10. Design Optimization Overview

This chapter introduces features in Altera’s Quartus® II software that you can use to achieve the highest design performance when you design for programmable logic devices (PLDs), especially high density FPGAs.

Introduction

Physical implementation can be an intimidating and challenging phase of the design process. The Quartus II software provides a comprehensive environment for FPGA designs, delivering unmatched performance, efficiency, and ease-of-use.

In a typical design flow, you must synthesize your design with Quartus II integrated synthesis or a third-party tool, place and route your design with the Fitter, and use the TimeQuest timing analyzer to ensure your design meets the timing requirements. With the PowerPlay Power Analyzer, you ensure the design’s power consumption is within limits.

Initial Compilation: Required Settings

This section describes the basic assignments and settings for your initial compilation. Check the following settings before compiling your design in the Quartus II software. Significantly varied compilation results can occur depending on the assignments that you set.

Device Settings

Device assignments determine the timing model that the Quartus II software uses during compilation. Choose the correct speed grade to obtain accurate results and the best optimization. The device size and the package determine the device pin-out and the available resources in the device.

Device Migration Settings

If you anticipate a change to the target device later in the design cycle, either because of changes in your design or other considerations, plan for the change at the beginning of your design cycle. Whenever you select a target device, you can also list any other compatible devices you can migrate by clicking on the Migration Devices button in the Device dialog box.

Selecting the migration device and companion device early in the design cycle helps to minimize changes to your design at a later stage.
I/O Assignments

The I/O standards and drive strengths specified for a design affect I/O timing. Specify I/O assignments so that the Quartus II software uses accurate I/O timing delays in timing analysis and Fitter optimizations.

If there is no PCB layout requirement, then you do not need to specify pin locations. If your pin locations are not fixed due to PCB layout requirements, then leave the pin locations unconstrained. If your pin locations are already fixed, then make pin assignments to constrain the compilation appropriately.

For more information about recommendations for making pin assignments that can have a large effect on your results in smaller macrocell-based architectures, refer to Optimizing Resource Utilization (Macrocell-Based CPLDs) in the Timing Closure and Optimization chapter in volume 2 of the Quartus II Handbook.

Use the Assignment Editor and Pin Planner to assign I/O standards and pin locations.

For more information about I/O standards and pin constraints, refer to the appropriate device handbook. For more information about planning and checking I/O assignments, refer to the I/O Management chapter in volume 2 of the Quartus II Handbook.

For information about using the Assignment Editor, refer to About the Assignment Editor in Quartus II Help.

Timing Requirement Settings

You must use comprehensive timing requirement settings to 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 your design and make trade-offs for performance. This optimization can also save area or power utilization in non-critical parts of your design.
- If enabled, the Quartus II software performs physical synthesis optimizations based on timing requirements.
- Depending on the Fitter Effort setting, the Fitter can reduce runtime if your design meets the timing requirements.

For more information about optimization with physical synthesis, refer to Physical Synthesis Optimization in the Timing Closure and Optimization chapter in volume 2 of the Quartus II Handbook.

For more information about reducing runtime by changing Fitter effort, refer to Fitter Settings Page in the Quartus II Help.

Use your real requirements to get the best results. If you apply more demanding timing requirements than you need, then increased resource usage, higher power utilization, increased compilation time, or all of these may result.
The Quartus II TimeQuest Timing Analyzer determines if the design implementation meets the timing requirement. The Compilation Report shows whether your design meets the timing requirements, while the timing analysis reporting commands provide detailed information about the timing paths.

To create timing constraints for the TimeQuest analyzer, create a Synopsys Design Constraints File (.sdc). You can also enter constraints in the TimeQuest GUI. Use the `write_sdc` command, or the Constraints menu in the TimeQuest analyzer. Click **Write SDC File** to write your constraints to an .sdc. You can add an .sdc to your project on the **Quartus II Settings** page under **Timing Analysis Settings**.

If you already have an .sdc in your project, using the `write_sdc` command from the command line or using the **Write SDC File** option from the TimeQuest GUI allows you to create a new .sdc that combines the constraints from your current .sdc and any new constraints added through the GUI or command window, or overwrites the existing .sdc with your newly applied constraints.

Ensure that every clock signal has an accurate clock setting constraint. If clocks arrive from a common oscillator, then they are related. Ensure that you set up all related or derived clocks in the constraints correctly. You must constrain all I/O pins that require I/O timing optimization. Specify both minimum and maximum timing constraints as applicable. If your design contains more than one clock or contains pins with different I/O requirements, make multiple clock settings and individual I/O assignments instead of using a global constraint.

Make any complex timing assignments required in your design, including false path and multicycle path assignments. Common situations for these types of assignments include reset or static control signals (when the time required for a signal to reach a destination is not important) or paths that have more than one clock cycle available for operation in a design. These assignments enable the Quartus II software to make appropriate trade-offs between timing paths and can enable the Compiler to improve timing performance in other parts of your design.

For more information about timing assignments and timing analysis, refer to the **Quartus II TimeQuest Timing Analyzer** chapter in volume 3 of the **Quartus II Handbook** and the **Quartus II TimeQuest Timing Analyzer Cookbook**.

To ensure that you apply constraints or assignments to all design nodes, you can report all unconstrained paths in your design with the **Report Unconstrained Paths** command in the Task pane of the Quartus II TimeQuest Timing Analyzer or the `report_ucp` Tcl command.

### Partitions and Floorplan Assignments for Incremental Compilation

The Quartus II incremental compilation feature enables hierarchical and team-based design flows in which you can compile parts of your design while other parts of your design remain unchanged and import parts of your design from separate Quartus II projects.
Using incremental compilation for your design with good design partitioning methodology helps to achieve timing closure. Creating design partitions on some of the major blocks in your design and assigning them to LogicLock™ regions, reduces Fitter time and improves the quality and repeatability of the results. LogicLock regions are flexible, reusable floorplan location constraints that help you place logic on the target device. When you assign entity instances or nodes to a LogicLock region, you direct the Fitter to place those entity instances or nodes inside the region during fitting.

For more information about LogicLock regions, refer to About LogicLock Regions in Quartus II Help.

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

For more information, refer to the “Incremental Compilation” section in the Reducing Compilation Time chapter in volume 2 of the Quartus II Handbook.

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

For more information about guidelines to create partition and floorplan assignments for your design, refer to the Best Practices for Incremental Compilation Partitions and Floorplan Assignments chapter in volume 1 of the Quartus II Handbook.

**Physical Implementation**

Most optimization issues involve preserving previous results, reducing area, reducing critical path delay, reducing power consumption, and reducing runtime. The Quartus II software includes advisors to address each of these issues and helps you optimize your design. Run these advisors during physical implementation for advice about your specific design.

You can reduce the time spent on design iterations by following the recommended design practices for designing with Altera® devices. Design planning is critical for successful design timing implementation and closure.

For more information, refer to the Design Planning with the Quartus II Software chapter in volume 1 of the Quartus II Handbook.

**Trade-Offs and Limitations**

Many optimization goals can conflict with one another, so you might need to resolve conflicting goals. For example, one major trade-off during physical implementation is between resource usage and critical path timing, because certain techniques (such as logic duplication) can improve timing performance at the cost of increased area. Similarly, a change in power requirements can result in area and timing trade-offs, such as if you reduce the number of available high-speed tiles, or if you attempt to shorten high-power nets at the expense of critical path nets.
In addition, system cost and time-to-market considerations can affect the choice of device. For example, a device with a higher speed grade or more clock networks can facilitate timing closure at the expense of higher power consumption and system cost. Finally, not all designs can be realized in a hardware circuit with limited resources and given constraints. If you encounter resource limitations, timing constraints, or power constraints that cannot be resolved by the Fitter, consider rewriting parts of the HDL code.

For more information, refer to the *Timing Closure and Optimization* and *Area Optimization* chapters in volume 2 of the *Quartus II Handbook*.

**Preserving Results and Enabling Teamwork**

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

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

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

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

For more information, refer to *Quartus II Incremental Compilation for Hierarchical and Team-Based Designs* in volume 1 of the *Quartus II Handbook* and *About Incremental Compilation* in Quartus II Help.

**Reducing Area**

By default, the Quartus II 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.

For more information, refer to the *Netlist Optimizations and Physical Synthesis, Timing Closure and Optimization*, and *Area Optimization* chapters in volume 2 and the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*. 
Reducing Critical Path Delay

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

By default, the Quartus II Fitter tries to meet the specified timing requirements and stops trying when the requirements are met. Therefore, using realistic constraints is important to successfully close timing. If you under-constrain your design, you may get sub-optimal results. By contrast, if you over-constrain your design, the Fitter might over-optimize non-critical paths at the expense of true critical paths. In addition, you might incur an increased area penalty. Compilation time may also increase because of excessively tight constraints.

If your resource usage is very high, the Quartus II Fitter might have trouble finding a legal placement. In such circumstances, the Fitter automatically modifies some of its settings to try to trade off performance for area.

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

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

In high-density FPGAs, routing accounts for a major part of critical path timing. Because of this, duplicating or retiming logic can allow the Fitter to reduce delay on critical paths. The Quartus II 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 your HDL to manually duplicate or adjust the timing logic.

Reducing Power Consumption

The Quartus II software has features that help reduce design power consumption. The PowerPlay power optimization options control the power-driven compilation settings for Synthesis and the Fitter.

For more information, refer to the Power Optimization chapter in volume 2 of the Quartus II Handbook.

Reducing Runtime

Many Fitter settings influence compilation time. Most of the default settings in the Quartus II software are set for reduced compilation time. You can modify these settings based on your project requirements.
The Quartus II software supports parallel compilation in computers with multiple processors. This can reduce compilation times by up to 15% while giving the identical result as serial compilation.

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

Using Quartus II Tools

The following sections describe several Quartus II tools that you can use to help optimize your design.

Design Analysis

The Quartus II software provides tools that help with a visual representation of your design. You can use the RTL Viewer to see a schematic representation of your design before synthesis and place-and-route. The Technology Map Viewer provides a schematic representation of the design implementation in the selected device architecture after synthesis and place-and-route. It can also include timing information.

With incremental compilation, the Design Partition Planner and the Chip Planner allow you to partition and layout your design at a higher level. In addition, you can perform many different tasks with the Chip Planner, including: making floorplan assignments, implementing engineering change orders (ECOs), and performing power analysis. Also, you can analyze your design and achieve a faster timing closure with the Chip Planner. The Chip Planner provides physical timing estimates, critical path display, and a routing congestion view to help guide placement for optimal performance.

For more information, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Designs and Best Practices for Incremental Compilation Partitions and Floorplan Assignments chapters in volume 1 and the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

Advisors

The Quartus II software includes several advisors to help you optimize your design and reduce compilation time. You can complete your design faster by following the recommendations in the Compilation Time Advisor, Incremental Compilation Advisor, Timing Optimization Advisor, Area Optimization Advisor, Resource Optimization Advisor, and Power Optimization Advisor. These advisors give recommendations based on your project settings and your design constraints.

For more information about advisors, refer to Running Advisors in the Quartus II Software in Quartus II Help.
Design Space Explorer

Use the Design Space Explorer (DSE) to find optimal settings in the Quartus II software. DSE automatically tries different combinations of netlist optimizations and advanced Quartus II software compiler settings, and reports the best settings for your design, based on your chosen primary optimization goal. You can try different seeds with the DSE if you are fairly close to meeting your timing or area requirements and find one seed that meets timing or area requirements. Finally, the DSE can run the different compilations on multiple computers in parallel, which shortens the timing closure process.

For more information, refer to About Design Space Explorer in Quartus II Help.

Conclusion

The Quartus II software includes a number of features and tools that you can use to optimize area, timing, power, and compilation time when you design for programmable logic devices (PLDs).

Document Revision History

Table 10–1 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2013</td>
<td>13.1.0</td>
<td>Minor changes for HardCopy.</td>
</tr>
<tr>
<td>May 2013</td>
<td>13.0.0</td>
<td>Added the information about initial compilation requirements. This section was moved from the Area Optimization chapter of the Quartus II Handbook. Minor updates to delineate division of Timing and Area optimization chapters.</td>
</tr>
<tr>
<td>June 2012</td>
<td>12.0.0</td>
<td>Removed survey link.</td>
</tr>
<tr>
<td>November 2011</td>
<td>10.0.3</td>
<td>Template update.</td>
</tr>
<tr>
<td>December 2010</td>
<td>10.0.2</td>
<td>Changed to new document template. No change to content.</td>
</tr>
<tr>
<td>August 2010</td>
<td>10.0.1</td>
<td>Corrected link.</td>
</tr>
<tr>
<td>July 2010</td>
<td>10.0.0</td>
<td>Initial release. Chapter based on topics and text in Section III of volume 2.</td>
</tr>
</tbody>
</table>

For previous versions of the Quartus II Handbook, refer to the Quartus II Handbook Archive.