal placement for the new node.

**Usage**

The following example creates a new_lut LUT node with input ports DATAA and DATAB, and with outputs connected accordingly. `modify_lutmask` the modifies the lutmask to perform A&B logic. `place_node` next places the new LUT. The connections route after node placement is complete.

```plaintext
create_new_node -name new_lut -type lut
make_connection -from src_a -to new_lut -port DATAA
make_connection -from src_b -to new_lut -port DATAB
make_connection -from new_lut -to dst_reg -port D
modify_lutmask -to new_lut -eqn A&B
place_node -name new_lut
```

**Arguments**

**name**  Name of the new LUT node.
7.4.10. place_node

Description
Places the node that you specify in a location that the ECO Fitter selects. Optionally, you can specify the location argument to specify a particular device region location constraint. You can also run this command for nodes already placed by the Fitter.

Usage
The following examples show three placement cases. For node1, the ECO Fitter determines the placement location. For node2, the command specifies the exact LAB location constraint. For node3, the command specifies a placement region constraint.

```
place_node -name node1 # let ECO Fitter decide placement
place_node -name node2 -location "X1 Y1 X1 Y1" # place node at exact location
place_node -name node3 -location "X10 Y10 X20 Y20" # place node in region
```

Arguments

- `name` Name of the node.
- `location` Device region coordinates (X1 Y1 X10 Y10).

7.4.11. create_wirelut

Description
Creates and inserts a wire LUT node in the connection that you specify. The ECO Fitter places the new LUT and routes the modified connections automatically. Optionally, you can specify the location argument to specify a particular device region location constraint.

The name of the new node is hierarchical. Therefore, when creating node a|b|c|d, you must ensure that hierarchy a|b|c exists in the netlist. If the source or destination node lies under a partition, the new wire LUT inserts under that partition.

create_wirelut also supports adding connections from and to Hyper-Registers. In the case of Hyper-Registers, the command first disconnects the destination port, before making a new connection. You can then run the create_wirelut command to specify a replacement signal source or destination.

If a port is shared among multiple atoms (for example, RAM), then the ECO Fitter automatically updates all relevant atoms, and reports them accordingly.

Usage
The following example creates the my_wirelut wire LUT, connects my_wirelut output to the D input port of dest_node, and connects the output of src_output to the input port of the wire LUT. Finally, the ECO Fitter places the new node within region (20, 20) to (40, 40) and routes automatically.

```
create_wirelut -name my_wirelut -from src_output -to dest_node -port D -location "X20 Y20 X40 Y40"
```
Arguments

name  Name of the node.

From  Source of the connection.

To  Name of destination node.

Port  Input port name of destination node.

location  Device region coordinates (X1 Y1 X10 Y10).

7.5. ECO Command Limitations

The ECO commands have the following limitations due to connection dependencies within Intel FPGA devices. You can only use the ECO commands to modify connections using core routing resources.

- You cannot use ECO commands to modify dedicated connections, or connections routed using the global clock network.
- You can only remove connections that include core routing from the -from signal to the destination.
- You cannot modify dedicated connections within a single ALM. This limitation applies to direct connections between LUT and flip-flop nodes.
- You cannot modify connections using global clock routing resources.
- You can add additional connections to an existing Hyper-Register output, but you cannot remove any existing connections from a Hyper-Register.
- Use of the place_node command with location arguments does not overwrite the existing location or region assignments in the .qsf. Therefore, the ECO Fitter does not place nodes with set_location_assignment or place_region assignments outside the existing .qsf constraints.
- If a LAB already has the maximum number of legal connections where a node is placed, the place_node or make_connection commands can fail, preventing the connection to the first placed node that cannot be legalized. If that situation occurs, you can either move the original node to a different location, or move other nodes from the LAB to free up routing resources.
- When creating and connecting multiple new LUT nodes, you must run the place_node command to place the nodes one by one. The nodes you connect to must have an existing placement already.

The recommended order for creating and placing new LUTs is:

1. Create the node by using the create_new_node command.
2. Make connections to and from the node by using the make_connection command.
3. Update the lutmask by using the modify_lutmask command.
4. Place the node by using the place_node command.
This flow ensures that analysis includes all routing requirements when determining a legal placement for the new node. For example:

```bash
set lut_name new_lut
cREATE_new_node -name $lut_name -type lut
make_connection -from input1 -to $lut_name -port DATAA
make_connection -from input2 -to $lut_name -port DATAB
modify_lutmask -to $lut_name -eqn {A&B}
place_node -name $lut_name -location "X80 Y80 X85 Y95"
```

The `create_new_node` command does not support creation of new node that has any connection to other unplaced nodes. For example, `create_new_node` does not support creating two new adjacent nodes connected to each other.

**Note:** To minimize issues with name matching caused by escaped characters, it can be useful to surround entity names with `{}` characters, instead of `""`. This technique is particularly useful if entity names contain backslashes or any other special characters.

### 7.6. Interactive ECO Fitting

The `quartus_fit` executable supports ECO changes in an interactive shell through `quartus_fit -s`.

In an interactive context, the ECO Fitter legalizes the changes, when appropriate. For example, for `make_connection` changes immediately after node creation, the Fitter does not attempt to route the connections immediately; rather, the Fitter waits until after the placement of the node prior to routing.

#### 7.6.1. `eco_load_design` and `eco_commit_design` Commands

**Description**
- `eco_load_design`—loads the final netlist in the ECO context.
- `eco_commit_design`—commits the ECO modified netlist to disk while running in interactive mode.

**Usage**

The following example shows an interactive ECO session to modify the `lut_x` lutmask for project `top`. If the ECO modifications are legal, the `eco_commit_design` command commits the final netlist.

```
$ quartus_fit -s
>> load_package eco
>> project_open top
>> eco_load_design
>> modify_lutmask -to lut_x -eqn B&C # some ECO changes
>> eco_commit_design
>> project_close
```
### 7.7. Implementing Engineering Change Orders Revision History

The following revision history applies to this chapter:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2020.05.08        | 20.1                        | - Added descriptions of new `create_new_node`, `place_node`, and `create_wiredlut` commands.  
- Referenced support for multi-node ECOs in `make_connection`, `remove_connection`, and `create_wiredlut` command topics.  
- Referenced Support for ECO connections to Hyper-Registers in the `make_connection` topic.  
- Described updates to ECO reporting in "Viewing ECO Compilation Reports."  
- Updated limitations in "ECO Command Limitations."  
- Added ECO Command Quick Reference |
| 2019.09.30        | 19.3.0                      | - Added information about `tieoff` option for `make_connection` command.  
- Added support for `modify_io_slew_rate` command.  
- Added support for `modify_io_current_strength` command.  
- Added support for `modify_io_delay_chain` command.  
- Added "Viewing ECO Compilation Reports" topic.  
- Added information about `num` option for `modify_lutmask` command.  
- Mentioned RTL Viewer for locating node names.  
- Added device support note. |
| 2019.07.01        | 19.2.0                      | - First release of chapter. |

If the table does not list a software version, the user guide for the previous software version applies.

<table>
<thead>
<tr>
<th>Intel Quartus Prime Version</th>
<th>User Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.0</td>
<td>Design Optimization User Guide Intel Quartus Prime Pro Edition</td>
</tr>
</tbody>
</table>
A. Intel Quartus Prime Pro Edition User Guides

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

**Related Information**

- **Intel Quartus Prime Pro Edition User Guide: Getting Started**
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.

  Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.

  Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.

- **Intel Quartus Prime Pro Edition User Guide: Design Compilation**
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.

  Describes Intel Quartus Prime Pro Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).

  Describes operation of the Intel Quartus Prime Pro Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.

- **Intel Quartus Prime Pro Edition User Guide: Block-Based Design**
  Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.
• **Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration**
  Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.

• **Intel Quartus Prime Pro Edition User Guide: Third-party Simulation**
  Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Mentor Graphics*, and Synopsys that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.

• **Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis**
  Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics*, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.

• **Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools**
  Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin*.

• **Intel Quartus Prime Pro Edition User Guide: Debug Tools**
  Describes a portfolio of Intel Quartus Prime Pro Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or "tapping") signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.

• **Intel Quartus Prime Pro Edition User Guide: Timing Analyzer**
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.

• **Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization**
  Describes the Intel Quartus Prime Pro Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.

• **Intel Quartus Prime Pro Edition User Guide: Design Constraints**
  Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.

• **Intel Quartus Prime Pro Edition User Guide: PCB Design Tools**
  Describes support for optional third-party PCB design tools by Mentor Graphics* and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.

• **Intel Quartus Prime Pro Edition User Guide: Scripting**
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.