Introduction

Physical implementation can be an intimidating and challenging phase of the design process. This section 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. 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 static 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. If your design does not meet all of your constraints, reiterate this process either partially or completely (based on the specific situation). Refer to “Further Reading” for more information on any particular feature.

Physical Implementation

Most optimization issues are about 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 situation.

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.

Trade Offs and Limitations

Many optimization goals can conflict with one another, so you might be required to make trade offs between different 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. For example, if you reduce the number of high-speed tiles available, 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 the 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 limitation, timing constraints, or power constraints that cannot be resolved by the Fitter, you might have to consider rewriting parts of the HDL code.
Preserving Results and Enabling Teamwork

Some of the Quartus II Fitter algorithms are pseudo-random in nature, which means that 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 have trouble meeting timing on a portion of your design, you can partition the troublesome 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 of the optimization techniques often result in longer compilation times, but by applying them only on specific partitions, you can reduce this impact and complete more iterations per day.

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

Reducing Area

By default, the Quartus II Fitter might spread out a design 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 more area savings, you can enable certain physical synthesis options to modify your netlist to create more area-efficient implementation, but at the cost of increased runtime and decreased performance.

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 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 are likely to 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 might increase because of excessively tight constraints.

If your resource use 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 in improving 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 the results for each 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 shorten 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. Alternately, you can modify your HDL to manually duplicate or retime logic.

**Reduce Power Consumption**

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

**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**

**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 behavioral simulation, 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 routing congestion view to help guide placement for optimal performance.

**Advisors**

The Quartus II software includes several advisors to help you optimize your design. You can save time by following the recommendations in the timing optimization advisor, the area optimization advisor, and the power optimization advisor. These advisors give recommendations based on your project settings and your design constraints.

**Design Space Explorer**

Use the Design Space Explorer (DSE) to find optimum 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. You can try different seeds with the DSE if you are fairly close to meeting timing requirements. Finally, the DSE can run the different compilations on multiple computers at once, which shortens the timing closure process.

**Further Reading**

This section includes the following chapters:

- Chapter 10, Area and Timing Optimization
- Chapter 11, Power Optimization
- Chapter 12, Analyzing and Optimizing the Design Floorplan
- Chapter 13, Netlist Optimizations and Physical Synthesis
- Chapter 14, Design Space Explorer

Other supporting documents in volume 1 of the Quartus II Handbook are:

- Design Planning with the Quartus II Software
- Quartus II Incremental Compilation for Hierarchical and Team-Based Designs
- Design Recommendations for Altera Devices and the Quartus II Design Assistant
- Recommended HDL Coding Styles
- Section IV. Engineering Change Management

Other documents of interest:

- AN 584: Timing Closure Methodology for Advanced FPGA Designs
This chapter describes techniques to reduce resource usage, improve timing performance, and reduce compilation times when designing for Altera® devices.

Introduction

Good optimization techniques are essential for achieving the best results when designing for programmable logic devices (PLDs). The optimization features available in the Quartus® II software allow you to meet design requirements by applying these techniques at multiple points in the design process.

This chapter explains how and when to use some of the features described in other chapters of the Quartus II Handbook. This introduction describes the various stages in a design optimization process, and points you to the appropriate sections in the chapter for area, timing, or compilation time optimization.

Topics in this chapter include:

- “Initial Compilation: Required Settings” on page 10–3
- “Initial Compilation: Optional Settings” on page 10–6
- “Design Analysis” on page 10–11
- “Resource Utilization Optimization Techniques (LUT-Based Devices)” on page 10–19
- “Timing Optimization Techniques (LUT-Based Devices)” on page 10–32
- “Resource Utilization Optimization Techniques (Macrocell-Based CPLDs)” on page 10–54
- “Timing Optimization Techniques (Macrocell-Based CPLDs)” on page 10–60
- “Compilation-Time Optimization Techniques” on page 10–64
- “Other Optimization Resources” on page 10–70
- “Scripting Support” on page 10–71

The application of these techniques varies from design to design. Applying each technique does not always improve results. Settings and options in the Quartus II software have default values that generally 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.

When using advanced optimization settings and tools, it is important to benchmark their effect on your results and to use them only if they improve results for your design.

You can use the optimization flow described in this chapter to explore various compiler settings and determine the techniques that provide the best results.
Optimizing Your Design

The first stage in the optimization process is to perform an initial compilation on your design. “Initial Compilation: Required Settings” on page 10–3 provides guidelines for some of the settings and assignments that are recommended for your initial compilation. “Initial Compilation: Optional Settings” on page 10–6 describes settings that you might turn on based on your design requirements. “Design Analysis” on page 10–11 explains how to analyze the compilation results.

You can use incremental compilation in the optimization process. Incremental compilation can preserve timing to aid in timing closure, as well as compilation time reduction; however, it can cause a slight increase in resource utilization.

For more details about Quartus II incremental compilation flow, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook.

After you have analyzed the compilation results, perform the optimization stages in the recommended order, as described in this chapter.

For LUT-based devices (FPGAs, MAX® II series of devices), perform optimizations in the following order:

1. If your design does not fit, refer to “Resource Utilization Optimization Techniques (LUT-Based Devices)” on page 10–19 before trying to optimize I/O timing or register-to-register timing.

2. If your design does not meet the required I/O timing performance, refer to “I/O Timing Optimization Techniques (LUT-Based Devices)” on page 10–73 before trying to optimize register-to-register timing.

3. If your design does not meet the required slack on any of the clock domains in the design, refer to “Register-to-Register Timing Optimization Techniques (LUT-Based Devices)” on page 10–73.

For macrocell-based devices (MAX 7000 and MAX 3000 CPLDs), perform optimizations in the following order:

1. If your design does not fit, refer to “Resource Utilization Optimization Techniques (Macrocell-Based CPLDs)” on page 10–54 before trying to optimize I/O timing or register-to-register timing.

2. If your timing performance requirements are not met, refer to “Timing Optimization Techniques (Macrocell-Based CPLDs)” on page 10–60.

3. For device-independent techniques to reduce compilation time, refer to “Compilation-Time Optimization Techniques” on page 10–64.

You can use all these techniques in the GUI or with Tcl commands. For more information about scripting techniques, refer to “Scripting Support” on page 10–71.
Initial Compilation: Required Settings

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

You should verify the following settings:

- “Device Settings”
- “I/O Assignments”
- “Timing Requirement Settings” on page 10–4
- “Device Migration Settings” on page 10–5
- “Partitions and Floorplan Assignments for Incremental Compilation” on page 10–5

Device Settings

Specific 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 number of resources available in the device.

To select the target device, on the Assignments menu, click Device.

In a Tcl script, use the following command to set the device:

```tcl
set_global_assignment -name DEVICE <device>
```

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.

The Quartus II software can select pin locations automatically. If your pin locations are not fixed due to PCB layout requirements, leave pin locations unconstrained. If your pin locations are already fixed, make pin assignments to constrain the compilation appropriately. “Resource Utilization Optimization Techniques (Macrocell-Based CPLDs)” on page 10–54 includes recommendations for making pin assignments that can have a large effect on your results in smaller macrocell-based architectures.

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 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 the Assignment Editor chapter in volume 2 of the Quartus II Handbook.
Timing Requirement Settings

Using comprehensive timing requirement settings is an important step for achieving the best results for the following reasons:

- Correct timing assignments allow 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.

- The Quartus II software performs physical synthesis optimizations based on timing requirements (refer to “Physical Synthesis Optimizations” on page 10–40 for more information).

- Depending on the Fitter Effort setting, the Quartus II Fitter can reduce runtime considerably if your timing requirements are being met. For a description of the different effort levels, refer to “Fitter Effort Setting” on page 10–10

Use your real requirements to get the best results. If you apply more demanding timing requirements than you actually need, increased resource usage, higher power utilization, increased compilation time, or all of these may result.

The TimeQuest Timing Analyzer checks your design against the timing constraints. The Compilation Report and timing analysis reporting commands show whether timing requirements are met and provide detailed timing information about paths that violate timing requirements.

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

If you already have an .sdc file in your project, using the write_sdc command from the command line or using the Write SDC File option from the TimeQuest GUI overwrites the existing file with your newly applied constraints.

If you are using the Quartus II Classic Timing Analyzer, refer to the Quartus II Help topic “Classic Timing Analyzer Settings Page (Settings Dialog Box)”. For some older Altera device families, you can create clock and other timing constraints using the Classic Timing Analyzer. For details about how to create these constraints, refer to the Quartus II Help topic “Specifying Timing Requirements and Options (Classic Timing Analyzer)”.

Ensure that every clock signal has an accurate clock setting constraint. If clocks come from a common oscillator, they can be considered related. Ensure that all related or derived clocks are set up correctly in the constraints. All I/O pins that require I/O timing optimization must be constrained. You should also specify minimum timing constraints as applicable. If there is more than one clock or there are different I/O requirements for different pins, make multiple clock settings and individual I/O assignments instead of using a global constraint.
Make any complex timing assignments required in the design, including false path and multicycle path assignments. Common situations for these types of assignments include reset or static control signals, cases in which it is not important how long it takes a signal to reach a destination, and paths that can operate in more than one clock cycle. These assignments allow 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 the design.

For more information about timing assignments and timing analysis, refer to *The Quartus II TimeQuest Timing Analyzer* and the *Quartus II Classic Timing Analyzer* chapters in volume 3 of the *Quartus II Handbook* and the *Quartus II TimeQuest Timing Analyzer Cookbook*. For more information about how to specify multicycle exceptions in the TimeQuest Timing Analyzer, refer to *AN 481: Applying Multicycle Exceptions in the TimeQuest Timing Analyzer*.

**Timing Constraint Check—Report Unconstrained Paths**

To ensure that all constraints or assignments have been applied to design nodes, you can report all unconstrained paths in your design.

While using the Quartus II TimeQuest Timing Analyzer, you can report all the unconstrained paths in your design with the *Report Unconstrained Paths* command in the Task pane or the `report_ucp` Tcl command.

**Device Migration Settings**

If you anticipate a change to the target device later in the design cycle, either because of changes in the design or other considerations, plan for it at the beginning of your design cycle. Whenever you select a target device in the Settings dialog box, you can also list any other compatible devices you can migrate to by clicking on the Migration Devices button on the Device page. If you plan to move your design to a HardCopy® device, make sure to select the device from the list under the Companion device tab on the Device page.

By selecting the migration device and companion device early in the design cycle, you help to minimize changes to the design at a later stage.

**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 the design remain unchanged, or import parts of your design from separate Quartus II projects.

Using incremental compilation for your design with good design partitioning methodology can often help to achieve timing closure. Creating LogicLock™ regions and using incremental compilation can help you achieve timing closure block by block, and preserve the timing performance between iterations, which helps achieve timing closure for the entire design.

Using incremental compilation may also help reduce compilation times. For more information, refer to “Incremental Compilation” on page 10–64.
If you want to take advantage of incremental compilation for a team-based design flow to reduce your compilation times, or to improve the timing performance of your design during iterative compilation runs, make meaningful design partitions and create a floorplan for your design partitions. Good assignments can improve your results. Assignments can negatively affect a design’s results if you do not follow Altera’s recommendations.

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

For guidelines about how 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.

Initial Compilation: Optional Settings

This section describes optional settings that can help to compile your design. You can selectively set all the optional settings that help to improve performance (if required) and reduce compilation time. These settings vary between designs and there is no standard set that applies to all designs. Significantly different compilation results can occur depending on the assignments you have set.

The following settings are optional:

- “Design Assistant”
- “Smart Compilation Setting” on page 10-7
- “Early Timing Estimation” on page 10-7
- “Optimize Hold Timing” on page 10-8
- “Asynchronous Control Signal Recovery/Removal Analysis” on page 10-8
- “Limit to One Fitting Attempt” on page 10-9

Design Assistant

You can run the Design Assistant to analyze the post-fitting results of your design during a full compilation. The Design Assistant checks rules related to gated clocks, reset signals, asynchronous design practices, and signal race conditions. This is especially useful during the early stages of your design, so that you can work on any areas of concern in your design before proceeding with design optimization.

On the Assignments menu, click Settings. In the Category list, select Design Assistant and turn on Run Design Assistant during compilation.

You can also specify which rules you want the Design Assistant to apply when analyzing and generating messages for a design.

For more information about the rules in the Design Assistant, refer to the Design Recommendations for Altera Devices and the Quartus II Design Assistant chapter in volume 1 of the Quartus II Handbook.
Smart Compilation Setting

Smart compilation can reduce compilation time by skipping compiler stages that are not required to recompile the design. This is especially useful when you perform multiple compilation iterations during the optimization phase of the design process. However, smart compilation uses more disk space. To turn on smart compilation, on the Assignments menu, click Settings. In the Category list, select Compilation Process Settings and turn on Use smart compilation.

Smart compilation skips entire compiler stages (such as Analysis and Synthesis) when they are not required. This feature is different from incremental compilation, which you can use to compile parts of your design while preserving results for unchanged parts. For information about using the incremental compilation feature to reduce your compilation time, refer to “Incremental Compilation” on page 10–64.

Early Timing Estimation

The Quartus II software provides an Early Timing Estimation feature that estimates your design’s timing results before the software performs full placement and routing. On the Processing menu, point to Start, and click Start Early Timing Estimate to generate initial compilation results after you have run analysis and synthesis. When you want a quick estimate of a design’s performance before proceeding with further design or synthesis tasks, this command can save significant compilation time. Using this feature provides a timing estimate up to 45× faster than running a full compilation, although the fit is not fully optimized or routed. Therefore, the timing report is only an estimate. On average, the estimated delays are within 11% of the final timing results as achieved by a full compilation.

You can specify the type of delay estimates to use with Early Timing Estimation. On the Assignments menu, click Settings. In the Category list, select Compilation Process Settings, and select Early Timing Estimate. On the Early Timing Estimate page, the following options are available:

- The Realistic option, which is the default, generates delay estimates that are likely to be close to the results of a full compilation.
- The Optimistic option uses delay estimates that are lower than those likely to be achieved by a full compilation, which results in an optimistic performance estimate.
- The Pessimistic option uses delay estimates that are higher than those likely to be achieved by a full compilation, which results in a pessimistic performance estimate.

All three options offer the same reduction in compilation time.

You can view the critical paths in the design by locating these paths in the Chip Planner. Then, if necessary, you can add or modify floorplan constraints such as LogicLock regions, or make other changes to the design. You can then rerun the Early Timing Estimator to quickly assess the impact of any floorplan assignments or logic changes, enabling you to try different design variations and find the best solution.
Optimize Hold Timing

The **Optimize Hold Timing** option directs the Quartus II software to optimize minimum delay timing constraints. This option is available for all Altera device families except MAX 3000 and MAX 7000 series devices. By default, the Quartus II software optimizes hold timing for all paths for designs using newer devices such as Arria II GX, Arria GX, Stratix III, Stratix IV, and Cyclone III devices. By default, the Quartus II software optimizes hold timing only for I/O paths and minimum TPD paths for older devices.

When you turn on **Optimize Hold Timing**, the Quartus II software adds delay to paths to guarantee that the minimum delay requirements are satisfied. In the Fitter Settings pane, if you select **I/O Paths and Minimum TPD Paths** (the default choice for older devices such as Cyclone II and Stratix II family of devices if you turn on **Optimize Hold Timing**), the Fitter works to meet the following criteria:

- Hold times ($t_{hi}$) from device input pins to 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 in Figure 10–1, where a derived clock generated with logic causes a hold time problem on another register. However, if your design has internal hold time violations between registers, Altera recommends that you correct the problems by making changes to your design, such as using a clock enable signal instead of a derived or gated clock.

![Figure 10–1. Optimize Hold Timing Option Fixing an Internal Hold Time Violation](image)

For design practices that can help eliminate internal hold time violations, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*.

Asynchronous Control Signal Recovery/Removal Analysis

The asynchronous control signal Recovery/Removal analysis option checks paths that end at an asynchronous clear, preset, or load of a register to determine if recovery and removal times are met for all registers. Recovery and removal times are similar to the setup and hold time requirements, respectively, but they are applicable to the control signals rather than the data. Recovery time is the minimum length of time an asynchronous control signal such as a reset must be stable before the active clock edge. Removal time is the minimum time an asynchronous control signal must be stable after the active clock edge.
When you use the TimeQuest Timing Analyzer for timing analysis, Recovery/Removal analysis and optimization are always performed during placement and routing. You can use the `create_timing_summary` Tcl command to report the recovery and removal analysis. The slack for Removal/Recovery analysis is determined in a similar way to setup and hold checks. Running the asynchronous control signal Recovery/Removal analysis helps you make sure that there are no timing failures related to the asynchronous controls in your design.

For more details about Recovery/Removal analysis with the TimeQuest Timing Analyzer, refer to *The Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

When using the Quartus II Classic Timing Analyzer for timing analysis, Recovery/Removal analysis is turned off by default. To turn on this option, on the Assignments menu, click Settings. In the Category list, select Timing Requirements & Options, then click More Settings. Turn on Enable Recovery/Removal analysis. Turning on this option adds additional constraints during placement and routing, which can increase compilation time.

For designs containing FIFOs, Altera recommends turning on Recovery/Removal analysis if you are using the Quartus II Classic Timing Analyzer.

### Limit to One Fitting Attempt

A design might fail to fit for several reasons, such as logic overuse or illegal assignments. For most failures, the Quartus II software informs you of the problem. However, if the design uses too much routing, the Quartus II software makes up to two additional attempts to fit your design. Each of these fit attempts takes significantly longer than the previous attempt.

For large designs, you might not want to wait for all three fitting attempts to be completed. To have the Quartus II software issue an error message after the first failed attempt, turn on Limit to one fitting attempt on the Fitter Settings page.

Refer to “Routing” on page 10–28 for instructions about how to lower the design’s routing utilization, so your design can be made to fit into the target device if it fails to fit due to the lack of routing resources.

### Optimize Multi-Corner Timing

Historically, FPGA timing analysis has been performed using only worst-case delays, which are described in the slow corner timing model. However, due to process variation and changes in the operating conditions, delays on some paths can be significantly smaller than those in the slow corner timing model. This can result in hold time violations on those paths, and in rare cases, additional setup time violations.
Also, because of the small process geometries of the Cyclone III, Cyclone IV, Stratix III, and Stratix IV device families, the slowest circuit performance of designs targeting these devices does not necessarily occur at the highest operating temperature. The temperature at which the circuit is slowest depends on the selected device, the design, and the Quartus II compilation results. Therefore, the Quartus II software provides the Cyclone III series, Cyclone IV, Stratix III, and Stratix IV device families with three different timing corners in commercial devices—Slow 85°C corner, Slow 0°C corner, and Fast 0°C corner. For other device families two timing corners are available in commercial devices—Fast 0°C and Slow 85°C corner.

By default, the Fitter optimizes constraints using only the slow corner timing model. You can turn on the Optimize multi-corner timing option to instruct the Fitter to also optimize constraints considering all timing corners, at the cost of a slight increase in runtime. By optimizing for all process corners, you can create a design implementation that is more robust across process, temperature, and voltage variations. This option is available only for Arria GX, Stratix, Cyclone, and MAX II series of devices.

To turn on the Optimize multi-corner timing option, on the Assignments menu, click Settings. In the Category list, select Fitter Settings and turn on Optimize multi-corner timing. Using the different timing models can be important to account for process, voltage, and temperature variations for each device. Turning this option on increases compilation time by approximately 10%.

For designs with external memory interfaces such as DDR and QDR, Altera recommends that you turn on the Optimize multi-corner timing setting.

**Fitter Effort Setting**

Fitter effort refers to the amount of effort the Quartus II software uses to fit your design. To set the Fitter effort, on the Assignments menu, click Settings. In the Category list, select Fitter Settings. The Fitter effort settings are Auto Fit, Standard Fit, and Fast Fit. The default setting depends on the device family specified.

**Auto Fit**

The Auto Fit option (available for Arria GX, Stratix, Cyclone, HardCopy, and MAX II series of devices) focuses the full Fitter effort only on those aspects of the design that require further optimization. Auto Fit can significantly reduce compilation time relative to Standard Fit if your design has easy-to-meet timing requirements, low routing resource utilization, or both. However, those designs that require full optimization generally receive the same effort as is achieved by selecting Standard Fit. Auto Fit is the default Fitter effort setting for all devices for which this option is available.

If you want the Fitter to attempt to exceed the timing requirements by a certain margin instead of simply meeting them, specify a minimum slack in the Desired worst case slack box.

Specifying a minimum slack does not guarantee that the Fitter achieves the slack requirement; it only guarantees that the Fitter applies full optimization unless the target slack is exceeded.
In some designs with multiple clocks, it might be possible to improve the timing performance on one clock domain while reducing the performance on other clock domains by over-constraining the most important clock. If you use this technique, perform a sweep over multiple seeds to ensure that any performance improvements that you see are real gains. For more information, refer to “Fitter Seed” on page 10–46.

Over-constraining the clock for which you require maximum slack, while using the Auto Fit option, increases the chances that the Fitter is able to meet this requirement. The Auto Fit option also causes the Quartus II Fitter to optimize for shorter compilation times instead of maximum possible performance if the design includes no timing assignments.

If your design has aggressive timing requirements or is hard to route, the placement does not stop early and the compilation time is the same as using the Standard Fit option.

The Auto Fit option might increase the number of routing wires used. This can lead to an increase in the dynamic power when compared to using the Standard Fit option, unless the Extra effort option in the PowerPlay power optimization list is also enabled. When you turn on Extra effort, Auto Fit continues to optimize for reduction of wire usage even after meeting the register-to-register requirement. There is no adverse effect on the dynamic power consumption. If dynamic power consumption is a concern, select Extra effort in both the Analysis & Synthesis Settings and the Fitter Settings pages.

For more details, refer to the “Power Driven Compilation” section in the Power Optimization chapter in volume 2 of the Quartus II Handbook.

Standard Fit

Use the Standard Fit option to exceed specified timing requirements and achieve the best possible timing results and lowest routing resource utilization for your design. The Standard Fit setting usually increases compilation time relative to Auto Fit, because it applies full optimization, regardless of the design requirement. In designs with no timing assignments, on average, using the Standard Fit option results in a $f_{\text{MAX}}$ about 10% higher than that achieved using the Auto Fit option. In designs where timing requirements can be easily met, using the Standard Fit option can result in considerably longer compilation times than using the Auto Fit option.

Fast Fit

The Fast Fit option reduces the amount of optimization effort for each algorithm employed during fitting. This option reduces the compilation time by about 50%, resulting in a fit that has, on average, 10% lower $f_{\text{MAX}}$ than that achieved using the Standard Fit setting.

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 Quartus II software. After design analysis, proceed to optimization, as described in “Optimizing Your Design” on page 10–2.
Error and Warning Messages

After first compiling the design, it is important to evaluate all error and warning messages to see if any design or setting changes are required. If changes are required, make these changes and recompile the design before proceeding with design optimization.

To suppress messages that you have already evaluated and do not want to see again, right-click on the message in the Messages window and click *Suppress*.

For more information about message suppression, refer to the “Message Suppression” section in the *Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*.

Ignored Timing Constraints

The Quartus II software ignores illegal, obsolete, and conflicting constraints.

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

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

If any constraints were ignored, analyze why they were ignored. If necessary, correct the constraints and recompile the design before proceeding with design optimization.

For more information about the *report_sdc* command and its options, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

If you are using the Classic Timing Analyzer, open the *Ignored Timing Assignments* page in the Compilation Report to view any constraints that were ignored.

Resource Utilization

Determining device utilization is important regardless of whether a successful fit is achieved. If your compilation results in a no-fit error, resource utilization information is important for analyzing the fitting problems in your design. If your fitting is successful, review the resource utilization information to determine whether the future addition of extra logic or other design changes might introduce fitting difficulties.

To determine resource usage, refer to the *Flow Summary* section of the Compilation Report. This section reports how many resources are used, including pins, memory bits, digital signal processing (DSP) block 9-bit elements (for Arria GX, Stratix, and Stratix II devices) or 18-bit elements (for Arria II GX, Stratix IV, and Stratix III devices), and phase-locked loops (PLLs). The *Flow Summary* indicates whether the design exceeds the available device resources. More detailed information is available by viewing the reports under *Resource Section* in the *Fitter* section of the Compilation Report.
For Arria II GX, Arria GX, Stratix IV, Stratix III, and Stratix II devices, a device with low utilization does not have the lowest adaptive logic module (ALM) utilization possible. For these devices, the Fitter uses adaptive look-up tables (ALUTs) in different ALMs—even when the logic can be placed within one ALM—to achieve the best timing and routing results. In achieving these results, the Fitter can spread logic throughout the device. As the device fills up, the Fitter automatically searches for logic functions with common inputs to place in one ALM. The number of partnered ALUTs and packed registers also increases. Therefore, a design that is reported as close to 100% full might still have space for extra logic if logic and registers can be packed together more aggressively.

If resource usage is reported as less than 100% and a successful fit cannot be achieved, either there are not enough routing resources or some assignments are illegal. In either case, a message appears in the Processing tab of the Messages window describing the problem.

If the Fitter finishes faster than the Fitter runs on similar designs, a resource might be over-utilized or there might be an illegal assignment. If the Quartus II software seems to run for an excessively long time compared to runs on similar designs, a legal placement or route probably cannot be found. In the Compilation Report, look for errors and warnings that indicate these types of problems.

Refer to “Limit to One Fitting Attempt” on page 10–9 for more information about how to get a quick error message on hard-to-fit designs.

You can use the Chip Planner or the Timing Closure Floorplan (for supported devices) to find areas of the device that have routing congestion. 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.

For details about using the Chip Planner and the Timing Closure Floorplan tools, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

I/O Timing (Including t\textsubscript{pd})

The Quartus II TimeQuest Timing Analyzer supports the Synopsys Design Constraints (SDC) format for constraining your design. When using the TimeQuest Timing Analyzer for timing analysis, use the \texttt{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 \texttt{set_output_delay} command to specify the data arrival time at an output port with respect to a given clock. You can use the \texttt{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 TimeQuest Timing Analyzer Report pane. In cases where you do not apply an explicit I/O timing constraint to an I/O pin, the Quartus II 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.
If you are using the Quartus II Classic Timing Analyzer, refer to the Quartus II Help topic “Classic Timing Analyzer and Timing I/O analysis reports”.

For more information about how timing numbers are calculated, refer to the Quartus II TimeQuest Timing Analyzer chapter or the Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook.

Register-to-Register Timing

This section contains the following sections:

- “Timing Analysis with the TimeQuest Timing Analyzer”
- “Tips for Analyzing Failing Paths” on page 10–16
- “Tips for Analyzing Failing Clock Paths that Cross Clock Domains” on page 10–17

Timing Analysis with the TimeQuest Timing Analyzer

If you are using the TimeQuest Timing Analyzer, you should analyze all valid register-to-register paths by using appropriate constraints. Use the report_timing command to generate the required timing reports for any register-to-register path. Your design meets timing requirements when you do not have negative slack on any register-to-register path on any of the clock domains.

All paths that do not meet the timing requirement are shown with a negative slack and appear in red in the TimeQuest Timing Analyzer GUI.

When you select a path listed in the TimeQuest Report Timing pane, the tabs in the corresponding path detail pane show a path summary of source and destination registers and their timing, statistics about the path delay, detailed information about the complete data path with all nodes in the path and the waveforms of the relevant signals (Figure 10–2). You can locate a selected path in the Chip Planner or the Technology Map Viewer by using the shortcut menu. Similarly, if you know that a path is not a valid path, you can set it to be a false path using the shortcut menu.

To see the path details of any selected path, click on the Data Path tab in the path details pane. This displays the details of the Data Arrival Path, as well as the Data Required Path. For a graphical view of the information, click on the Waveform tab.
Timing Analysis with the Classic Timing Analyzer

If you are using the Quartus II Classic Timing Analyzer, in the Compilation Report window, refer to the **Timing Analyzer** section to determine whether register-to-register timing requirements are met. The **Clock Setup** folder displays setup slacks between registers on each clock domain in the design. The paths that do not meet timing requirements have a negative slack and appear in red.

To determine why your timing requirements were not met, right-click on an entry in the report and click **List Paths**. A message listing the paths appears in the **System** tab of the Messages window. The expanded report for the path appears (Figure 10–3). Click the “+” icon at the beginning of the line to see where the greatest delay is located along the path.

The List Paths report shows the slack time and how that slack time was calculated. By expanding the various entries, you can see the incremental delay through each node in the path as well as the total delay. The incremental delay is the sum of the interconnect delay (IC) and the cell delay (CELL) through the logic.
To visually analyze register-to-register timing paths, right-click on a path, point to Locate, and click Locate in Chip Planner. For MAX 3000 and MAX 7000 devices, click Locate in Timing Closure Floorplan to perform this analysis. The Chip Planner or Timing Closure Floorplan appears with the path highlighted. On the View menu in the Chip Planner, click Critical Path Settings to select the paths you want to view. To turn critical paths on or off in the Chip Planner, on the View menu of the Chip Planner, click Show Critical Paths.

For more information about how timing analysis results are calculated, refer to the Quartus II TimeQuest Timing Analyzer chapter or the Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook.

You can also see the logic in a particular path by locating the logic in the RTL Viewer or Technology Map Viewer. These viewers allow you to see a gate-level or technology-mapped representation of your design netlist. To locate a timing path in one of the viewers, right-click on a path in the report, point to Locate, and click Locate in RTL Viewer or Locate in Technology Map Viewer. When you locate a timing path in the Technology Map Viewer, the annotated schematic displays the same delay information that is shown when you use the List Paths command.

For more information about netlist viewers, refer to the Analyzing Designs with Quartus II Netlist Viewers chapter in volume 1 of the Quartus II Handbook.

**Tips for Analyzing Failing Paths**

When you are analyzing clock path failures, 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 particular nodes that appear in many failing paths. Look for paths that have common source registers, destination registers, or common intermediate combinational nodes. In some cases, the registers might not be identical, but are part of the same bus. In the timing analysis report panels, clicking on the From or To column headings can be helpful to sort the paths by the source or destination registers. Clicking first on From, then on To, uses the registers in the To column as the primary sort and From as the secondary sort. If you see common nodes, these nodes indicate areas of your design that might be improved through source code changes or Quartus II 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.

**Tips for Analyzing Failing Clock Paths that Cross Clock Domains**

When analyzing clock path failures, check whether these paths cross between two clock domains. This is the case if the From Clock and To Clock in the timing analysis report are different. There can also be paths that involve a different clock in the middle of the path, even if the source and destination register clock are the same. To analyze these paths in more detail, right-click on the entry in the report and click List Paths.

Expand the List Paths entry in the Messages window and analyze the largest register-to-register requirement. Evaluate the setup relationship between the source and destination (launch edge and latch edge) to determine if that is reducing the available setup time. 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.

Check to see if the PLL phase shift is reducing the setup requirement. You might be able to adjust this using PLL parameters and settings.

If you are using the Quartus II Classic Timing Analyzer, you can direct the software to analyze the PLL compensation delay as clock skew by enabling Clock Latency analysis. On the Assignments menu, click Timing Analysis Settings. In the Category list, select Classic Timing Analyzer Settings and click More Settings. In the Name list, select Enable Clock Latency. In the Setting list, select On. Typically, you must enable this option if your design results in timing violations for paths that pass between PLL clock domains. The Quartus II TimeQuest Timing Analyzer performs this analysis by default.

Paths that cross clock domains are generally protected with synchronization logic (for example, FIFOs or double-data synchronization registers) to allow asynchronous interaction between the two clock domains. In such cases, you can ignore the timing paths between registers in the two clock domains while running timing analysis, even if the clocks are related.

The Fitter attempts to optimize all failing timing paths. If there are paths that can be ignored for optimization and timing analysis, but the paths do not have constraints that instruct the Fitter to ignore them, the Fitter tries to optimize those paths as well. In some cases, optimizing unnecessary paths can prevent the Fitter from meeting the timing requirements on timing paths that are critical to the design. It is beneficial to specify all paths that can be ignored, so that the Fitter can put more effort into the paths that must meet their timing requirements instead of optimizing paths that can be ignored.
For more details about how to ignore timing paths that cross clock domains, refer to the Quartus II TimeQuest Timing Analyzer chapter or the Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook.

Evaluate the clock skew between the source clock and the destination clock to determine if that is reducing the available setup time. You can check the shortest and longest clock path reports to see what is causing the clock skew. Avoid using combinational logic in clock paths because it contributes to clock skew. Differences in the logic or in its routing between the source and destination can cause clock skew problems and result in warnings during compilation.

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 some smaller portion, such as a quadrant. The Quartus II software attempts to assign signals to global routing resources automatically, but you might be able to make more suitable assignments manually.

Refer to the relevant device handbook for details about the number and types of global routing resources available.

Check the global signal utilization in your design to ensure that appropriate signals have been placed on global routing resources. In the Compilation Report, open the Fitter report and click the 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 clock 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. You can also use the Global Signal setting in the Assignment Editor to control global routing resources.

Compilation Time

In long compilations, most of the time is spent in the Analysis and Synthesis and Fitter modules. Analysis and Synthesis includes synthesis netlist optimizations, if you have turned on that option. The Fitter includes two steps, placement and routing, and also includes physical synthesis if you turned on that option. The Flow Elapsed Time section of the Compilation Report shows how much time is spent running the Analysis and Synthesis and Fitter modules. The Fitter Messages report in the Fitter section of the Compilation Report shows the time that was spent in placement and the time that was spent in routing.

Placement is the process of finding optimum locations for the logic in your design. Routing is the process of connecting the nets between the logic in your design. There are many possible placements for the logic in a design, and finding better placements typically uses more compilation time. Good logic placement allows you to more easily meet your timing requirements and makes the design easier to route.
The applicable messages are indicated as shown in the following example, with each time component in two-digit format:

Info: Fitter placement operations ending: elapsed time = <days:hours:mins:secs>
Info: Fitter routing operations ending: elapsed time = <days:hours:mins:secs>

Days are not shown if the time is less than one day.

While the Fitter is running (including Placement and Routing), hourly info messages similar to the following message are displayed every hour to indicate Fitter operations are progressing normally.

Info: Placement optimizations have been running for x hour(s)
In this case, x indicates the number of hours the process has run.

Resource Utilization Optimization Techniques (LUT-Based Devices)

After design analysis, the next stage of design optimization is to improve resource utilization. Complete this stage before proceeding to I/O timing optimization or register-to-register timing optimization. Ensure that you have already set the basic constraints described in “Initial Compilation: Required Settings” on page 10–3 before proceeding with the resource utilization optimizations discussed in this section. If a design does not fit into a specified device, use the techniques in this section to achieve a successful fit. After you optimize resource utilization and your design fits in the desired target device, optimize I/O timing as described in “I/O Timing Optimization Techniques (LUT-Based Devices)” on page 10–73. These tips are valid for all FPGA families and the MAX II family of CPLDs.

Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize the resource usage. To run the Resource Optimization Advisor, on the Tools menu, point to Advisors, and click Resource Optimization Advisor.

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

Resolving Resource Utilization Issues Summary

Resource utilization issues can be divided into the following three categories:

- Issues relating to I/O pin utilization or placement, including dedicated I/O blocks such as PLLs or LVDS transceivers (refer to “I/O Pin Utilization or Placement”).
- Issues relating to logic utilization or placement, including logic cells containing registers and look-up tables as well as dedicated logic, such as memory blocks and DSP blocks (refer to “Logic Utilization or Placement” on page 10–20).
- Issues relating to routing (refer to “Routing” on page 10–28).
I/O Pin Utilization or Placement

Use the suggestions in the following sections to help you resolve I/O resource problems.

Use I/O Assignment Analysis

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

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

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

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 a Quartus II 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 are available, a successful fit can often be obtained by using a larger device package (which can be the same device density) that has more available user I/O pins.

For more information about I/O assignment analysis, refer to the I/O Management chapter in volume 2 of the Quartus II Handbook.

Logic Utilization or Placement

Use the suggestions in the following subsections to help you resolve logic resource problems, including logic cells containing registers and lookup tables (LUTs), as well as dedicated logic such as memory blocks and DSP blocks.
Optimize Synthesis for Area, Not Speed

If your design fails to fit because it uses too much logic, resynthesize the design to improve the area utilization. First, ensure that you have set your device and timing constraints correctly in your 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 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, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus II integrated synthesis, select Balanced or Area for the Optimization Technique. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the Area setting (potentially at the expense of register-to-register timing performance) while leaving the default Optimization Technique setting at Balanced (for the best trade-off between area and speed for certain device families) or Speed. 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 less resource utilization.

In the Quartus II 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.

For information about setting timing requirements and synthesis options in Quartus II integrated synthesis and other synthesis tools, refer to the appropriate chapter in Section III, Synthesis in volume 1 of the Quartus II Handbook, or your synthesis software’s documentation.

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

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 Altera device.

The Quartus II software provides the Restructure Multiplexers logic option, which can extract and optimize buses of multiplexers during synthesis. This option is available on the Analysis & Synthesis Settings page of the Settings dialog box and is useful if your design contains buses of fragmented multiplexers. This option restructures multiplexers more efficiently for area, allowing the design to implement multiplexers with a reduced number of logic elements (LEs) or ALMs. Using the Restructure Multiplexers logic option can reduce your design’s register-to-register timing performance. This option is turned on automatically when you set the Quartus II Analysis & Synthesis Optimization Technique option to Area or Balanced. To change the default setting, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and click the appropriate option from the Restructure Multiplexers list to set the option globally.
For design guidelines to achieve optimal resource utilization for multiplexer designs, refer to the Recommended HDL Coding Styles chapter in volume 1 of the Quartus II Handbook. For more information about the Restructure Multiplexers option in the Quartus II software, refer to the Quartus II Integrated Synthesis chapter in volume 1 of the Quartus II Handbook.

Perform WYSIWYG Resynthesis with Balanced or Area Setting

If you use another EDA synthesis tool and want to determine if the Quartus II software can remap the circuit to use fewer LEs or ALMs, perform the following steps:

1. On the Assignments menu, click Settings.
2. In the Category list, select Analysis & Synthesis Settings. The Analysis & Synthesis Settings page appears.
3. Turn on Perform WYSIWYG primitive resynthesis (using optimization techniques specified in Analysis & Synthesis settings). Or, on the Assignments menu, click Assignment Editor, and set the Perform WYSIWYG Primitive Resynthesis logic option for a specific module in your design.
4. On the same page, select Balanced or Area under Optimization Technique. Or, on the Assignments menu, click Assignment Editor. Set the Optimization Technique to Balanced or Area for a specific module in your design.
5. Recompile the design.

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.

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. Figure 10–4 shows register packing and the gain of one logic cell in the design.

Figure 10–4. Register Packing
Registers can also be packed into DSP blocks (Figure 10–5).

**Figure 10–5.** Register Packing in DSP Blocks

The following list shows the most common cases in which register packing helps to optimize a design:

- A LUT can be implemented in the same cell as an unrelated register with a single data input
- A LUT can be implemented in the same cell as the register that is fed by the LUT
- A LUT can be implemented in the same cell as the register that feeds the LUT
- A register can be packed into a RAM block
- A register can be packed into a DSP block
- A register can be packed into an I/O Element (IOE)

The following options are available for register packing (for certain device families):

- **Off**—Does not pack registers
- **Normal**—Packs registers when this is not expected to adversely affect timing results
- **Minimize Area**—Aggressively packs registers to reduce area, even at the cost of design performance
- **Minimize Area with Chains**—Aggressively packs registers to reduce area. This option packs registers with carry chains. It also converts registers into register cascade chains and packs them with other logic to reduce area. This option is available only for Arria GX, Stratix, Cyclone, and MAX II series of devices.
- **Auto**—This is the default setting for register packing. This setting tells the Fitter to attempt to achieve the best performance while maintaining a fit for the design in the specified device. The Fitter combines all combinational (LUT) and sequential (register) functions that benefit circuit speed. In addition, more aggressive combinations of unrelated combinational and sequential functions are performed to the extent required to reduce the area of the design to achieve a fit in the specified device. This option is available only for the Arria GX, Stratix, and Cyclone series of devices.
■ **Sparse**—In this mode, the combinational (LUT) and sequential (register) functions are combined such that the combined logic has either a combinational output or a sequential output, but not both. This mode is available only for Arria II GX, Arria GX, Stratix III, Stratix II, Cyclone III, and Cyclone II devices. This option results in a higher logic array block (LAB) usage, but might give you better timing performance because of reduced routing congestion.

■ **Sparse Auto**—In this mode, the Quartus II Fitter starts with sparse mode packing, and then attempts to achieve best performance while maintaining a fit for the specified device. Later optimizations are carried out in a way similar to the **Auto** mode. This mode is available only for Arria II GX, Arria GX, Stratix IV, Stratix III, Stratix II, Cyclone III, and Cyclone II devices.

Turning on register packing decreases the number of LEs or ALMs in the design, but could also decrease performance in some cases. On the Assignments menu, click **Settings**. In the **Category** list, select **Fitter Settings**, and then click **More Settings**. Turn on **Auto Packed Registers** to turn on register packing.

The area reduction and performance results with register packing can vary greatly depending on the design.

The **Auto** setting performs more aggressive register packing as required, so the typical results vary depending on the device resource utilization.

### Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet might not fit in the targeted device. This can occur when the location or LogicLock assignments are too strict and not enough routing resources are available on the device.

In this case, use the **Routing Congestion** view in the Chip Planner to locate routing problems in the floorplan, then remove any location or LogicLock 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 LogicLock 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 LogicLock assignments in the Chip Planner, in the LogicLock 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.

For more information about the **Routing Congestion** view in the Chip Planner, refer to *Analyzing and Optimizing the Design Floorplan* in volume 2 of the *Quartus II Handbook*. Also refer to the Quartus II Help.

### Change State Machine Encoding

State machines can be encoded using various techniques. Using binary or gray code encoding typically results in fewer state registers than one-hot encoding, which requires one register for every state bit. If your design contains state machines, changing the state machine encoding to one that uses the minimal number of registers might reduce resource utilization. The effect of state machine encoding varies depending on the way your design is structured.
If your design does not manually encode the state bits, you can specify the state machine encoding in your synthesis tool. When using Quartus II integrated synthesis, on the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings** and turn on **Minimal Bits for State Machine Processing**. You can also specify this logic option for specific modules or state machines in your design with the Assignment Editor.

You can also use the following Tcl command in scripts to modify the state machine encoding.

```
set_global_assignment -name state_machine_processing <value>
```

In this case, `<value>` can be AUTO, ONE-HOT, MINIMAL BITS, or USER-ENCODE.

### 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, optimizing across hierarchical boundaries allows the synthesis tool to perform the most logic minimization, which can reduce area. Therefore, to achieve the best results, flatten your design hierarchy whenever possible. If you are using Quartus II integrated synthesis, ensure that the **Preserve Hierarchical Boundary** logic option is turned off; that is, make sure that you have not turned on the option in the Assignment Editor or with Tcl assignments. If you are using Quartus II incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions. Follow Altera’s recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

For more information about using incremental compilation and recommendations for design partitioning, refer to the **Quartus II Incremental Compilation for Hierarchical and Team-Based Design** chapter in volume 1 of the **Quartus II Handbook**.

### Retarget Memory Blocks

If the design fails to fit because it runs out of device memory resources, your design might require a certain type of memory the device does not have. For example, a design that requires two M-RAM blocks can be targeted to a Stratix EP1S10 device, which has only one M-RAM block. You might be able to obtain a fit by building one of the memories with a different size memory block, such as an M4K memory block.

If the memory block was created with the MegaWizard™ Plug-In Manager, open the MegaWizard Plug-In Manager and edit the RAM block type so it targets a new memory block size.

ROM and RAM memory blocks can also be inferred from your HDL code, and your synthesis software can place large shift registers into memory blocks by inferring the ALTSHIFT_TAPS megafunction. This inference can be turned off in your synthesis tool to cause the memory to be placed in logic instead of in memory blocks. To disable inference when using Quartus II integrated synthesis, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.

2. In the **Category** list, select **Analysis & Synthesis**. The **Analysis & Synthesis** page appears.
3. Turn off the Auto RAM Replacement, Auto ROM Replacement, or Auto Shift Register Replacement logic option as appropriate for your project. Or, disable the option for a specific entity in the Assignment Editor.

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

Consider the resource utilization by hierarchy in the report file, and determine whether there is an unusually high register count in any of the modules. Some coding styles can prevent the Quartus II software from inferring RAM blocks from the source code because of their architectural implementation, and forces the software to implement the logic in flipflops. As an example, a function such as an asynchronous reset on a register bank might make it incompatible with the RAM blocks in the device architecture, so that the register bank is implemented in flipflops. It is often possible to move a large register bank into RAM by slight modification of associated logic.

For more information about memory inference control in other synthesis tools, refer to the appropriate chapter in Section III. Synthesis in volume 1 of the Quartus II Handbook, or your synthesis software’s documentation. For more information about coding styles and HDL examples that ensure memory inference, refer to the Recommended HDL Coding Styles chapter in volume 1 of the Quartus II Handbook.

Use Physical Synthesis Options to Reduce Area
The physical synthesis options for fitting can help you decrease the resource usage; additional optimizations for fitting are available. When you enable these settings for physical synthesis for fitting, the Quartus II software makes placement-specific changes to the netlist that reduce resource utilization for a specific Altera device.

The compilation time might increase considerably when you use physical synthesis options.

With the Quartus II software, you can apply physical synthesis options to specific instances, which can reduce the impact on compilation time. Physical synthesis instance assignments allow you to enable physical synthesis algorithms for specific portions of their design.

If you want the performance gain from physical synthesis, but do not want a specific hierarchy of the design to be modified, you can selectively disable physical synthesis for that hierarchy. Likewise, if you do not want to run physical synthesis for most parts of the design, but require the algorithms for a specific module in the design, you can enable physical synthesis for a single module.

The following physical synthesis optimizations for fitting are available:

- Physical synthesis for combinational logic
- Map logic into memory
On the Assignments menu, click Settings. In the Category list, expand Compilation Process Settings and select Physical Synthesis Optimization. The Physical Synthesis Optimization page appears. Under Optimize for fitting, turn on the options to enable physical synthesis optimizations during fitting. You can also specify the physical synthesis effort, which sets the level of physical synthesis optimization that you want the Quartus II software to perform.

The Perform physical synthesis for combinational logic option allows the Quartus II Fitter to resynthesize the combinational logic in a design to reduce the resource utilization to help achieve a fit.

The Perform logic to memory mapping option allows the Quartus II Fitter to automatically map logic into unused memory blocks during fitting, reducing the number of logic elements required to implement the design.

To apply physical synthesis assignments for fitting on a per instance basis, use the Quartus II Assignment Editor. The following assignments are available as instance assignments for fitting:

- Perform physical synthesis for combinational logic
- Perform logic to memory mapping

In the Assignment Editor, indicate the module instance you want to apply the setting to in the To tab. Select the required physical synthesis assignment in the Assignment Name tab. In the Value tab, select ON. In the Enabled tab, select Yes.

Retarget or Balance DSP Blocks

A design might not fit because it requires too many DSP blocks. All DSP block functions can be implemented 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 MegaWizard Plug-In Manager, open the MegaWizard Plug-In Manager and edit the function so it targets logic cells instead of DSP blocks. The Quartus II software uses the DEDICATED_MULTIPLIER_CIRCUITRY megafuction parameter to control the implementation.

DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. This inference can be turned off in your synthesis tool. When you are using Quartus II integrated synthesis, you can disable inference by turning off the Auto DSP Block Replacement logic option for your entire project. On the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and turn off Auto DSP Block Replacement. Alternatively, you can disable the option for a specific block with the Assignment Editor.

For more information about disabling DSP block inference in other synthesis tools, refer to the appropriate chapter in Section III. Synthesis in volume 1 of the Quartus II Handbook, or your synthesis software’s documentation.
The Quartus II 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 Quartus II 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 megafuntion variations.

For more details about the Quartus II logic options described in this section, refer to the Quartus II Help.

**Optimize Source Code**

If your design does not fit because of logic utilization, and the methods described in the preceding sections do not sufficiently improve the resource utilization of the design, modify the design at the source to achieve the desired results. 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 LEs or ALMs, but you have unused memory or DSP blocks, check to see 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 Quartus II 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.

For coding style guidelines, including examples of HDL code for inferring memory and DSP functions, refer to the “Instantiating Altera Megafuntions” and the “Inferring Altera Megafuntions” sections of the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*. For guidelines and sample HDL code for state machines, refer to the “General Coding Guidelines” section of the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

**Use a Larger Device**

If a successful fit cannot be achieved because of a shortage of LEs or ALMs, memory, or DSP blocks, you might require a larger device.

**Routing**

Use the suggestions in the following subsections to help you resolve routing resource problems.
Set Auto Register Packing to Sparse or Sparse Auto

This option is useful for reducing LE or ALM count in a design. This option is available for Arria GX, Cyclone, and Stratix series of devices. On the Assignments menu, click Settings. The Settings dialog box appears. In the Category list, select Fitter Settings. Click More Settings. Under Option, in the Name list, select Auto Packed Registers. In the Settings list, select the Sparse or Sparse Auto from the list.

When you select Sparse, the Fitter combines functions to improve the performance of many designs. When you select Sparse Auto, the Fitter attempts to achieve the highest performance with the possibility of increasing the area, but without exceeding the logic capacity of the device. These options might help improve the routing because they do not aggressively pack registers.

Selecting the default Auto setting can help routing in many designs. However, for some dense designs, the Fitter attempts to combine additional logic to reduce the area of the design to achieve the best performance. It does this by fitting the design within the best area of the selected device. Therefore, the Fitter can turn on the more aggressive Minimize the area with chains option, making it more difficult to route the design.

As an alternative, select Normal, and then increase the aggressiveness of register packing to reduce LE/ALM count if the design does not fit.

When you select a register packing setting to perform more aggressive register packing than the Auto setting, the extra register packing can affect the routability of the design as an unintended result. The Minimize the area with chains setting restricts placement and reduces routability significantly more than using the Minimize Area setting. For more information about register packing, refer to “Use Register Packing” on page 10–22.

Set Fitter Aggressive Routability Optimizations to Always

If routing resources are resulting in no-fit errors, use this option to reduce routing wire utilization. On the Assignments menu, click Settings. In the Category list, select Fitter Settings. Click More Settings. In the More Fitter Settings dialog box, set Fitter Aggressive Routability Optimizations to Always and click OK.

If there is a significant imbalance between placement and routing time (during the first fitting attempt), it might be because of high wire utilization. By turning on this option, you might be able to reduce your compilation time.

On average, in Arria GX and Stratix II devices, this option saves approximately 3% wire utilization but can reduce performance by approximately 1%. In Stratix III devices, this option saves approximately 6% wire utilization, at the same time reducing the performance by approximately 3%. In Cyclone III devices, using this option saves approximately 4.5% wire utilization while reducing the performance by about 4%.

These optimizations are used automatically when the Fitter performs more than one fitting attempt, but turning the option on increases the optimization effort on the first fitting attempt. This option also ensures that the Quartus II software uses maximum optimization to reduce routability, even if the Fitter Effort is set to Auto Fit.
Increase Placement Effort Multiplier

Increasing the placement effort can improve the routability of the design, allowing the software to route a design that otherwise requires too many routing resources. On the Assignments menu, click Settings. In the Category list, select Fitter Settings. Click More Settings. In the More Fitter Settings dialog box, increase the value of the Placement Effort Multiplier to increase placement effort. The default value is 1.0. Legal values must be greater than 0 and can be non-integer values. Numbers less than 1 reduce the placement effort and might affect placement quality. Higher numbers increase compilation time but can improve placement quality. For example, a value of 4 increases fitting time by approximately 2 to 4 times but can improve results. Increasing the placement effort multiplier does not tend to improve timing optimization unless the design also has very high routing resource usage.

Increased effort is used automatically when the Fitter performs more than one fitting attempt. Setting a multiplier higher than one (before compilation) increases the optimization effort on the first fitting attempt. The second and third fitting loops increase the Placement Effort Multiplier to 4 and then to 16. These loops result in increased compilation times, with possible improvement in the quality of placement.

You can modify the Placement Effort Multiplier using the following Tcl command:

```
set_global_assignment -name PLACEMENT_EFFORT_MULTIPLIER <value>
```

<value> can be any positive, non-zero number.

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 (as high as 3 is generally reasonable) can improve routing quality at the expense of run-time on difficult-to-route circuits. 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 about 2%. There is usually no gain in performance beyond a multiplier value of 3.

You can set the Router Effort Multiplier to a value higher than the default value for difficult-to-route designs. To set the Router Effort Multiplier, on the Assignments menu, click Settings, and then click Fitter Settings. Click the More Settings button. From the options available, select Router Effort Multiplier and edit the value in the dialog box that appears.

You can modify the Router Effort Multiplier by entering the following Tcl command:

```
set_global_assignment -name ROUTER_EFFORT_MULTIPLIER <value>
```

<value> can be any positive, non-zero number.

Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit the targeted device. This can occur when location or LogicLock assignments are too strict and there are not enough routing resources.

In this case, use the Routing Congestion view in the Chip Planner to locate routing problems in the floorplan, then remove all location and LogicLock region assignments from that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and LogicLock 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. Remove LogicLock assignments in the Chip Planner, in the LogicLock 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.

For more information about the Routing Congestion view in the Chip Planner, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook. You can also refer to the Quartus II Help.

**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 important to improving the routing results in your design, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus II integrated synthesis, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and select Balanced or Area under Optimization Technique.

You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the Area setting (potentially at the expense of register-to-register timing performance). You can apply the setting to specific modules while leaving the default Optimization Technique setting at Balanced (for the best trade-off between area and speed for certain device families) or Speed. 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 the Quartus II software, the Balanced setting typically produces utilization results that are very similar to those obtained with the Area setting, with better performance results. The Area setting can yield better results in some unusual cases.

In some synthesis tools, not specifying an $f_{\text{MAX}}$ requirement can result in less resource utilization, which can improve routability.

For information about setting timing requirements and synthesis options in Quartus II integrated synthesis and other synthesis tools, refer to the appropriate chapter in Section III, Synthesis in volume 1 of the Quartus II Handbook, or your synthesis software’s documentation.
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.

Use a Larger Device

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

Timing Optimization Techniques (LUT-Based Devices)

This section contains guidelines if your design does not meet its timing requirements.

Timing Optimization Advisor

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

When you open the Timing Optimization Advisor after compilation, you find recommendations to improve the timing performance of your design. Some of the recommendations in these advisors can contradict each other. Altera recommends evaluating these options and choosing the settings that best suit the given requirements.

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 will meet its setup and hold time requirements. The mean time between failure (MTBF) is an estimate of the average time between instances when metastability could cause a design failure.

For more information about metastability and MTBF, refer to the Understanding Metastability in FPGAs white paper.

You can use the Quartus II software to analyze the average MTBF due to metastability when a design synchronizes asynchronous signals, and optimize the design to improve the MTBF. These metastability features are supported only for designs constrained with the TimeQuest 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.
For details about the metastability features in the Quartus II software, refer to the Managing Metastability with the Quartus II Software chapter in volume I of the Quartus II Handbook. 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.

I/O Timing Optimization

The example in Figure 10–6 shows the Timing Optimization Advisor after compiling a design that meets its frequency requirements, but requires setting changes to improve the timing.

Figure 10–6. Timing Optimization Advisor

When you expand one of the categories in the Advisor, such as Maximum Frequency (fmax) or I/O Timing (tsu, tco, tpd), the recommendations are divided into stages. The stages show the order in which you should 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 Figure 10–6, 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. Refer to the “How to use” page in the Advisor for a legend that provides more information for each icon.
There is a link from each recommendation to the appropriate location in the Quartus II user interface 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. In some cases, you can also use the Correct the Settings button to automatically make the suggested change to global settings.

For some entries in the advisor, a button appears that allows you to further analyze your design and gives you more information. The advisor provides a table with the clocks in the design and indicates whether they have been assigned a timing constraint.

The next stage of design optimization focuses on I/O timing. Ensure that you have made the appropriate assignments as described in “Initial Compilation: Required Settings” on page 10–3, and that the resource utilization is satisfactory before proceeding with I/O timing optimization. The suggestions provided in this section are applicable to all Altera FPGA families and to the MAX II family of CPLDs.

Because changes to the I/O paths affect the internal register-to-register timing, complete this stage before proceeding to the register-to-register timing optimization stage as described in the “Register-to-Register Timing Optimization Techniques (LUT-Based Devices)” on page 10–39.

The options presented in this section address how to improve I/O timing, including the setup delay ($t_{SU}$), hold time ($t_H$), and clock-to-output ($t_{CO}$) parameters.

### Improving Setup and Clock-to-Output Times Summary

Table 10–1 shows the recommended order in which to use techniques to reduce $t_{SU}$ and $t_{CO}$ times. Checkmarks indicate which timing parameters are affected by each technique. Reducing $t_{SU}$ times increases hold ($t_H$) times.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Affects $t_{SU}$</th>
<th>Affects $t_{CO}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure that the appropriate constraints are set for the failing I/Os</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(page 10–3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use timing-driven compilation for I/O (page 10–35)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Use fast input register (page 10–36)</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>Use fast output register, fast output enable register, and fast OCT</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>register (page 10–36)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decrease the value of Input Delay from Pin to Input Register</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>or set Decrease Input Delay to Input Register = ON (page 10–37)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decrease the value of Input Delay from Pin to Internal Cells</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>or set Decrease Input Delay to Internal Cells = ON (page 10–37)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decrease the value of Delay from Output Register to Output Pin</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>or set Increase Delay to Output Pin = OFF (page 10–37)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increase the value of Input Delay from Dual-Purpose Clock Pin to Fan-Out</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>Destinations (page 10–38)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use PLLs to shift clock edges (page 10–38)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Use the Fast Regional Clock (page 10–39)</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>For MAX II series of devices, set Guarantee I/O paths to zero, Hold</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>Time at Fast Timing Corner to OFF, or when $t_{SU}$ and $t_{H0}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constraints permit (page 10–39)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Timing Optimization Techniques (LUT-Based Devices)

Table 10–1. Improving Setup and Clock-to-Output Times  
(\textit{Note 1}) (Part 2 of 2)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Affects $t_{SU}$</th>
<th>Affects $t_{CO}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase the value of Delay to output enable pin or set Increase delay to output enable pin (page 10–38)</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

\textbf{Note to Table 10–1:}

(1) These options may not apply to all device families.

\textbf{Timing-Driven Compilation}

To perform IOC timing optimization using the \textit{Optimize IOC Register Placement For Timing} option, perform the following steps:

1. On the Assignments menu, click \textit{Settings}.
2. In the \textit{Category} list, select \textit{Fitter Settings} and click \textit{More Settings}.
3. In the \textit{More Fitter Settings} dialog box, under \textit{Existing option settings}, select \textit{Optimize IOC Register Placement for Timing}.

This option moves registers into I/O elements if required 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 option does not apply to MAX II series of devices because they do not contain I/O registers.

The \textit{Optimize IOC Register Placement for Timing} 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. This setting does not affect registers with any of the following characteristics:

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

Registers with the characteristics listed are optimized using the regular Quartus II Fitter optimizations.

\textbf{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. For an input register, use the \textit{Fast Input Register} option; for an output register, use the \textit{Fast Output Register} option; and for an output enable register, use the \textit{Fast Output Enable Register} option. Stratix II devices also support the \textit{Fast OCT (on-chip termination) Register} option. In the MAX II series of devices, which have no I/O registers, these assignments lock the register into the LAB adjacent to the I/O pin if there is a pin location assignment for that I/O pin.
If the fast I/O setting is on, the register is always placed in the I/O element. If the fast I/O setting is off, the register is never placed in the I/O element. This is true even if the Optimize IOC Register Placement for Timing option is turned on. If there is no fast I/O assignment, the Quartus II software determines whether to place registers in I/O elements if the Optimize IOC Register Placement for Timing option is turned on.

The four fast I/O options (Fast Input Register, Fast Output Register, Fast Output Enable Register, and Fast OCT Register) can also be used to override the location of a register that is in a LogicLock region, and force it into an I/O cell. If this assignment is applied to a register that feeds multiple pins, the register is duplicated and placed in all relevant I/O elements. In MAX II series of devices, the register is duplicated and placed in each distinct LAB location that is next to an I/O pin with a pin location assignment.

**Programmable Delays**

Various programmable delay options can be used to minimize the $t_{SU}$ and $t_{CO}$ times. For Arria GX, Stratix, and Cyclone series devices, and MAX II series of devices, the Quartus II software automatically adjusts the applicable programmable delays to help meet timing requirements. Programmable delays are advanced options that you should use only after you compile a project, check the I/O timing, and determine that the timing is unsatisfactory. 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 for 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 such as Arria II GX, Stratix IV, and Stratix III are user-controllable, Altera recommends their use for advanced users only. However, the Quartus II software might use the programmable delays internally during the Fitter phase.

For more details about Stratix III programmable delays, refer to the *Stratix III Device Handbook* and AN 474: *Implementing Stratix III Programmable I/O Delay Settings in the Quartus II Software*. For more information about using the Chip Planner and Resource Property Editor, refer to the *Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*.

Table 10–2 summarizes the programmable delays available for Altera devices.
## Table 10–2. Programmable Delays for Altera Devices (Part 1 of 2)

<table>
<thead>
<tr>
<th>Programmable Delay</th>
<th>Description</th>
<th>I/O Timing Impact</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrease input delay to input register</td>
<td>Decreases propagation delay from an input pin to the data input of the input register in the I/O cell associated with the pin. Applied to an input/bidirectional pin or register it feeds.</td>
<td>Decreases ( t_{su} ) Increases ( t_{pu} )</td>
<td>Stratix \n Stratix GX \n Cyclone \n MAX 7000B</td>
</tr>
<tr>
<td>Input delay from pin to input register</td>
<td>Sets propagation delay from an input pin to the data input of the input register implemented in the I/O cell associated with the pin. Applied to an input/bidirectional pin.</td>
<td>Changes ( t_{su} ) Changes ( t_{pu} )</td>
<td>Arria GX \n Stratix II \n Stratix II GX \n Cyclone III \n Cyclone II \n HardCopy series</td>
</tr>
<tr>
<td>Decrease input delay to internal cells</td>
<td>Decreases the propagation delay from an input or bidirectional pin to logic cells and embedded cells in the device. Applied to an input/bidirectional pin or register it feeds.</td>
<td>Decreases ( t_{su} ) Increases ( t_{pu} )</td>
<td>Arria GX \n Stratix \n Stratix GX \n Cyclone</td>
</tr>
<tr>
<td>Input delay from pin to internal cells</td>
<td>Sets the propagation delay from an input or bidirectional pin to logic and embedded cells in the device. Applied to an input or bidirectional pin.</td>
<td>Changes ( t_{su} ) Changes ( t_{pu} )</td>
<td>Arria GX \n Stratix II \n Stratix II GX \n Cyclone III \n Cyclone II \n HardCopy series \n MAX IIE and MAX IIIZ</td>
</tr>
<tr>
<td>Decrease input delay to output register</td>
<td>Decreases the propagation delay from the interior of the device to an output register in an I/O cell. Applied to an input/bidirectional pin or register it feeds.</td>
<td>Decreases ( t_{pD} )</td>
<td>HardCopy series \n Stratix \n Stratix GX</td>
</tr>
<tr>
<td>Increase delay to output enable pin</td>
<td>Increases the propagation delay through the tri-state output to the pin. The signal can either come from internal logic or the output enable register in an I/O cell. Applied to an output/bidirectional pin or register feeding it.</td>
<td>Increases ( t_{oE} )</td>
<td>Stratix \n Stratix GX \n HardCopy series</td>
</tr>
<tr>
<td>Delay to output enable pin</td>
<td>Sets the propagation delay to an output enable pin from internal logic or the output enable register implemented in an I/O cell.</td>
<td>Changes ( t_{oE} )</td>
<td>Arria GX \n Stratix II \n Stratix II GX \n Cyclone III</td>
</tr>
<tr>
<td>Increase delay to output pin</td>
<td>Increases the propagation delay to the output or bidirectional pin from internal logic or the output register in an I/O cell. Applied to output/bidirectional pin or register feeding it.</td>
<td>Increases ( t_{oE} )</td>
<td>Stratix \n Stratix GX \n Cyclone</td>
</tr>
</tbody>
</table>
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 $t_{CO}$ at the expense of $t_{SU}$, while shifting it forward gives a better $t_{SU}$ at the expense of $t_{CO}$ and $t_{H}$. Refer to Figure 10–7. This technique can be used only in devices that offer PLLs with the phase shift option.

**Figure 10–7.** Shift Clock Edges Forward to Improve $t_{SU}$ at the Expense of $t_{CO}$

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**, described in Table 10–2.
Use Fast Regional Clock Networks and Regional Clocks Networks

Altera devices have a variety of hierarchical clock structures. These include dedicated global clock networks (GCLKs), regional clock networks (RCLKs), fast regional clock networks (FCLK) and periphery clock networks (PCLKs). The available resources differ between various Altera device families. Refer to the appropriate device handbook to get the number of various clocking resources available in your target device.

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. Regional clocks provide the lowest clock delay and skew for logic contained in a single quadrant. Placing clocks on these low-skew and low-delay clock nets provides better tCO performance.

Change How Hold Times are Optimized for MAX II Devices

For MAX II series of devices, you can use the **Guarantee I/O paths have zero hold time at Fast Timing Corner** option to control how hold time is optimized by the Quartus II software. On the Assignments menu, click **Settings**. In the **Category** list, select **Fitter Settings**. Click **More Settings**. In the **More Fitter Settings** dialog box, set the option globally. Or, on the Assignments menu, click **Assignment Editor** to set this option for specific I/Os.

The option controls whether the Fitter uses timing-driven compilation to optimize a design to achieve a zero hold time for I/Os that feed globally clocked registers at the fast (best-case) timing corner, even in the absence of any user timing assignments. When this option is set to **On** (default), the Fitter guarantees zero hold time (tH) for I/Os feeding globally clocked registers at the fast timing corner, at the expense of possibly violating tSU or tPD timing constraints. When this option is set to **When tsu and tpd constraints permit**, the Fitter achieves zero hold time for I/Os feeding globally clocked registers at the fast timing corner only when tSU or tPD timing constraints are not violated. When this option is set to **Off**, designs are optimized to meet user timing assignments only.

By setting this option to **Off** or **When tsu and tpd constraints permit**, you improve tSU at the expense of tH.

Register-to-Register Timing Optimization Techniques (LUT-Based Devices)

The next stage of design optimization is to improve register-to-register (fMAX) timing. There are a number of options available if the performance requirements are not achieved after compilation.

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

For more details about synchronous design practices and coding styles, refer to the *Design Recommendations for Altera Devices and the Quartus II Design Assistant* chapter in volume 1 of the *Quartus II Handbook*.

When using the Quartus II TimeQuest Timing Analyzer, register-to-register timing optimization is the same as maximizing the slack on the clock domains in your domain. You can use the techniques described in this section to improve the slack on different timing paths in your design.
Before optimizing your design, you should understand the structure of your design as well as the type of logic affected by each optimization. An optimization can decrease performance if the optimization does not benefit your logic structure.

**Improving Register-to-Register Timing Summary**

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. For details, refer to “Timing Requirement Settings” on page 10–4.

2. Ensure that you have reviewed all warning messages from your initial compilation and check for ignored timing assignments. Refer to “Design Analysis” on page 10–11 for details and fix any of these problems before proceeding with optimization.

3. Apply netlist synthesis optimization options and physical synthesis.

4. Try different Fitter seeds (page 10–47). You can omit this step if a large number of critical paths are failing, or if the paths are failing badly.

5. Apply the following synthesis options to optimize for speed:
   - “Optimize Synthesis for Speed, Not Area” (page 10–43)
   - “Flatten the Hierarchy During Synthesis” (page 10–25)
   - “Set the Synthesis Effort to High” (page 10–44)
   - “Change State Machine Encoding” (page 10–44)
   - “Prevent Shift Register Inference” (page 10–46)
   - “Use Other Synthesis Options Available in Your Synthesis Tool” (page 10–46)

6. Make LogicLock assignments (page 10–48) to control placement.

7. Make design source code modifications to fix areas of the design that are still failing timing requirements by significant amounts (page 10–48).

8. Make location assignments, or as a last resort, perform manual placement by back-annotating the design (page 10–51).

You can use the Design Space Explorer (DSE) to automate the process of running several different compilations with different settings. For more information, refer to the Design Space Explorer chapter in volume 2 of the Quartus II Handbook.

If these techniques do not achieve performance requirements, additional design source code modifications might be required (page 10–48).

**Physical Synthesis Optimizations**

The Quartus II software offers physical synthesis optimizations that can help improve the performance of many designs regardless of the synthesis tool used. Physical synthesis optimizations can be applied both during synthesis and during fitting.
Physical synthesis optimizations that occur during the synthesis stage of the Quartus II compilation operate either on the output from another EDA synthesis tool or as an intermediate step in Quartus II integrated synthesis. These optimizations make changes to the synthesis netlist to improve either area or speed, depending on your selected optimization technique and effort level.

To view and modify the synthesis netlist optimization options, on the Assignments menu, click Settings. In the Category list, expand Compilation Process Settings and select Physical Synthesis Optimizations.

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

To turn on the Perform WYSIWYG Primitive Resynthesis option, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings and check the box for Perform WYSIWYG Primitive Resynthesis.

The Quartus II technology mapper optimizes the design for Speed, Area, or Balanced, according to the setting of the Optimization Technique option. To change this setting, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and select Speed or Balanced under Optimization Technique.

The physical synthesis optimizations occur during the Fitter stage of the Quartus II compilation. Physical synthesis optimizations make placement-specific changes to the netlist that improve speed performance results for a specific Altera device.

The following physical synthesis optimizations are available during the Fitter stage for improving performance:

- Physical synthesis for combinational logic
- Automatic asynchronous signal pipelining
- Physical synthesis for registers
  - Register duplication
  - Register retiming

You can apply physical synthesis options on specific instances if you want the performance gain from physical synthesis only on parts of your design.

To view and modify the Physical Synthesis Optimizations, on the Assignments menu, click Settings. In the Category list, select Fitter Settings, and specify the physical synthesis optimization options on the Physical Synthesis Optimizations page. You can also specify the Physical synthesis effort, which sets the level of physical synthesis optimization that you want the Quartus II software to perform.

The Perform physical synthesis for combinational logic option allows the Quartus II Fitter to resynthesize the combinational logic in a design to reduce delay along the critical path and improve design performance.
The **Perform automatic asynchronous signal pipelining** option allows the Quartus II Fitter to insert pipeline stages for asynchronous clear and asynchronous load signals automatically during fitting to increase circuit performance. You can use this option if asynchronous control signal recovery and removal times do not meet your requirements. The option improves performance for designs in which asynchronous signals in very fast clock domains cannot be distributed across the chip quickly enough (because of long global network delays).

To apply physical synthesis assignments for fitting on a per instance basis, use the Quartus II Assignment Editor. The following assignments are available as instance assignments:

- Perform physical synthesis for combinational logic
- Perform register duplication for performance
- Perform register retiming for performance
- Perform automatic asynchronous signal pipelining

In the Assignment Editor, indicate the module instance you want to apply to the specific physical synthesis setting in the To tab. Select the required physical synthesis assignment in the Assignment Name tab. In the Value tab, select ON. In the Enabled tab, select Yes.

The **Perform automatic asynchronous signal pipelining** option adds registers to nets driving the asynchronous clear or asynchronous load ports of registers. This adds register delays (and latency) to the reset, adding the same number of register delays for each destination using the reset. Therefore, the option should be used only when adding latency to reset signals does not violate any design requirements. This option also prevents the promotion of signals to use global routing resources.

The **Perform register duplication** physical synthesis option allows the Quartus II software to duplicate registers based on Fitter placement information to improve design performance. The Fitter can also duplicate combinational logic when this option is enabled.

The **Perform register retiming** physical synthesis option allows the Quartus II software to move registers across combinational logic to balance timing. This option applies to registers and combinational logic that have already been placed into logic cells.

The Quartus II software generally does not retime register paths that cross clock domains. However, if you are using the Classic Timing Analyzer and have a universal \( f_{\text{MAX}} \) specified for your compilation, the Quartus II software considers all clocks as related to each other, and might retime paths between clock domains. To avoid the retiming of paths, specify individual \( f_{\text{MAX}} \) requirements for each of the clock domains in your design when using the Classic Timing Analyzer.

You can perform physical synthesis during the fitting stage to improve the fitting results as well. The Quartus II software performs the optimizations that help achieve a better fit when you turn on the **Perform physical synthesis for combinational logic** option.
The Fitter performs physical synthesis optimizations on logic and registers, allowing the mapping of logic and registers into unused memory blocks in the device to achieve a fit, when you turn on the Perform logic to memory mapping option.

For more information and detailed descriptions of these netlist optimization options, refer to the Netlist Optimizations and Physical Synthesis chapter in volume 2 of the Quartus II Handbook.

Because performance results are design-dependent, try the physical synthesis options in different combinations until you achieve the best results. Generally, turning on all the options gives the best results but significantly increases compilation time. The following information provides typical benchmark results on different designs with varying amounts of logic using synthesis netlists from leading third-party synthesis tools and compiled with the Quartus II software. These results use the default Balanced setting for the Optimization Technique for WYSIWYG resynthesis. Changing the setting to Speed or Area can affect your results.

In many designs, using WYSIWYG primitive resynthesis can reduce area or improve \( f_{\text{MAX}} \). By using other physical synthesis options for combinational logic and registers, you might be able to achieve an additional increase in \( f_{\text{MAX}} \).

Compilation time might increase considerably when you use high physical synthesis effort levels. The optimizations are design dependent, and some designs might not improve much with physical synthesis.

**Turn Off Extra-Effort Power Optimization Settings**

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

To change the PowerPlay power optimization level, on the Assignments menu, click Settings. The Settings dialog box appears. From the Category list, select Analysis & Synthesis Settings. From the pull-down menu, select the appropriate level of PowerPlay power optimization level.

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

**Optimize Synthesis for Speed, Not Area**

The manner in which the design is synthesized has a large impact on design performance. Design performance varies depending on the way the design is coded, the synthesis tool used, and the options specified when synthesizing. Change your synthesis options if a large number of paths are failing, or if specific paths are failing badly and have many levels of logic.

Set your device and timing constraints in your synthesis tool. Synthesis tools are timing-driven and optimized to meet specified timing requirements. If you do not specify target frequency, some synthesis tools optimize for area.

Some synthesis tools offer an easy way to instruct the tool to focus on speed instead of area.
For Quartus II integrated synthesis, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and specify Speed as the Optimization Technique option. You can also specify this logic option 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 to specify that all combinational logic in or between the specified clock domain(s) is 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 the DSE to experiment with different Quartus II synthesis options to optimize your design for the best performance.

For information about setting timing requirements and synthesis options in Quartus II integrated synthesis and third-party synthesis tools, refer to the appropriate chapter in Section III. Synthesis in volume 1 of the Quartus II Handbook, or refer to your synthesis software documentation.

**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. If you are using Quartus II integrated synthesis, ensure that the Preserve Hierarchical Boundary option is turned off. If you are using Quartus II incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions. Follow Altera’s recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

For more information about using incremental compilation and recommendations for design partitioning, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook.

**Set the Synthesis Effort to High**

Some 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.

**Change State Machine Encoding**

State machines can be encoded using various techniques. One-hot encoding, which uses one register for every state bit, usually provides the best performance. If your design contains state machines, changing the state machine encoding to one-hot can improve performance at the cost of area.
If your design does not manually encode the state bits, you can select the state machine encoding chosen in your synthesis tool. In Quartus II integrated synthesis, on the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and for State Machine Processing, select One-Hot. You also can specify this logic option for specific modules or state machines in your design with the Assignment Editor.

In some cases (especially in Stratix II and Stratix III devices), encoding styles other than the default offer better performance. Experiment with different encoding styles to see what effect the style has on your resource utilization and timing performance.

**Duplicate Logic for Fan-Out Control**

Duplicating logic or registers can help improve timing in cases where moving a register in a failing timing path to reduce routing delay creates other failing paths, or where there are timing problems due to the fan-out of the registers. Most often, timing failures occur not because of the high Fan-Out registers, but because of the location of those registers. Duplicating registers, where source and destination registers are physically close, can help improve slack on critical paths.

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

Logic duplication using Maximum Fan-Out assignments normally increases resource utilization and can potentially increase compilation time, depending on the placement and the total resource usage within the selected device. The improvement in timing performance that results because of Maximum Fan-Out assignments is very design-specific.

If you are using Maximum Fan-Out assignments, Altera recommends benchmarking 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 Quartus II software regardless of the synthesis tool used. To duplicate a register, apply the Manual Logic Duplication option to the register with the Assignment Editor.

The Manual Logic Duplication option also accepts wildcards. This is an easy and powerful duplication technique that you can use without editing your source code. For example, you can use this technique to make a duplicate of a large fan-out node for all of its destinations in a certain design hierarchy, such as hierarchy_A. To apply such an assignment in the Assignment Editor, make an entry such as the one shown in Table 10–3.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Assignment Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>My_high_fanout_node</td>
<td><em>hierarchy_A</em></td>
<td>Manual Logic Duplication</td>
<td>high_fanout_to_A</td>
</tr>
</tbody>
</table>
For more information about the manual logic duplication option, refer to the Quartus II Help.

**Prevent Shift Register Inference**

In some cases, turning off the inference of shift registers increases performance. Doing so forces the software to use logic cells to implement the shift register instead of implementing the registers in memory blocks using the ALTSHIFT_TAPS megafunction. If you implement shift registers in logic cells instead of memory, logic utilization is increased.

**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. They typically increase the resource utilization of your design.

**Fitter Seed**

The Fitter seed affects the initial placement configuration of the design. Changing the seed value changes the Fitter results, because the fitting results change whenever there is a change in the initial conditions. Each seed value results in a somewhat different fit, and you can experiment with several different seeds to attempt to obtain better fitting results and timing performance.

When there are changes in your design, there is some random variation in performance between compilations. This variation is inherent in placement and routing algorithms—there are too many possibilities to try them all and get the absolute best result, so the initial conditions change the compilation result.

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, **Analysis & Synthesis Settings**, **Fitter 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 slightly 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 could be 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 has improved after an optimization change and whether a setting that increases compilation time has benefits worth the increased time (such as setting the **Physical Synthesis Effort** to **Extra**). The sweep also shows the amount of random variation you should 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 will likely have to perform seed sweep again.

On the Assignments menu, select Fitter Settings to control the initial placement with the seed. You can use the DSE to perform a seed sweep easily.

You can use the following Tcl command from a script to specify a Fitter seed:

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

For more information about compiling with different seeds using the DSE script, refer to the Design Space Explorer chapter in volume 2 of the Quartus II Handbook.

**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 timing requirements. Setting this option to Maximum can increase design speed slightly at the cost of increased compilation time. Setting this option to Minimum can reduce compilation time at the cost of slightly reduced design speed. The default value is Normal.

To modify the Router Timing Optimization Level, on the Assignments menu, click Settings. The Settings dialog box appears. In the Category list, click Fitter Settings. Click on the More Settings tab. From the available settings, select Router Timing Optimization Level and select the required setting from the list.

**Enable Beneficial Skew Optimization**

The Quartus II Fitter intentionally inserts some small delays on global clock networks to improve performance on designs that target Arria II GX, Stratix IV, Stratix III, and Cyclone III devices. This is called beneficial skew optimization and is enabled by default for devices that support this feature. The value of skew introduced depends on the device family and the speed grade of the chosen device. For example, when this option is turned on for a Stratix III device (-2 speed grade), a skew value of approximately 150 ps is introduced if the inclusion improves the timing performance of your design. If you are targeting a Cyclone III device (-6 speed grade), the delay value introduced is approximately 350 ps. For Arria II GX and Stratix IV devices, an approximate skew of 100 ps could be introduced. To enable the Beneficial Skew Optimization option, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
2. In the Category list, select Fitter Settings. The Fitter Settings page appears.
4. Under Options, in the Name list, select Enable Beneficial Skew Optimization. In the Setting list, select On.
5. Click OK.
6. In the Settings dialog box, click OK.

When you turn on Enable Beneficial Skew Optimization globally, you can disable skew insertion on a particular clock or destination by using an instance level ENABLE_BENEFICIAL_SKEW_OPTIMIZATION assignment.
When you want to use **Enable Beneficial Skew Optimization**, you must also set the **Optimize hold timing** option to **All paths** (in the Fitter Settings page of the Settings dialog box). If you turn on **Enable Beneficial Skew Optimization**, the fitter overrides the setting of **Optimize hold timing** if it is not set to **All paths**, and displays an info message describing the change.

### Optimize Source Code

If the methods described in the preceding sections do not sufficiently improve timing of the design, modify your design files to achieve the desired results. Try restructuring the design to use pipelining or more efficient coding techniques. 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 LogicLock or location assignments.

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.

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 Quartus II 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.

For coding style guidelines including examples of HDL code for inferring memory, functions, guidelines, and sample HDL code for state machines, refer to the **Recommended HDL Coding Styles** chapter in volume 1 of the Quartus II Handbook.

### LogicLock Assignments

Using LogicLock assignments to improve timing performance is only recommended for older Altera devices, such as the MAX II family. For designs using these devices, you can make LogicLock assignments for based nodes optimization, design hierarchy, or critical paths. This method can be used if a large number of paths are failing, and recoding the design does not seem to be necessary. LogicLock assignments can help if routing delays form a large portion of your critical path delay, and placing logic closer together in the device improves the routing delay.

Improving fitting results with LogicLock assignments, especially for larger devices, such as the Stratix and Arria GX series of devices, can be difficult. The LogicLock feature is intended to be used for performance preservation and to floorplan your design. Therefore, LogicLock assignments do not always improve the performance of the design. In many cases, you cannot improve upon results from the Fitter by making location assignments.
If there are existing LogicLock assignments in your design, remove the assignments if your design methodology permits it. Recompile the design to see if the assignments are making the performance worse.

When making LogicLock assignments, it is important to consider how much flexibility to give the Fitter. LogicLock assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher Fitter effort, but reduce the chance of design over-constraint. The following types of LogicLock assignments are available, listed in the order of decreasing flexibility:

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

For more information about using LogicLock regions, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

To determine what to put into a LogicLock region, refer to the timing analysis results and analyze the critical paths in the Chip Planner. The register-to-register timing paths in the Timing Analyzer section of the Compilation Report help you recognize patterns.

The following sections describe cases in which LogicLock regions can help to optimize a design.

**Hierarchy Assignments**

For a design with the hierarchy shown in Figure 10–8, which has failing paths in the timing analysis results similar to those shown in Table 10–4, mod_A is probably a problem module. In this case, a good strategy to fix the failing paths is to place the mod_A hierarchy block in a LogicLock region so that all the nodes are closer together in the floorplan.

**Figure 10–8.** Design Hierarchy

![Design Hierarchy Diagram]

Table 10–4 shows the failing paths connecting two regions together within mod_A listed in the timing analysis report.

**Table 10–4.** Failing Paths in a Module Listed in Timing Analysis

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod_A</td>
<td>reg1</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg3</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg4</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg7</td>
</tr>
<tr>
<td>mod_A</td>
<td>reg0</td>
</tr>
</tbody>
</table>
Hierarchical LogicLock regions are also important if you are using an incremental compilation flow. Each design partition for incremental compilation should be placed in a separate LogicLock region to reduce conflicts and ensure good results as the design develops. You can use auto size and floating location regions to find a good design floorplan, but you should fix the size and placement to achieve the best results in future compilations.

For more information about using incremental compilation and recommendations for creating a design floorplan using LogicLock regions, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design and Best Practices for Incremental Compilation and Floorplan Assignments chapters in volume 1 of the Quartus II Handbook, and Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

Path Assignments

If you see a pattern such as the one shown in Figure 10–9 and Table 10–5, it often indicates paths with a common problem. In this case, a path-based assignment can be made from all `d_reg` registers to all `memaddr` registers. You can make a path-based assignment to place all source registers, destination registers, and the nodes between them in a LogicLock region with the wildcard characters “*” and “?.”

You can also explicitly place the nodes of a critical path in a LogicLock region. However, using this method instead of path assignments can result in alternate paths between the source and destination registers becoming critical paths.

**Figure 10–9.** Failing Paths in Timing Analysis

Table 10–5 shows the failing paths listed in the timing analysis report.

**Table 10–5.** Failing Paths in Timing Analysis (Part 1 of 2)

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
</table>
For more information about path-based LogicLock assignments, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

### Location Assignments and Back-Annotation

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 Quartus II Fitter than LogicLock assignments. In some cases, when you are familiar with your design, you can enter location constraints in a way that produces better results.

Improving fitting results, especially for larger devices, such as the Stratix and Arria GX series of 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.

The following commonly used location assignments are listed in the order of decreasing flexibility:

- Custom regions
- Back-annotated LAB location assignments
- Back-annotated LE or ALM location assignments

#### Custom Regions

A custom region is a rectangular region containing user-assigned nodes, which are constrained in the region’s boundaries. If any portion of a block in the device floorplan overlaps a custom region, such as an M-RAM block, it is considered to be entirely in that region.

Custom regions are hard location assignments that cannot be overridden and are very similar to fixed-size, locked-location, LogicLock regions. Custom regions are commonly used when logic must be constrained to a specific portion of the device.

#### Back-Annotation and Manual Placement

Assigning the location of nodes in a design to the locations to which they were assigned during the last compilation is called “back-annotation.” When nodes are locked to their assigned locations in a back-annotated design, you can manually move specific nodes without affecting other back-annotated nodes. The process of manually moving and reassigning specific nodes is called manual placement.

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>d_reg[2]</code></td>
<td><code>memaddr[0]</code></td>
</tr>
</tbody>
</table>

Table 10–5. Failing Paths in Timing Analysis (Part 2 of 2)
Back-annotation is very restrictive to the compiler, so you should back-annotate only when the design has been finalized and no further changes are expected. Assignments can become invalid if the design is changed. Combinational nodes often change names when a design is resynthesized, even if they are unrelated to the logic that was changed.

Moving nodes manually can be very difficult for large devices. In many cases, you cannot improve upon the Fitter’s results. Illegal or unroutable location constraints can cause “no fit” errors. Before making location assignments, determine whether to back-annotate to lock down the assigned locations of all nodes in the design. When you are using a hierarchical design flow, you can lock down node locations in one LogicLock region only, while other node locations are left floating in a fixed LogicLock region. By implementing a hierarchical approach, you can use the LogicLock design methodology to reduce the dependence of logic blocks on other logic blocks in the device.

Consistent node names are required to perform back-annotation. If you use Quartus II integrated synthesis or any Quartus II optimizations, such as the WYSIWYG primitive resynthesis netlist optimization or any physical synthesis optimizations, you must create an atom netlist before you back-annotate to lock down the placement of any nodes. This creates consistent node names.

Physical synthesis optimizations are placement-specific as well as design-specific. Unless you back-annotate the design before recompilation, the physical synthesis results can differ. This happens because the atom netlist creates different placement results. By back-annotating the design, the design source and the atom netlist use the same placement when the design is recompiled. When you use an atom netlist and you want to maintain the same placement results as a previous compilation, use LogicLock regions and back-annotate the placement of all nodes in the design. Not back-annotating the design can result in the design source and the atom netlist having different placement results and therefore different synthesis results.

For more information about creating atom netlists for your design, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

When you back-annotate a design, you can choose whether to assign the nodes either to LABs (this is preferred because of increased flexibility) or LEs/ALMs. You also can choose to back-annotate routing to further restrict the Fitter and force a specific routing within the device.

Using back-annotated routing with physical synthesis optimizations can result in a routing failure.

For more information about back-annotating routing, refer to the Quartus II Help.

When performing manual placement at a detailed level, Altera recommends that you move LABs, not logic cells (LEs or ALMs). The Quartus II software places nodes that share the same control signals in appropriate LABs. Successful placement and routing is more difficult when you move individual logic cells. This is because LEs with different control signals that are put into the same LAB might not have any unused control signals available, and the design might not fit.
In general, when you are performing manual placement and routing, fix all I/O paths first, because often fewer options are available to meet I/O timing. After I/O timing is met, focus on manually placing register-to-register timing paths. This strategy is consistent with the methodology outlined in this chapter.

The best way to meet performance is to move nodes closer together. For a critical path such as the one shown in Figure 10–10, moving the destination node closer to the other nodes reduces the delay and helps meet your timing requirements.

![Figure 10–10. Reducing Delay of Critical Path](image)

Optimizing Placement for Stratix, Stratix II, Arria GX, and Cyclone II Devices

In the Arria GX, Stratix, and Cyclone series of devices, the row interconnect delay is slightly faster than the column interconnect delay. Therefore, when placing nodes, optimal placement is typically an ellipse around the source or destination node. In Figure 10–11, if the source is located in the center, any of the shaded LABs should give approximately the same delay.

![Figure 10–11. Possible Optimal Placement Ellipse](image)

In addition, you should avoid crossing any M-RAM memory blocks for node-to-node routing, because routing paths across M-RAM blocks requires using R24 or C16 routing lines.

The Quartus II software calculates the interconnect delay based on different electrical characteristics of each individual wire, such as the length, fan-out, distribution of the parasitic loading on the wire, and so forth.
To determine the actual delays to and from a resource, use the **Show Physical Timing Estimate** feature in the Chip Planner.

For more information about using the Chip Planner, refer to the *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*.

### Optimizing Placement for Cyclone Devices

In Cyclone devices, the row and column interconnect delays are similar; therefore, when placing nodes, optimal placement is typically a circle around the source or destination node.

Try to avoid long routes across the device. Long routes require more than one routing line to cross the Cyclone device.

### Resource Utilization Optimization Techniques (Macrocell-Based CPLDs)

The following recommendations help you take advantage of the macrocell-based architecture in the MAX 7000 and MAX 3000 devices to yield maximum speed, reliability, and device resource utilization while minimizing fitting difficulties.

After design analysis, the first stage of design optimization is to improve resource utilization. Complete this stage before proceeding to timing optimization. First, ensure that you have set the basic constraints described in “Initial Compilation: Required Settings” on page 10–3. If your design is not fitting into a specified device, use the techniques in this section to achieve a successful fit.

### Use Dedicated Inputs for Global Control Signals

MAX 7000 and MAX 3000 devices have four dedicated inputs that can be used for global register control. Because the global register control signals can bypass the logic cell array and directly feed registers, product terms can be preserved for primary logic. Also, because each signal has a dedicated path into the LAB, global signals also can bypass logic and data path interconnect resources.

Because the dedicated input pins are designed for high fan-out control signals and provide low skew, you should always assign global signals (such as clock, clear, and output enable) to the dedicated input pins.

You can use logic-generated control signals for global control signals instead of dedicated inputs. However, the following list shows the disadvantages of using logic-generated control signals:

- More resources are required (logic cells, interconnect).
- More data skew is introduced.
- If the logic-generated control signals have high fan-out, the design can be more difficult to fit.
By default, the Quartus II software uses dedicated inputs for global control signals automatically. You can assign control signals to dedicated input pins in one of the following ways:

- In the Assignment Editor, select one of the two following methods:
  - Assign pins to dedicated pin locations.
  - Assign a *Global Signal* setting to the pins.

- On the Assignments menu, click *Settings*. On the *Analysis & Synthesis Settings* page, in the *Auto Global Options* section, in the *Category* list, select *Register Control Signals*.

- Insert a *GLOBAL* primitive after the pins.

- If you have already assigned pins for the design in the MAX+PLUS® II software, on the Assignments menu, click *Import Assignments*.

**Reserve Device Resources**

Because pin and logic option assignments can be necessary for board layout and performance requirements, and because full utilization of the device resources can increase the difficulty of fitting the design, Altera recommends that you leave 10% of the device’s logic cells and 5% of the I/O pins unused to accommodate future design modifications. Following the Altera-recommended device resource reservation guidelines for macrocell-based CPLDs increases the chance that the Quartus II software can fit the design during recompilation after changes or assignments have been made.

**Pin Assignment Guidelines and Procedures**

Sometimes user-specified pin assignments are necessary for board layout. This section discusses pin assignment guidelines and procedures.

To minimize fitting issues with pin assignments, follow these guidelines:

- Assign speed-critical control signals to dedicated inputs.
- Assign output enables to appropriate locations.
- Estimate fan-in to assign output pins to the appropriate LAB.
- Assign output pins that require parallel expanders to macrocells numbered 4 to 16.

Altera recommends that you allow the Quartus II software to select pin assignments automatically when possible. You can use the Quartus II Pin Advisor feature (accessible from the Tools menu) for pin connection guidelines. For more information about the Pin Advisor, refer to Quartus II Help.

**Control Signal Pin Assignments**

Assign speed-critical control signals to dedicated input pins. Every MAX 7000 and MAX 3000 device has four dedicated input pins (*GCLK1*, *OE2/GCLK2*, *OE1*, and *GCLRn*). You can assign clocks to global clock dedicated inputs (*GCLK1* and *OE2/GCLK2*), clear to the global clear dedicated input (*GCLRn*), and speed-critical output enable to global OE dedicated inputs (*OE1* and *OE2/GCLK2*).
Output Enable Pin Assignments
Occasionally, because the total number of required output enable pins is more than the dedicated input pins, output enable signals must be assigned to I/O pins.

To minimize possible fitting errors when assigning the output enable pins for MAX 7000 and MAX 3000 devices, refer to Pin-Out Files for Altera Devices on the Altera website (www.altera.com).

Estimate Fan-In When Assigning Output Pins
Macrocells with high fan-in can cause more placement problems for the Quartus II Fitter than those with low fan-in. The maximum fan-in per LAB should not exceed 36 in MAX 7000 and MAX 3000 devices. Therefore, estimate the fan-in of logic (such as an \(x\)-input AND gate) that feeds each output pin. If the total fan-in of logic that feeds each output pin in the same LAB exceeds 36, compilation can fail. To save resources and prevent compilation errors, avoid assigning pins that have high fan-in.

Outputs Using Parallel Expander Pin Assignments
Figure 10–12 illustrates how parallel expanders are used within a LAB. MAX 7000 and MAX 3000 devices contain chains that can lend or borrow parallel expanders. The Quartus II Fitter places macrocells in a location that allows them to lend and borrow parallel expanders appropriately.

As shown in Figure 10–12, only macrocells 2 through 16 can borrow parallel expanders. Therefore, assign output pins that might require parallel expanders to pins adjacent to macrocells 4 through 16. Altera recommends using macrocells 4 through 16 because they can borrow the largest number of parallel expanders.
Resolving Resource Utilization Problems

Two common Quartus II compilation fitting issues cause errors: excessive macrocell usage and lack of routing resources. Macrocell usage errors occur when the total number of macrocells in the design exceed the available macrocells in the device. Routing errors occur when the available routing resources are insufficient to implement the design. Check the Message window for the compilation results.

Messages in the Messages window are also copied in the Report Files. Right-click on a message and click Help for more information.

Resolving Macrocell Usage Issues

Occasionally, a design requires more macrocell resources than are available in the selected device, which results in the design not fitting. The following list provides tips for resolving macrocell usage issues as well as tips to minimize the number of macrocells used.

- On the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and turn off Auto Parallel Expanders. If the design’s clock frequency (f_{MAX}) is not an important design requirement, turn off parallel expanders for all or part of the project. The design usually requires more macrocells if parallel expanders are turned on.

Figure 10–12. LAB Macrocells and Parallel Expander Associations

- Macrocell 1 cannot borrow any parallel expanders.
- Macrocell 3 borrows up to ten parallel expanders from Macrocells 1 and 2.
- Macrocell 2 borrows up to five parallel expanders from Macrocell 1.
- Macrocells 4 through 16 borrow up to 15 parallel expanders from the three immediately-preceding macrocells.
- Change **Optimization Technique** from **Speed** to **Area**. Selecting **Area** instructs the compiler to give preference to area utilization rather than speed \( f_{\text{MAX}} \). On the Assignments menu, click **Settings**. In the **Category** list, change the **Optimization Technique** option in the **Analysis & Synthesis Settings** page.

- Use D-type flipflops instead of latches. Altera recommends that you always use D-type flipflops instead of latches in your design because D-type flipflops can reduce the macrocell fan-in, and thus reduce macrocell usage. The Quartus II software uses extra logic to implement latches in MAX 7000 and MAX 3000 designs because MAX 7000 and MAX 3000 macrocells contain D-type flipflops instead of latches.

- Use asynchronous clear and preset instead of synchronous clear and preset. To reduce the product term usage, use asynchronous clear and preset in your design whenever possible. Using other control signals such as synchronous clear produces macrocells and pins with higher fan-out.

After following the suggestions in this section, if your project still does not fit the targeted device, consider using a larger device. When upgrading to a different density, the vertical package-migration feature of the MAX 7000 and MAX 3000 device families allows pin assignments to be maintained.

**Resolving Routing Issues**

Routing is another resource that can cause design fitting issues. For example, if the total fan-in into a LAB exceeds the maximum allowed, a no-fit error can occur during compilation. If your design does not fit the targeted device because of routing issues, consider the following suggestions.

- Use dedicated inputs/global signals for high fan-out signals. The dedicated inputs in MAX 7000 and MAX 3000 devices are designed for speed-critical and high fan-out signals. Always assign high fan-out signals to dedicated inputs/global signals.

- Change the **Optimization Technique** option from **Speed** to **Area**. This option can resolve routing resource and macrocell usage issues. Refer to “Resolving Macrocell Usage Issues” on page 10–57.

- Reduce the fan-in per cell. If you are not limited by the number of macrocells used in the design, you can use the **Fan-in per cell (%)** option to reduce the fan-in per cell. The allowable values are 20–100%; the default value is 100%. Reducing the fan-in can reduce localized routing congestion but increase the macrocell count. You can set this logic option in the Assignment Editor or under **More Settings** in the **Analysis & Synthesis Settings** page of the **Settings** dialog box.

- On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings**, and turn off **Auto Parallel Expanders**. By turning off the parallel expanders, you give the Quartus II software more fitting flexibility for each macrocell, allowing macrocells to be relocated. For example, each macrocell (previously grouped together in the same LAB) can be moved to a different LAB to reduce routing constraints.
Insert logic cells. Inserting logic cells reduces fan-in and shared expanders used per macrocell, increasing routability. By default, the Quartus II software automatically inserts logic cells when necessary. Otherwise, Auto Logic Cell can be disabled as follows. On the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings. Under More Settings, turn off Auto Logic Cell Insertion. Refer to “Using LCELL Buffers to Reduce Required Resources” for more information.

Change pin assignments. If you want to discard your pin assignments, you can let the Quartus II Fitter ignore some or all of the assignments.

If you prefer reassigning pins to increase routing efficiency, refer to “Pin Assignment Guidelines and Procedures” on page 10–55.

Using LCELL Buffers to Reduce Required Resources

Complex logic, such as multilevel XOR gates, are often implemented with more than one macrocell. When this occurs, the Quartus II software automatically allocates shareable expanders—or additional macrocells (called synthesized logic cells)—to supplement the logic resources that are available in a single macrocell. You can also break down complex logic by inserting logic cells in the project to reduce the average fan-in and the total number of shareable expanders required. Manually inserting logic cells can provide greater control over speed-critical paths.

Instead of using the Quartus II software’s Auto Logic Cell Insertion option, you can manually insert logic cells. However, Altera recommends that you use the Auto Logic Cell Insertion option unless you know which part of the design is causing the congestion.

A good location to manually insert LCELL buffers is where a single complex logic expression feeds multiple destinations in your design. You can insert an LCELL buffer just after the complex expression; the Quartus II Fitter extracts this complex expression and places it in a separate logic cell. Rather than duplicate all the logic for each destination, the Quartus II software feeds the single output from the logic cell to all destinations.

To reduce fan-in and prevent no-fit compilations caused by routing resource issues, insert an LCELL buffer after a NOR gate (Figure 10–13). The design in Figure 10–13 was compiled for a MAX 7000AE device. Without the LCELL buffer, the design requires two macrocells and eight shareable expanders, and the average fan-in is 14.5 macrocells. However, with the LCELL buffer, the design requires three macrocells and eight shareable expanders, and the average fan-in is just 6.33 macrocells.
Timing Optimization Techniques (Macrocell-Based CPLDs)

After resource optimization, design optimization focuses on timing. Ensure that you have made the appropriate assignments as described in “Initial Compilation: Required Settings” on page 10–3, and that the resource utilization is satisfactory before proceeding with timing optimization.

The following five timing parameters are primarily responsible for a design’s performance:

- Setup time ($t_{SU}$) — the propagation time for input data signals
- Hold time ($t_H$) — the propagation time for input data signals
- Clock-to-output time ($t_{CO}$) — the propagation time for output signals
- Pin-to-pin delays ($t_{PD}$) — the time required for a signal from an input pin to propagate through combinational logic and appear at an external output pin
- Maximum clock frequency ($f_{MAX}$) — the internal register-to-register performance

This section provides guidelines to improve the timing if the timing requirements are not met. Figure 10–14 shows the parts of the design that determine the $t_{SU}$, $t_H$, $t_{CO}$, $t_{PD}$, and $f_{MAX}$ timing parameters.
Timing results for $t_{SU}$, $t_{HU}$, $t_{CO}$, $t_{PD}$, and $f_{MAX}$ are found in the Compilation Report for the Quartus II Classic Timing Analyzer, as discussed in “Design Analysis” on page 10–11.

When you are analyzing a design to improve performance, be sure to consider the two major contributors to long delay paths:

- Excessive levels of logic
- Excessive loading (high fan-out)

When a MAX 7000 or MAX 3000 device signal drives more than one LAB, the programmable interconnect array (PIA) delay increases by 0.1 ns per additional LAB fan-out. Therefore, to minimize the added delay, concentrate the destination macrocells into fewer LABs, minimizing the number of LABs that are driven. The main cause of long delays in circuit design is excessive levels of logic.

### Improving Setup Time

Sometimes the $t_{SU}$ timing reported by the Quartus II Fitter does not meet your timing requirements. To improve the $t_{SU}$ timing, refer to the following guidelines:

- Turn on the **Fast Input Register** option using the Assignment Editor. The **Fast Input Register** option allows input pins to directly drive macrocell registers via the fast-input path, thus minimizing the pin-to-register delay. This option is useful when a pin drives a D-type flipflop and there is no combinational logic between the pin and the register.

- Reduce the amount of logic between the input and the register. Excessive logic between the input pin and register causes more delays. To improve setup time, Altera recommends that you reduce the amount of logic between the input pin and the register whenever possible.

- Reduce fan-out. The delay from input pins to macrocell registers increases when the fan-out of the pins increases. To improve the setup time, minimize the fan-out.

### Improving Clock-to-Output Time

To improve a design’s clock-to-output time, minimize the register-to-output-pin delay. To improve the $t_{CO}$ timing, refer to the following guidelines:

- Use the global clock. In addition to minimizing the delay from a register to an output pin, minimizing the delay from the clock pin to the register can also improve $t_{CO}$ timing. Always use the global clock for low-skew and speed-critical signals.
Reduce the amount of logic between the register and output pin. Excessive logic between the register and the output pin causes more delay. Always minimize the amount of logic between the register and output pin for faster clock-to-output time.

Table 10–6 shows the timing results for an EPM7064AETC100-4 device when a combination of the Fast Input Register option, global clock, and minimal logic is used. When the Fast Input Register option is turned on, the $t_{SU}$ timing is improved ($t_{SU}$ decreases from 1.6 ns to 1.3 ns and from 2.8 ns to 2.5 ns). The $t_{CO}$ timing is improved when the global clock is used for low-skew and speed-critical signals ($t_{CO}$ decreases from 4.3 ns to 3.1 ns). However, if there is additional logic used between the input pin and the register or the register and the output pin, the $t_{SU}$ and $t_{CO}$ delays increase.

### Table 10–6. EPM7064AETC100-4 Device Timing Results

<table>
<thead>
<tr>
<th>Number of Registers</th>
<th>$t_{SU}$ (ns)</th>
<th>$t_{H}$ (ns)</th>
<th>$t_{CO}$ (ns)</th>
<th>Global Clock Used</th>
<th>Fast Input Register Option</th>
<th>D Input Location</th>
<th>Q Output Location</th>
<th>Additional Logic Between:</th>
<th>D Input Location &amp; Register</th>
<th>Register &amp; Q Output Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3</td>
<td>1.2</td>
<td>4.3</td>
<td>—</td>
<td>On</td>
<td>LAB A</td>
<td>LAB A</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>1.6</td>
<td>0.3</td>
<td>4.3</td>
<td>—</td>
<td>Off</td>
<td>LAB A</td>
<td>LAB A</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>2.5</td>
<td>0</td>
<td>3.1</td>
<td>✓</td>
<td>On</td>
<td>LAB A</td>
<td>LAB A</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>2.8</td>
<td>0</td>
<td>3.1</td>
<td>✓</td>
<td>Off</td>
<td>LAB A</td>
<td>LAB A</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>3.6</td>
<td>0</td>
<td>3.1</td>
<td>✓</td>
<td>Off</td>
<td>LAB A</td>
<td>LAB A</td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>2.8</td>
<td>0</td>
<td>7.0</td>
<td>✓</td>
<td>Off</td>
<td>LAB D</td>
<td>LAB A</td>
<td>—</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>16 with the same D and clock inputs</td>
<td>2.8</td>
<td>0</td>
<td>All 6.2</td>
<td>✓</td>
<td>Off</td>
<td>LAB D</td>
<td>LAB A, B</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>32 with the same D and clock inputs</td>
<td>2.8</td>
<td>0</td>
<td>All 6.4</td>
<td>✓</td>
<td>Off</td>
<td>LAB C</td>
<td>LAB A, B, C</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

### Improving Propagation Delay ($t_{PD}$)

Achieving fast propagation delay ($t_{PD}$) timing is required in many system designs. However, if there are long delay paths through complex logic, achieving fast propagation delays can be difficult. To improve your design’s $t_{PD}$, refer to the following guidelines.

On the Assignments menu, click Settings. In the Category list, select Analysis & Synthesis Settings, and turn on Auto Parallel Expanders. Turning on the parallel expanders for individual nodes or sub-designs can increase the performance of complex logic functions. However, if the project’s pin or logic cell assignments use parallel expanders placed physically together with macrocells (which can reduce routability), parallel expanders can cause the Quartus II Fitter to have difficulties finding and optimizing a fit. Additionally, the number of macrocells required to implement the design increases and results in a no-fit error during compilation if the device resources are limited. For more information about turning on the Auto Parallel Expanders option, refer to “Resolving Macrocell Usage Issues” on page 10–57.
Set the Optimization Technique to **Speed**. By default, the Quartus II software sets the Optimization Technique option to **Speed** for MAX 7000 and MAX 3000 devices. Reset the Optimization Technique option to **Speed** only if you previously set it to **Area**. On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings**, and turn on **Speed** under Optimization Technique.

Improving Maximum Frequency ($f_{\text{MAX}}$)

Maintaining the system clock at or above a certain frequency is a major goal in circuit design. For example, if you have a fully synchronous system that must run at 100 MHz, the longest delay path from the output of any register to the inputs of the registers it feeds must be less than 10 ns. Maintaining the system clock speed can be difficult if there are long delay paths through complex logic. Altera recommends that you perform the following guidelines to increase your design’s clock speed ($f_{\text{MAX}}$).

- On the Assignments menu, click **Settings**. In the **Category** list, select **Analysis & Synthesis Settings** and turn on **Auto Parallel Expanders**. Turning on the parallel expanders for individual nodes or subdesigns can increase the performance of complex logic functions. However, if the project’s pin or logic cell assignments use parallel expanders placed physically together with macrocells (which can reduce routability), parallel expanders can cause the Quartus II compiler to have difficulties finding and optimizing a fit. Additionally, the number of macrocells required to implement the design also increases and can result in a no-fit error during compilation if the device’s resources are limited. For more information about using the **Auto Parallel Expanders** option, refer to “Resolving Macrocell Usage Issues” on page 10–57.

- Use global signals or dedicated inputs. Altera MAX 7000 and MAX 3000 devices have dedicated inputs that provide low skew and high speed for high fan-out signals. Minimize the number of control signals in the design and use the dedicated inputs to implement them.

- Set the Optimization Technique to **Speed**. By default, the Quartus II software sets the Optimization Technique option to **Speed** for MAX 7000 and MAX 3000 devices. Reset the Optimization Technique option to **Speed** only if you have previously set it to **Area**. You can reset the Optimization Technique option. In the **Category** list, select **Analysis & Synthesis Settings**, and turn on **Speed** under Optimization Technique.

- Pipeline the design. Pipelining, which increases clock frequency ($f_{\text{MAX}}$), refers to dividing large blocks of combinational logic by inserting registers.

Optimizing Source Code—Pipelining for Complex Register Logic

If the methods described in the preceding sections do not sufficiently improve your results, modify the design at the source to achieve the desired results. Using additional register stages (pipeline registers) consumes more device resources, but it also lowers the propagation delay between registers, allowing you to maintain high system clock speed.

Refer to the application note **AN 584: Timing Closure Methodology for Advanced FPGA Designs** for more information about pipelining registers and other examples of optimizing source code.
Compilation-Time Optimization Techniques

If reducing the compilation time of your design is important, use the techniques in this section. Be aware that reducing compilation time with some of these techniques can reduce the overall quality of results. A Compilation Time Advisor is also available in the Quartus II software, which helps you to reduce the compilation time. You can run the Compilation Time Advisor on the Tools menu by pointing to Advisors and clicking Compilation Time Advisor. You can find all the compilation time optimizing techniques described in this section in the Compilation Time Advisor as well.

If you open the Compilation Time Advisor after compilation, it displays recommendations on settings that can reduce the compilation time. Some of the recommendations from different advisors can contradict each other; Altera recommends evaluating the options, and choosing the settings that best suit your design requirements.

Incremental Compilation

The incremental compilation feature can speed up design iteration time by an average of 60% when making changes to the design and helps you reach design timing closure more efficiently. Using incremental compilation allows you to organize your design into logical and physical partitions for design synthesis and fitting. Design iterations can be made faster by recompiling only a particular design partition and merging results with previous compilation results from other partitions. You can also use physical synthesis optimization techniques for specific design partitions while leaving other modules untouched to preserve performance.

If you are using a third-party synthesis tool, you can create separate atom netlist files for parts of your design that you already have synthesized and optimized so that you update only the parts of the design that change.

Regardless of your synthesis tool, you can use full incremental compilation along with LogicLock regions to preserve placement and routing results for unchanged partitions while working on other partitions. This ability provides the most reduction in compilation time and run-time memory usage because neither synthesis nor fitting is performed for unchanged partitions in the design.

You can also perform a bottom-up compilation in which parts of the design are compiled completely independently in separate Quartus II projects, and then exported into the top-level design. This flow is useful in team-based designs or when incorporating third-party IP.

For information about the full incremental compilation flow in the Quartus II software, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook. For information about creating multiple netlist files in third-party tools for use with incremental compilation, refer to the appropriate chapter in Section III. Synthesis in volume 1 of the Quartus II Handbook.
Use Multiple Processors for Parallel Compilation

The Quartus II software can run some algorithms in parallel to take advantage of multiple processors and reduce compilation time when more than one processor is available. Parallel compilation is turned on by default in the Quartus II software and the software can detect if multiple processors are available. You can also specify the maximum number of processors that the software can use if you want to reserve some of the available processors for other tasks. The Quartus II software supports up to four processors. The software does not necessarily use all the processors that you specify during a given compilation, but it never uses more than the specified number of processors. This allows you to work on other tasks on your computer without it becoming slow or less responsive. For interactive tasks such as word processing, it is typically not necessary to restrict the number of processors in this manner.

By allowing the Quartus II software to use two processors, you can reduce the compilation time by up to 10% on systems with two processing cores and by up to 15% on systems with four cores. With certain design flows in which timing analysis runs alone, using multiple processors can reduce the time required for timing analysis by an average of 12% when using two processors. This reduction can reach an average of 15% when using four processors.

The actual reduction in compilation time depends on the design and on the specific settings used for compilation. For example, compilations with multi-corner optimization turned on benefit more from using multiple processors than do compilations that do not use multi-corner optimization. The runtime requirement is not reduced for some other compilation goals, such as Analysis and Synthesis. The Fitter (quartus_fit), the Classic Timing Analyzer (quartus_tan), and the TimeQuest Timing Analyzer (quartus_sta) stages in the compilation might benefit from the use of multiple processors. The average number of processors used for these stages is shown in the Compilation Report, on the Flow Elapsed Time panel. A more detailed breakdown of processor usage is also shown in the Parallel Compilation panel of the appropriate report, such as the Fit report. This panel is only displayed if parallel compilation is enabled.

This feature is available for Arria GX, Stratix, Cyclone, and MAX II series of devices.

Do not consider processors with Intel Hyper-Threading to be more than one processor. If you have a single processor with Intel Hyper-Threading enabled, you should set the number of processors to one. Altera recommends that you do not use the Intel Hyper-Threading feature for Quartus II compilations, as it can increase runtimes.

Many factors can impact the performance of parallel compilation. For detailed information and instructions that can help improve the performance of this feature, refer to the solution to the problem “How can I improve the compilation time performance of the parallel compilation feature in the Quartus II software?” on the Altera website (www.altera.com).
The Quartus II software can detect the number of processors available on a computer and use up to four processors to reduce compilation time. You can also control the number of processors used during a compilation on a per user basis by performing the following steps:

1. On the Tools menu, click Options. The Options dialog box appears.
2. In the Category list, select Processing. The Processing page appears.
3. Under Parallel compilation, select Use all available processors or specify the Maximum processors allowed for compilation.

The Maximum processors allowed setting is applicable to all your projects unless you override the setting with local project-specific settings.

To control the number of processors used during compilation for a specific project, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
3. Under Parallel compilation, select Use global parallel compilation setting if you want a global setting for parallel compilation. If you want a different option for this project, select the Use all available processors which utilizes all processors. If you do not want to run the compilation on all available processors, select Maximum processors allowed and type in the number of processors to be used for compilation. The default value for the number of processors is 1.

Using multiple processors does not affect the quality of the fit. For a given Fitter seed on a specific design, the fit is exactly the same, regardless of whether the Quartus II software uses one processor or multiple processors. The only difference between such compilations using a different number of processors is the compilation time.

You can also set the number of processors available for Quartus II compilation using the following Tcl command in your script.

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS <value>
```

In this case, `<value>` is an integer from 1 to 4.

If you want the Quartus II software to detect the number of processors and use all of them for running the compilation, use the following Tcl command in your script:

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS ALL
```

**Reduce Synthesis Time and Synthesis Netlist Optimization Time**

You can reduce synthesis time by reducing your use of netlist optimizations and by using incremental compilation (with Netlist Type set to Post-Synthesis) without affecting the Fitter time. For tips for reducing synthesis time when using third-party EDA synthesis tools, refer to your synthesis software’s documentation.
Synthesis Netlist Optimizations

You can use Quartus II integrated synthesis to synthesize and optimize HDL designs, and you can use synthesis netlist optimizations to optimize netlists that were synthesized by third-party EDA software. Using these netlist optimizations can cause the Analysis and Synthesis module to take much longer to run. Read the Analysis and Synthesis messages to find out how much time these optimizations take. The compilation time spent in Analysis and Synthesis is usually small compared to the compilation time spent in the Fitter.

If your design meets your performance requirements without synthesis netlist optimizations, turn off the optimizations to save time. If you require synthesis netlist optimizations to meet performance, you can optimize parts of your design hierarchy separately to reduce the overall time spent in analysis and synthesis.

Check Early Timing Estimation Before Fitting

The Quartus II software can provide an estimate of your timing results after synthesis, before the design is fully processed by the Fitter. In cases where you want a quick estimate of your design results before proceeding with further design or synthesis tasks, this feature can save you significant compilation time. For more information, refer to “Early Timing Estimation” on page 10–7.

To view Early Timing Estimation, perform analysis and synthesis in the Quartus II software, and then on the Processing menu, point to Start, and click Start Early Timing Estimate.

Reduce Placement Time

The time required to place a design depends on two factors: the number of ways the logic in the design can be placed in the device and the settings that control how hard the placer works to find a good placement. You can reduce the placement time in two ways:

- Change the settings for the placement algorithm
- Use incremental compilation to preserve the placement for parts of the design

Sometimes there is a trade-off between placement time and routing time. Routing time can increase if the placer does not run long enough to find a good placement. When you reduce placement time, make sure that it does not increase routing time and negate the overall time reduction.

Fitter Effort Setting

Standard fit takes the most runtime and usually does not yield a better result than Auto Fit. To switch from Standard to Auto Fit, on the Assignments menu, click Settings. In the Category list, select Fitter Settings, and use the Fitter effort setting to shorten runtime by changing the effort level to Auto Fit. If you are certain that your design has only easy-to-meet timing constraints, you can select Fast Fit for an even greater runtime saving.
**Placement Effort Multiplier Settings**

You can control the amount of time the Fitter spends in placement by reducing one aspect of placement effort with the **Placement Effort Multiplier** option. On the Assignments menu, click **Settings**. Select **Fitter Settings**, and click **More Settings**. Under **Existing Option Settings**, select **Placement Effort Multiplier**. The default is 1.0. Legal values must be greater than 0 and can be non-integer values. Numbers between 0 and 1 can reduce fitting time, but also can reduce placement quality and design performance. Numbers higher than 1 increase placement time and placement quality, but can reduce routing time for designs with routing congestion. For example, a value of 4 increases placement time by approximately 2 to 4 times, but might result in better placement, which can result in reduced routing time.

**Final Placement Optimization Levels**

The **Final Placement Optimization Level** option specifies whether the Fitter performs final placement optimizations. This can be set to **Always**, **Never**, and **Automatically**. Performing optimizations can improve register-to-register timing and fitting, but might require longer compilation times. The default setting of **Automatically** can be used with the **Auto Fit** Fitter Effort Level (also the default) to let the Fitter decide whether these optimizations should run based on the routability and timing requirements of the design.

Setting the **Final Placement Optimization Level** to **Never** often reduces your compilation time, but typically affects routability negatively and reduces timing performance.

To change the **Final Placement Optimization Level**, on the Assignments menu, click **Settings**. The **Settings** dialog box appears. From the **Category** list, select **Fitter Settings**, and then click the **More Settings** button. Select **Final Placement Optimization Level**, and then from the list, select the required setting.

**Physical Synthesis Effort Settings**

You can use the physical synthesis options to optimize your post-synthesis netlist and improve your timing performance. These options, which affect placement, can significantly increase compilation time.

If your design meets your performance requirements without physical synthesis options, turn them off to save time. You also can use the **Physical synthesis effort** setting on the **Physical Synthesis Optimizations** page under **Fitter Settings** in the **Category** list to reduce the amount of extra compilation time that these optimizations use. The **Fast** setting directs the Quartus II software to use a lower level of physical synthesis optimization that, compared to the normal level, can cause a smaller increase in design performance.

**Limit to One Fitting Attempt**

This option causes the software to quit after one fitting attempt option, instead of repeating placement and routing with increased effort.

From the Assignments menu, select **Settings**. On the **Fitter Settings** page, turn on **Limit to one fitting attempt**.

For more details about this option, refer to “Limit to One Fitting Attempt” on page 10–9.
Preserving Placement, Incremental Compilation, and LogicLock Regions

Preserving information about previous placements can make future placements faster. The incremental compilation feature provides an easy-to-use methodology for preserving placement results. For more information, refer to “Incremental Compilation” on page 10–64.

Reduce Routing Time

The time required to route a design depends on three factors: the device architecture, the placement of the design in the device, and the connectivity between different parts of the design. Typically, the routing time is not a significant amount of the compilation time. If your design takes a long time to route, perform one or more of the following actions:

- Check for routing congestion
- Let the placer run longer to find a more routable placement
- Use incremental compilation to preserve routing information for parts of your design

Identify Routing Congestion in the Chip Planner

To identify areas of routing congestion in your design, open the Chip Planner. On the Tools menu, click Chip Planner. To view the routing congestion in the Chip Planner, click the Layers icon located next to the Task menu. Under Background Color Map, select Routing Utilization. Routing resource usage above 90% indicates routing congestion. You can change the connections in your design to reduce routing congestion. If the area with routing congestion is in a LogicLock region or between LogicLock regions, change or remove the LogicLock regions and recompile the design. If the routing time remains the same, the time is a characteristic of the design and the placement. If the routing time decreases, consider changing the size, location, or contents of LogicLock regions to reduce congestion and decrease routing time.

For information about identifying areas of congested routing using the Chip Planner tool, refer to the “Viewing Routing Congestion” subsection in the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

Identify Routing Congestion in the Timing Closure Floorplan for Legacy Devices

When using devices from the MAX 3000 and MAX 7000 device families, which are not supported by Chip Planner, you must use the Timing Closure Floorplan to identify areas of routing congestion in your design. To open the Timing Closure Floorplan, on the Assignments menu, click Timing Closure Floorplan, and turn on Show Routing Congestion. This feature is available only when you click Field View on the View menu. Routing resource usage above 90% indicates routing congestion. You can change the connections in your design to reduce routing congestion.

Placement Effort Multiplier Setting

Some designs might be time consuming and difficult to route because the placement is not optimal. In such cases, you can increase the Placement Effort Multiplier to get a better placement. Doing so might increase the placement time, but it can reduce the routing time, and even overall compilation time in some cases.
Preserve Routing Incremental Compilation and LogicLock Regions

Preserving information about the previous routing results for part of the design can reduce future routing time. LogicLock regions used with incremental compilation provides an easy-to-use methodology that preserves placement and routing results. For more information, refer to “Incremental Compilation” on page 10–64 and the references listed in the section.

Setting Process Priority

It might be necessary to reduce the computing resources allocated to the compilation at the expense of increased compilation time. It can be convenient to reduce the resource allocation to the compilation with single processor machines if you also have to run other tasks at the same time.

To run a compilation at a reduced process priority, perform the following steps:

1. On the Tools menu, click Options. The Options dialog box appears.
2. In the Category list, under General, select Processing. The Processing page appears.
3. Turn on Run design processing at a lower priority (recommended for single processor machines).

When you turn on this option, it is applied to all future compilations. Using this option can increase your compilation time.

Other Optimization Resources

The Quartus II software has additional resources to help you optimize your design for resource, performance, compilation time, and power.

Design Space Explorer

The Design Space Explorer (DSE) automates the process of running multiple compilations with different settings. You can use the DSE to try the techniques described in this chapter. The DSE utility helps automate the process of finding the best set of options for your design. The DSE explores the design space by applying various optimization techniques and analyzing the results.

For more information, refer to the Design Space Explorer chapter in volume 2 of the Quartus II Handbook.

Other Optimization Advisors

The Power Optimization Advisor provides guidance for reducing power consumption. In addition, the Incremental Compilation Advisor provides suggestions to improve your results when partitioning your design for a hierarchical or team-based design flow using the Quartus II incremental compilation feature.
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 Quartus II command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

For more information about Tcl scripting, refer to the Tcl Scripting chapter in volume 2 of the Quartus II Handbook. Refer to the Quartus II Settings File Reference Manual for information about all settings and constraints in the Quartus II software. For more information about command-line scripting, refer to the Command-Line Scripting chapter in volume 2 of the Quartus II Handbook.

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>
```

If the `<value>` field includes spaces (for example, “Standard Fit”), the value must be enclosed by straight double quotation marks.

Initial Compilation Settings

The Quartus II Settings File (.qsf) variable name is used 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 10–7 lists the .qsf file variable name and applicable values for the settings discussed in “Initial Compilation: Required Settings” on page 10–3. Table 10–8 shows the list of 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>Device Setting</td>
<td>DEVICE</td>
<td>&lt;device part number&gt;</td>
<td>Global</td>
</tr>
<tr>
<td>Use Smart Compilation</td>
<td>SPEED_DISK_USAGE_TRADEOFF</td>
<td>SMART, NORMAL</td>
<td>Global</td>
</tr>
<tr>
<td>Optimize IOC Register Placement For Timing</td>
<td>OPTIMIZE_IOC_REGISTER_PLLACEMENT_FOR_TIMING</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
</tbody>
</table>
Table 10–7. Initial Compilation Settings (Part 2 of 2)

<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 Hold Timing</td>
<td>OPTIMIZE_HOLD_TIMING</td>
<td>OFF, IO PATHS AND MINIMUM TPD PATHS, ALL PATHS</td>
<td>Global</td>
</tr>
<tr>
<td>Fitter Effort</td>
<td>FITTER_EFFECT</td>
<td>STANDARD FIT, FAST FIT, AUTO FIT</td>
<td>Global</td>
</tr>
</tbody>
</table>

Table 10–8. 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 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>

Resource Utilization Optimization Techniques (LUT-Based Devices)

Table 10–9 lists the .qsf file variable name and applicable values for the settings discussed in “Resource Utilization Optimization Techniques (LUT-Based Devices)” on page 10–19.

Table 10–9. Resource Utilization Optimization Settings (Part 1 of 2)

<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 (1)</td>
<td>AUTO_PACKED_REGISTERS_&lt;device family name&gt;</td>
<td>OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, 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>Physical Synthesis for Combinational Logic for Reducing Area</td>
<td>PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Physical Synthesis for Mapping Logic to Memory</td>
<td>PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Optimization Technique</td>
<td>&lt;device family name&gt;_OPTIMIZATION_TECHNIQUE</td>
<td>AREA, SPEED, BALANCED</td>
<td>Global, Instance</td>
</tr>
<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, MINIMAL BITS, USER-ENCODE</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Preserve Hierarchy</td>
<td>PRESERVE_HIERARCHICAL_BOUNDARY</td>
<td>OFF, RELAXED, FIRM</td>
<td>Instance</td>
</tr>
<tr>
<td>Auto RAM Replacement</td>
<td>AUTO_RAM_RECOGNITION</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
</tbody>
</table>
Table 10–9. Resource Utilization Optimization Settings (Part 2 of 2)

<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 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 4 inclusive, or ALL</td>
<td>Global</td>
</tr>
</tbody>
</table>

Note to Table 10–9:

(1) Allowed values for this setting depend on the device family that is selected.

I/O Timing Optimization Techniques (LUT-Based Devices)

Table 10–10 lists the .qsf file variable name and applicable values for the I/O timing optimization settings.

Table 10–10. 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>

Register-to-Register Timing Optimization Techniques (LUT-Based Devices)

Table 10–11 lists the .qsf file variable name and applicable values for the settings discussed in “Register-to-Register Timing Optimization Techniques (LUT-Based Devices)” on page 10–39.

Table 10–11. Register-to-Register Timing Optimization Settings (Part 1 of 2)

<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>Perform Physical Synthesis for Combinational Logic</td>
<td>PHYSICAL_SYNTHESIS_COMBO_LOGIC</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Perform Register Duplication</td>
<td>PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Perform Register Retiming</td>
<td>PHYSICAL_SYNTHESIS_REGISTER_RETIMING</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Perform Automatic Asynchronous Signal Pipelining</td>
<td>PHYSICAL_SYNTHESISASYNCRONOUS_SIGNAL_PIPELINING</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
</tbody>
</table>
Duplicate Logic for Fan-Out Control

The manual logic duplication option accepts wildcards. This is an easy and powerful duplication technique that you can use without editing your source code. You can use this technique, for example, to make a duplicate of a large fan-out node for all of its destinations in a certain design hierarchy, such as `hierarchy_A`. To make such an assignment with Tcl, use a command similar to Example 10–1.

Example 10–1. Duplication Technique

```
set_instance_assignment -name DUPLICATE_ATOM high_fanout_to_A -from \n high_fanout_node -to *hierarchy_A*
```

Conclusion

Using the recommended techniques described in this chapter can help you close timing quickly on complex designs, reduce iterations by providing more intelligent and better links between analysis and assignment tools, and balance multiple design constraints including multiple clocks, routing resources, and area constraints.

The Quartus II software provides many features to achieve optimal results. Follow the techniques presented in this chapter to efficiently optimize a design for area or timing performance, or to reduce compilation time.

Referenced Documents

This chapter references the following documents:

- **Analyzing and Optimizing the Design Floorplan** chapter in volume 2 of the Quartus II Handbook
- **Analyzing Designs with Quartus II Netlist Viewers** chapter in volume 1 of the Quartus II Handbook
- **Assignment Editor** chapter in volume 2 of the Quartus II Handbook
- **Best Practices for Incremental Compilation Partitions and Floorplan Assignments** chapter in volume 1 of the Quartus II Handbook
- **Command-Line Scripting** chapter in volume 2 of the Quartus II Handbook
- Design Analysis and Engineering Change Management with Chip Planner chapter in volume 3 of the Quartus II Handbook
- Design Recommendations for Altera Devices and the Quartus II Design Assistant chapter in volume 1 of the Quartus II Handbook
- Design Space Explorer chapter in volume 2 of the Quartus II Handbook
- I/O Management chapter in volume 2 of the Quartus II Handbook
- Managing Metastability with the Quartus II Software chapter in the Quartus II Handbook
- Managing Quartus II Projects chapter in volume 2 of the Quartus II Handbook
- Power Optimization chapter in volume 2 of the Quartus II Handbook
- Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook
- Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook
- Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook
- Quartus II Settings File Reference Manual
- Recommended HDL Coding Styles chapter in volume 1 of the Quartus II Handbook
- Switching to the Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook
- Tcl Scripting chapter in volume 2 of the Quartus II Handbook
Document Revision History

Table 10–12 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
</table>
| November 2009 v9.1.0       | ■ Removed unsupported Timing Closure Floorplan references  
                          ■ Removed references to unsupported device families  
                          ■ Added several notes  
                          ■ Minor text edits | Updated for the Quartus II 9.1 software release. |
| March 2009 v9.0.0          | ■ Was chapter 8 in the 8.1.0 release.  
                          ■ Updated the following sections:  
                          → “Timing Analysis with the TimeQuest Timing Analyzer” on page 10–14  
                          → “Perform WYSIWYG Resynthesis with Balanced or Area Setting” on page 10–22  
                          → “Use Physical Synthesis Options to Reduce Area” on page 10–26  
                          → “Metastability Analysis and Optimization Techniques” on page 10–32  
                          → “Use Fast Regional Clock Networks and Regional Clocks Networks” on page 10–39  
                          → “Register-to-Register Timing Optimization Techniques (LUT-Based Devices)” on page 10–40  
                          → “Physical Synthesis Optimizations” on page 10–41  
                          → “Duplicate Logic for Fan-Out Control” on page 10–45  
                          → “LogicLock Assignments” on page 10–49  
                          → “Enable Beneficial Skew Optimization” on page 10–48  
                          → “Use Multiple Processors for Parallel Compilation” on page 10–65  
                          ■ Removed “Analyze Your Design for Megastability”  
                          ■ Updated Table 10–11 and Table 10–9  
                          ■ Removed Tables 8-1, 8-2, 8-3, 8-6, and 8-7 from version 8.1 | Updated for the Quartus II 9.0 software release.  
Added Arria II GX support.  
Reorganized portions of this chapter. |
### Table 10–12. Document Revision History (Part 2 of 2)

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2008 v8.1.0</td>
<td>■ Changed document to 8½” × 11” page size.</td>
<td>Updated for the Quartus II 8.1 software release.</td>
</tr>
<tr>
<td></td>
<td>■ Updated the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Optimizing Your Design” on page 10–2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Timing Requirement Settings” on page 10–4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Optimize Hold Timing” on page 10–8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Limit to One Fitting Attempt” on page 10–9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Auto Fit” on page 10–10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Fast Fit” on page 10–11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Ignored Timing Assignments” on page 10–12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “I/O Timing (Including tPD)” on page 10–13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Register-to-Register Timing” on page 10–14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Timing Analysis with the TimeQuest Timing Analyzer” on page 10–14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Use I/O Assignment Analysis” on page 10–20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Flatten the Hierarchy During Synthesis” on page 10–25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Retarget Memory Blocks” on page 10–25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Use Physical Synthesis Options to Reduce Area” on page 10–26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Increase Placement Effort Multiplier” on page 10–30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Metastability Analysis and Optimization Techniques” on page 10–32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Synthesis Netlist Optimizations and Physical Synthesis Optimizations” on page 10–43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Incremental Compilation” on page 10–65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>〜 “Use Multiple Processors for Parallel Compilation” on page 10–66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Table 10–9 on page 10–73 and Table 10–11 on page 10–75.</td>
<td></td>
</tr>
<tr>
<td>May 2008 v8.0.0</td>
<td>■ Updated links</td>
<td>Changes made to this chapter reflect the software changes made in version 8.0.</td>
</tr>
<tr>
<td></td>
<td>■ Updated Other Optimization Resources</td>
<td>Removed support for Mercury devices. Added information for Stratix IV devices.</td>
</tr>
<tr>
<td></td>
<td>■ Updated Setting Process Priority</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Location Assignment and Back-Annotation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Fitter Effort Setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Synthesis Netlist Optimizations and Physical Synthesis Optimizations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Fast Fit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Added Metastability Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Added Enable Beneficial Skew Optimization and Analyze Your Design for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Metastability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Removed figures from “Optimizing Source Code—Pipelining for Complex</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Register Logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>■ Updated Table 8-5</td>
<td></td>
</tr>
</tbody>
</table>

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

Introduction

The Quartus® II software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing your design’s total power consumption using power-driven synthesis and power-driven place-and-route. This chapter describes the power-driven compilation feature and flow in detail, as well as low power design techniques that can further reduce power consumption in your design. The techniques primarily target Arria® GX, Stratix® and Cyclone® series of devices, and HardCopy® II devices. These devices utilize a low-k dielectric material that dramatically reduces dynamic power and improves performance. Stratix II, Stratix III, and Stratix IV device families include efficient logic structures called adaptive logic modules (ALMs) that obtain maximum performance while minimizing power consumption. Cyclone device families offer the optimal blend of high performance and low power in a low-cost FPGA.

For more information about Stratix IV and Stratix III device architecture, refer to the Stratix IV Device Handbook and Stratix III Device Handbook, respectively.

Altera provides the Quartus II PowerPlay Power Analyzer to aid you during the design process by delivering fast and accurate estimations of power consumption. You can minimize power consumption, while taking advantage of the industry’s leading FPGA performance, by using the tools and techniques described in this chapter.

For more information about the PowerPlay Power Analyzer, refer to the PowerPlay Power Analysis chapter in volume 3 of the Quartus II Handbook.

Total FPGA power consumption is comprised of I/O power, core static power, and core dynamic power. This chapter focuses on design optimization options and techniques that help reduce core dynamic power and I/O power. In addition to these techniques, there are additional power optimization techniques available for Stratix IV and Stratix III devices. These techniques include:

- Selectable Core Voltage (available only for Stratix III devices)
- Programmable Power Technology
- Device Speed Grade Selection

For more information about power optimization techniques available for Stratix III devices, refer to AN 437: Power Optimization in Stratix III FPGAs. For more information about power optimization techniques available for Stratix IV devices, refer to AN 514: Power Optimization in Stratix IV FPGAs.
Power Dissipation

This section describes the sources of power dissipation in Stratix III and Cyclone III devices. You can refine techniques that reduce power consumption in your design by understanding the sources of power dissipation.

Figure 11–1 shows the power dissipation of Stratix III and Cyclone III devices in different designs. All designs were analyzed at a fixed clock rate of 100 MHz and exhibited varied logic resource utilization across available resources.

![Figure 11–1. Average Core Dynamic Power Dissipation](image)

Notes to Figure 11–1:

1. 103 different designs were used to obtain these results.
2. 96 different designs were used to obtain these results.
3. In designs using DSP blocks, DSPs consumed 5% of core dynamic power.

As shown in Figure 11–1, a significant amount of the total power is dissipated in routing for both Stratix III and Cyclone III devices, with the remaining power dissipated in logic, clock, and RAM blocks.

In Stratix and Cyclone device families, a series of column and row interconnect wires of varying lengths provide signal interconnections between logic array blocks (LABs), memory block structures, and digital signal processing (DSP) blocks or multiplier blocks. These interconnects dissipate the largest component of device power.

FPGA combinational logic is another source of power consumption. The basic building block of logic in the latest Stratix series devices is the ALM, and in Cyclone IV GX, Cyclone III and Cyclone II devices, it is the logic element (LE).

For more information about ALMs and LEs in Stratix IV, Stratix III, Stratix II, Cyclone IV GX, Cyclone III, and Cyclone II devices, refer to the respective device handbook.
Memory and clock resources are other major consumers of power in FPGAs. Stratix II devices feature the TriMatrix memory architecture. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are configurable to support many features. Stratix IV and Stratix III TriMatrix on-chip memory is an enhancement based upon the Stratix II FPGA TriMatrix memory and includes three sizes of memory blocks: MLAB blocks, M9K blocks, and M144K blocks. Stratix II, Stratix III, and Stratix IV devices feature Programmable Power Technology, an advanced architecture that enables a smooth tradeoff between speed and power. The core of each Stratix III device is divided into tiles, each of which may be put into a high-speed or low-power mode. The primary benefit of Programmable Power Technology is to reduce static power, with a secondary benefit being a small reduction in dynamic power. Cyclone II devices have 4-Kbit M4K memory blocks, and Cyclone III and Cyclone IV GX devices have 9-Kbit M9K memory blocks.

**Design Space Explorer**

Design Space Explorer (DSE) is a simple, easy-to-use, design optimization utility that is included in the Quartus II software. DSE explores and reports optimal Quartus II software options for your design, targeting either power optimization, design performance, or area utilization improvements. You can use DSE to implement the techniques described in this chapter.

Figure 11–2 shows the DSE user interface. The Settings tab is divided into Project Settings and Exploration Settings.

**Figure 11–2.** Design Space Explorer User Interface
The **Search for Lowest Power** option, under **Exploration Settings**, uses a predefined exploration space that targets overall design power improvements. This setting focuses on applying different options that specifically reduce total design thermal power. You can also set the **Optimization Goal** option for your design using the Advanced tab in the DSE window. You can select your design optimization goal, such as optimize for power, from the list of available settings in the **Optimization Goal** list. The DSE then uses the selection from the **Optimization Goal** list, along with the **Search for Lowest Power** selection, to determine the best compilation results.

By default, the Quartus II PowerPlay Power Analyzer is run for every exploration performed by the DSE when the **Search for Lowest Power** option is selected. This helps you debug your design and determine trade-offs between power requirements and performance optimization.

For more information about the DSE, refer to the *Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*.

### Power-Driven Compilation

The standard Quartus II compilation flow consists of Analysis and Synthesis, placement and routing, Assembly, and Timing Analysis. Power-driven compilation takes place at the Analysis and Synthesis and Place-and-Route stages. Quartus II software settings that control power-driven compilation are located in the **PowerPlay power optimization** list on the **Analysis & Synthesis Settings** page, and **PowerPlay power optimization** on the **Fitter Settings** page. The following sections describe these power optimization options at the Analysis and Synthesis and Fitter levels.

#### Power-Driven Synthesis

Synthesis netlist optimization occurs during the synthesis stage of the compilation flow. The optimization technique makes changes to the synthesis netlist to optimize your design according to the selection of area, speed, or power optimization. This section describes power optimization techniques at the synthesis level.

The **Analysis & Synthesis Settings** page allows you to specify logic synthesis options. The **PowerPlay power optimization** option is available for the Arria GX, Stratix and Cyclone families of devices, and MAX® II devices (Figure 11–3).

To perform power optimization at the synthesis level in the Quartus II software, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Analysis & Synthesis**. The **Analysis & Synthesis** page appears.
3. In the **PowerPlay power optimization** list, select your preferred setting. This option determines how aggressively Analysis and Synthesis optimizes the design for power.
Table 11–1 shows the settings in the **PowerPlay power optimization** list. You can apply these settings on a project or entity level.

### Table 11–1. Optimize Power During Synthesis Options

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>No netlist, placement, or routing optimizations are performed to minimize power.</td>
</tr>
<tr>
<td>Normal compilation</td>
<td>Low compute effort algorithms are applied to minimize power through netlist optimizations as long as they are not expected to reduce design performance.</td>
</tr>
<tr>
<td>Extra effort</td>
<td>High compute effort algorithms are applied to minimize power through netlist optimizations. Max performance might be impacted.</td>
</tr>
</tbody>
</table>

The **Normal compilation** setting is turned on by default. This setting performs memory optimization and power-aware logic mapping during synthesis.

Memory blocks can represent a large fraction of total design dynamic power as described in “Reducing Memory Power Consumption” on page 11–14. Minimizing the number of memory blocks accessed during each clock cycle can significantly reduce memory power. Memory optimization involves effective movement of user-defined read/write enable signals to associated read-and-write clock enable signals for all memory types (Figure 11–4).
Figure 11–4 shows a default implementation of a simple dual-port memory block in which write-clock enable and read-clock enable signals are connected to VCC, making both read-and-write memory ports active during each clock cycle. Memory transformation effectively moves the read-enable and write-enable signals to the respective read-clock enable and write-clock enable signals. By using this technique, memory ports are shut down when they are not accessed. This significantly reduces your design’s memory power consumption. For more information about clock enable signals, refer to “Reducing Memory Power Consumption” on page 11–14. For Stratix IV and Stratix III devices, the memory transformation takes place at the Fitter level by selecting the Normal compilation settings for the power optimization option.

In Stratix III, Cyclone III, and Cyclone IV GX devices, the specified read-during-write behavior can significantly impact the power of single-port and bidirectional dual-port RAMs. It is best to set the read-during-write parameter to “Don’t care” (at the HDL level), as it allows an optimization whereby the read-enable signal can be set to the inversion of the existing write-enable signal (if one exists). This allows the core of the RAM to shut down (that is, not toggle), which saves a significant amount of power.

The other type of power optimization that takes place with the Normal compilation setting is power-aware logic mapping. The power-aware logic mapping reduces power by rearranging the logic during synthesis to eliminate nets with high toggle rates.

The Extra effort setting performs the functions of the Normal compilation setting and other memory optimizations to further reduce memory power by shutting down memory blocks that are not accessed. This level of memory optimization can require extra logic, which can reduce design performance.

The Extra effort setting also performs power-aware memory balancing. Power-aware memory balancing automatically chooses the best memory configuration for your memory implementation and provides optimal power saving by determining the number of memory blocks, decoder, and multiplexer circuits required. If you have not previously specified target-embedded memory blocks for your design’s memory functions, the power-aware balancer automatically selects them during memory implementation.

Figure 11–5 shows an example of a 4k × 4 (4k deep and 4 bits wide) memory implementation in two different configurations using M4K memory blocks available in Stratix II devices. The minimum logic area implementation uses M4K blocks configured as 4k × 1. This implementation is the default in the Quartus II software because it has the minimum logic area (0 logic cells) and the highest speed. However,
all four M4K blocks are active on each memory access in this implementation, which increases RAM power. The minimum RAM power implementation is created by selecting Extra effort in the PowerPlay power optimization list. This implementation automatically uses four M4K blocks configured as 1k × 4 for optimal power saving. An address decoder is implemented by the RAM megafunction to select which of the four M4K blocks should be activated on a given cycle, based on the state of the top two user address bits. The RAM megafunction automatically implements a multiplexer to feed the downstream logic by choosing the appropriate M4K output. This implementation reduces RAM power because only one M4K block is active on any cycle, but it requires extra logic cells, costing logic area and potentially impacting design performance.

There is a trade-off between power saved by accessing fewer memories and power consumed by the extra decoder and multiplexor logic. The Quartus II software automatically balances the power savings against the costs to choose the lowest power configuration for each logical RAM. The benchmark data shows that the power-driven synthesis can reduce memory power consumption by as much as 60% in Stratix devices.

Figure 11–5. 4K × 4 Memory Implementation Using Multiple M4K Blocks

Memory optimization options can also be controlled by the Low_Power_Mode parameter in the Default Parameters page of the Settings dialog box. The settings for this parameter are None, Auto, and ALL. None corresponds to the Off setting in the PowerPlay power optimization list. Auto corresponds to the Normal compilation setting and ALL corresponds to the Extra effort setting, respectively. You can apply PowerPlay power optimization either on a compiler basis or on individual entities. The Low_Power_Mode parameter always takes precedence over the Optimize Power for Synthesis option for power optimization on memory.
You can also set the MAXIMUM_DEPTH parameter manually to configure the memory for low power optimization. This technique is the same as the power-aware memory balancer, but it is manual rather than automatic like the Extra effort setting in the PowerPlay power optimization list. You can set the MAXIMUM_DEPTH parameter for memory modules manually in the megafunction instantiation or in the MegaWizard™ Plug-In Manager for power optimization as described in “Reducing Memory Power Consumption” on page 11–14. The MAXIMUM_DEPTH parameter always takes precedence over the Optimize Power for Synthesis options for power optimization on memory optimization.

**Power-Driven Fitter**

The Fitter Settings page enables you to specify options for fitting (Figure 11–6). The PowerPlay power optimization option is available for Arria GX, Stratix IV, Stratix III, Stratix II, Stratix II GX, Cyclone IV GX, Cyclone III, Cyclone II, HardCopy II, and MAX II devices.

To perform power optimization at the Fitter level, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
2. In the Category list, select Fitter Settings. The Fitter Settings page appears.
3. In the PowerPlay power optimization list, select your preferred setting. This option determines how aggressively the Fitter optimizes the design for power.

**Figure 11–6.** Fitter Settings Page
Table 11–2 lists the settings in the PowerPlay power optimization list. These settings can only be applied on a project-wide basis. The Extra effort setting for the Fitter requires extensive effort to optimize the design for power and can increase the compilation time.

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>No netlist, placement, or routing optimizations are performed to minimize power.</td>
</tr>
<tr>
<td>Normal compilation (Default)</td>
<td>Low compute effort algorithms are applied to minimize power through placement and routing optimizations as long as they are not expected to reduce design performance.</td>
</tr>
<tr>
<td>Extra effort</td>
<td>High compute effort algorithms are applied to minimize power through placement and routing optimizations. Max performance might be impacted.</td>
</tr>
</tbody>
</table>

The Normal compilation setting is selected by default and performs DSP optimization by creating power-efficient DSP block configurations for your DSP functions. For Stratix III devices, this setting, which is based on timing constraints entered for the design, enables the Programmable Power Technology to configure tiles as high-speed mode or low-power mode. Programmable Power Technology is always turned ON even when the OFF setting is selected for the Fitter PowerPlay power optimization option. Tiles are the combination of LAB and MLAB pairs (including the adjacent routing associated with LAB and MLAB), which can be configured to operate in high-speed or low-power mode. This level of power optimization does not have any affect on the fitting, timing results, or compile time. Also, for Stratix III devices, this setting enables the memory transformation as described in “Power-Driven Synthesis” on page 11–4.

For more information about Stratix III power optimization, refer to AN 437: Power Optimization in Stratix III FPGAs. For more information about Stratix IV power optimization, refer to AN 514: Power Optimization in Stratix IV FPGAs.

The Extra effort setting performs the functions of the Normal compilation setting and other place-and-route optimizations during fitting to fully optimize the design for power. The Fitter applies an extra effort to minimize power even after timing requirements have been met by effectively moving the logic closer during placement to localize high-toggling nets, and using routes with low capacitance. However, this effort can increase the compilation time.

The Extra effort setting uses a Signal Activity File (.saf) or Verilog Value Change Dump File (.vcd) that guides the Fitter to fully optimize the design for power, based on the signal activity of the design. The best power optimization during fitting results from using the most accurate signal activity information. Signal activities from full post-fit netlist (timing) simulation provide the highest accuracy because all node activities reflect the actual design behavior, provided that supplied input vectors are representative of typical design operation. If you do not have a .saf file (from simulation or other source), the Quartus II software uses assignments, clock assignments, and vectorless estimation values (PowerPlay Power Analyzer Tool settings) to estimate the signal activities. This information is used to optimize your design for power during fitting. The benchmark data shows that the power-driven Fitter technique can reduce power consumption by as much as 19% in Stratix devices.
Only the **Extra effort** setting in the *PowerPlay power optimization* list for the Fitter option uses the signal activities (from `.vcd` or `.saf` file) during fitting. The settings made in the *PowerPlay Power Analyzer Settings* page in the *Settings* dialog box are used to calculate the signal activity of your design.

For more information about `.saf` and `.vcd` files, and how to create them, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

### Recommended Flow for Power-Driven Compilation

Figure 11–7 shows the recommended design flow to fully optimize your design for power during compilation. This flow utilizes the power-driven synthesis and power-driven Fitter options. On average, you can reduce core dynamic power by 16% with the extra effort synthesis and extra effort fitting settings, as compared to the **Off** settings in both synthesis and Fitter options for power-driven compilation.

#### Figure 11–7. Recommended Flow for Power-Driven Compilation

**Area-Driven Synthesis**

Using area optimization rather than timing or delay optimization during synthesis saves power because you use fewer logic blocks. Using less logic usually means less switching activity. The Quartus II integrated synthesis tool provides **Speed**, **Balanced**, or **Area** for the *Optimization Technique* option. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families). The **Speed Optimization Technique** can increase the resource usage of your design if the constraints are too aggressive, and can also result in increased power consumption.
The benchmark data shows that the area-driven technique can reduce power consumption by as much as 31% in Stratix devices and as much as 15% in Cyclone devices.

**Gate-Level Register Retiming**

You can also use gate-level register retiming to reduce circuit switching activity. Retiming shuffles registers across combinational blocks without changing design functionality. The **Perform gate-level register retiming** option in the Quartus II software enables the movement of registers across combinational logic to balance timing, allowing the software to trade off the delay between timing critical and non-critical timing paths.

Retiming uses fewer registers than pipelining. **Figure 11–8** shows an example of gate-level register retiming, where the 10 ns critical delay is reduced by moving the register relative to the combinational logic, resulting in the reduction of data depth and switching activity.

**Figure 11–8. Gate-Level Register Retiming**

---

Gate-level register retiming makes changes at the gate level. If you are using an atom netlist from a third-party synthesis tool, you must also select the **Perform WYSIWYG primitive resynthesis** option to undo the atom primitives to gates mapping (so that register retiming can be performed), and then to remap gates to Altera® primitives. When using the Quartus II integrated synthesis, retiming occurs during synthesis before the design is mapped to Altera primitives. The benchmark data shows that the combination of WYSIWYG remapping and gate-level register retiming techniques can reduce power consumption by as much as 6% in Stratix devices and as much as 21% in Cyclone devices.

For more information about register retiming, refer to the *Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*. 
Design Guidelines

Several low-power design techniques can reduce power consumption when applied during FPGA design implementation. This section provides detailed design techniques for Stratix III, Stratix II, Cyclone IV GX, Cyclone III, and Cyclone II devices that affect overall design power. The results of these techniques might be different from design to design.

Clock Power Management

Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. Figure 11–1 on page 11–2 shows a 14% average contribution to power consumption for global clock routing in Stratix III devices and 16% in Cyclone III devices. Actual clock-related power consumption is higher than this because the power consumed by local clock distribution within logic, memory, and DSP or multiplier blocks is included in the power consumption for the respective blocks.

Clock routing power is automatically optimized by the Quartus II software, which only enables those portions of the clock network that are required to feed downstream registers. Power can be further reduced by gating clocks when they are not required. It is possible to build clock-gating logic, but this approach is not recommended because it is difficult to generate a glitch-free clock in FPGAs using ALMs or LEs.

Arria GX, Stratix IV, Stratix III, Stratix II, Cyclone IV GX, Cyclone III, and Cyclone II devices use clock control blocks that include an enable signal. A clock control block is a clock buffer that lets you dynamically enable or disable the clock network and dynamically switch between multiple sources to drive the clock network. You can use the Quartus II MegaWizard Plug-In Manager to create this clock control block with the ALTCLKCTRL megafunction. Arria GX, Stratix IV, Stratix III, Stratix II, Cyclone IV GX, Cyclone III, and Cyclone II devices provide clock control blocks for global clock networks. In addition, Stratix IV, Stratix III and Stratix II devices have clock control blocks for regional clock networks. The dynamic clock enable feature lets internal logic control the clock network. When a clock network is powered down, all the logic fed by that clock network does not toggle, thereby reducing the overall power consumption of the device. Figure 11–9 shows a 4-input clock control block diagram.

Figure 11–9. Clock Control Block Diagram

The enable signal is applied to the clock signal before being distributed to global routing. Therefore, the enable signal can either have a significant timing slack (at least as large as the global routing delay) or it can reduce the $f_{\text{MAX}}$ of the clock signal.

For more information about using clock control blocks, refer to the Clock Control Block Megafunction User Guide (ALTCLKCTRL).
Another contributor to clock power consumption is the LAB clock that distributes a clock to the registers within a LAB. LAB clock power can be the dominant contributor to overall clock power. For example, in Cyclone III and Cyclone II devices, each LAB can use two clocks and two clock enable signals, as shown in Figure 11–10. Each LAB’s clock signal and clock enable signal are linked. For example, an LE in a particular LAB using the \texttt{labclk1} signal also uses the \texttt{labclkena1} signal.

To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB-level. All registers within an LAB that share a common clock and clock enable are controlled by a shared gated clock. To take advantage of these clock enables, use a clock enable construct in the relevant HDL code for the registered logic.

**LAB-Wide Clock Enable Example**

The VHDL code in Example 11–1 makes use of a LAB-wide clock enable. This clock-gating logic is automatically turned into an LAB-level clock enable signal.

```
IF clk'event AND clock = '1' THEN
  IF logic_is_enabled = '1' THEN
    reg <= value;
  ELSE
    reg <= reg;
  END IF;
END IF;
```

For more information about LAB-wide control signals, refer to the *Stratix II Architecture*, *Cyclone III Device Family Overview*, or *Cyclone II Architecture* chapters in the respective device handbook.
Reducing Memory Power Consumption

The memory blocks in FPGA devices can represent a large fraction of typical core dynamic power. Memory represents 21% of the core dynamic power in a typical Stratix III device design and 20% in a Cyclone III device design. Memory blocks are unlike most other blocks in the device because most of their power is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines.

When a memory block is clocked, there is a sequence of timed events that occur within the block to execute a read or write. The circuitry controlled by the clock consumes the same amount of power regardless of whether or not the address or data has changed from one cycle to the next. Thus, the toggle rate of input data and the address bus have no impact on memory power consumption.

The key to reducing memory power consumption is to reduce the number of memory clocking events. You can achieve this through clock network-wide gating described in “Clock Power Management” on page 11–12, or on a per-memory basis through use of the clock enable signals on the memory ports. Figure 11–11 shows the logical view of the internal clock of the memory block. Use the appropriate enable signals on the memory to make use of the clock enable signal instead of gating the clock.

Figure 11–11. Memory Clock Enable Signal

Using the clock enable signal enables the memory only when necessary and shuts it down for the rest of the time, reducing the overall memory power consumption. You can use the MegaWizard Plug-In Manager to create these enable signals by selecting the Clock enable signal option for the appropriate port when generating the memory block function (Figure 11–12).
For example, consider a design that contains a 32-bit-wide M4K memory block in ROM mode that is running at 200 MHz. Assuming that the output of this block is only required approximately every four cycles, this memory block will consume 8.45 mW of dynamic power according to the demands of the downstream logic. By adding a small amount of control logic to generate a read clock enable signal for the memory block only on the relevant cycles, the power can be cut 75% to 2.15 mW.

You can also use the \texttt{MAXIMUM\_DEPTH} parameter in your memory megafunction to save power in Stratix IV, Stratix III, Stratix II, Cyclone IV GX, Cyclone III, and Cyclone II devices; however, this approach might increase the number of LEs required to implement the memory and affect design performance.

You can set the \texttt{MAXIMUM\_DEPTH} parameter for memory modules manually in the megafunction instantiation or in the MegaWizard Plug-In Manager (Figure 11–13). The Quartus II software automatically chooses the best design memory configuration for optimal power, as described in “Power-Driven Compilation” on page 11–4.
Memory Power Reduction Example

Table 11–3 shows power usage measurements for a 4K × 36 simple dual-port memory implemented using multiple M4K blocks in a Stratix II EP2S15 device. For each implementation, the M4K blocks are configured with a different memory depth.

<table>
<thead>
<tr>
<th>M4K Configuration</th>
<th>Number of M4K Blocks</th>
<th>ALUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K × 1 (Default setting)</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>2K × 2</td>
<td>36</td>
<td>40</td>
</tr>
<tr>
<td>1K × 4</td>
<td>36</td>
<td>62</td>
</tr>
<tr>
<td>512 × 9</td>
<td>32</td>
<td>143</td>
</tr>
<tr>
<td>256 × 18</td>
<td>32</td>
<td>302</td>
</tr>
<tr>
<td>128 × 36</td>
<td>32</td>
<td>633</td>
</tr>
</tbody>
</table>

Figure 11–14 shows the amount of power saved using the MAXIMUM_DEPTH parameter. For all implementations, a user-provided read enable signal is present to indicate when read data is required. Using this power-saving technique can reduce power consumption by as much as 60%.
As the memory depth becomes more shallow, memory dynamic power decreases because unaddressed M4K blocks can be shut off using a decoded combination of address bits and the read enable signal. For a 128-deep memory block, power used by the extra LEs starts to outweigh the power gain achieved by using a more shallow memory block depth. The power consumption of the memory blocks and associated LEs depends on the memory configuration.

**Pipelining and Retiming**

Designs with many glitches consume more power because of faster switching activity. Glitches cause unnecessary and unpredictable temporary logic switches at the output of combinational logic. A glitch usually occurs when there is a mismatch in input signal timing leading to unequal propagation delay.

For example, consider an input change on one input of a 2-input XOR gate from 1 to 0, followed a few moments later by an input change from 0 to 1 on the other input. For a moment, both inputs become 1 (high) during the state transition, resulting in 0 (low) at the output of the XOR gate. Subsequently, when the second input transition takes place, the XOR gate output becomes 1 (high). During signal transition, a glitch is produced before the output becomes stable, as shown in Figure 11–15. This glitch can propagate to subsequent logic and create unnecessary switching activity, increasing power consumption. Circuits with many XOR functions, such as arithmetic circuits or cyclic redundancy check (CRC) circuits, tend to have many glitches if there are several levels of combinational logic between registers.

Pipelining can reduce design glitches by inserting flipflops into long combinational paths. Flipflops do not allow glitches to propagate through combinational paths. Therefore, a pipelined circuit tends to have less glitching. Pipelining has the additional benefit of generally allowing higher clock speed operations, although it does increase the latency of a circuit (in terms of the number of clock cycles to a first result). Figure 11–16 shows an example where pipelining is applied to break up a long combinational path.
Pipelining is very effective for glitch-prone arithmetic systems because it reduces switching activity, resulting in reduced power dissipation in combinational logic. Additionally, pipelining allows higher-speed operation by reducing logic-level numbers between registers. The disadvantage of this technique is that if there are not many glitches in your design, pipelining can increase power consumption by adding unnecessary registers. Pipelining can also increase resource utilization. The benchmark data shows that pipelining can reduce dynamic power consumption by as much as 31% in Stratix devices and as much as 30% in Cyclone devices.

**Architectural Optimization**

You can use design-level architectural optimization by taking advantage of specific device architecture features. These features include dedicated memory and DSP or multiplier blocks available in FPGA devices to perform memory or arithmetic-related functions. You can use these blocks in place of LUTs to reduce power consumption. For example, you can build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

The Stratix device family allows you to efficiently target small, medium, and large memories with the TriMatrix memory architecture. Each TriMatrix memory block is optimized for a specific function. The M512 memory blocks available in Stratix II devices are useful for implementing small FIFO buffers, DSP, and clock domain transfer applications. M512 memory blocks are more power-efficient than the distributed memory structures in some competing FPGAs. The M4K memory blocks are used to implement buffers for a wide variety of applications, including processor code storage, large look-up table implementation, and large memory applications. The M-RAM blocks are useful in applications where a large volume of data must be stored on-chip. Effective utilization of these memory blocks can have a significant impact on power reduction in your design.

The latest Stratix and Cyclone device families have configurable M9K memory blocks that provide various memory functions such as RAM, FIFO buffers, and ROM.
For more information about using DSP and memory blocks efficiently, refer to the *Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*.

### I/O Power Guidelines

Non-terminated I/O standards such as LVTTL and LVCMOS have a rail-to-rail output swing. The voltage difference between logic-high and logic-low signals at the output pin is equal to the $V_{CCIO}$ supply voltage. If the capacitive loading at the output pin is known, the dynamic power consumed in the I/O buffer can be calculated as shown in Equation 11-1:

**Equation 11-1.** Capacitive loading at the output pin

$$ P = 0.5 \times F \times C \times V^2 $$

In this equation, $F$ is the output transition frequency and $C$ is the total load capacitance being switched. $V$ is equal to $V_{CCIO}$ supply voltage. Because of the quadratic dependence on $V_{CCIO}$, lower voltage standards consume significantly less dynamic power. In addition, lower pin capacitance is an important factor in considering I/O power consumption. Hardware and simulation data show that Stratix II device I/O pins have half the pin capacitance of the nearest competing FPGA. Cyclone II devices exhibit 20% less I/O power consumption than competitive, low-cost, 90 nm FPGAs.

Transistor-to-transistor logic (TTL) I/O buffers consume very little static power. As a result, the total power consumed by a LVTTL or LVCMOS output is highly dependent on load and switching frequency.

When using resistively terminated I/O standards like SSTL and HSTL, the output load voltage swings by a small amount around some bias point. The same dynamic power equation is used, where $V$ is the actual load voltage swing. Because this is much smaller than $V_{CCIO}$, dynamic power is lower than for non-terminated I/O under similar conditions. These resistively terminated I/O standards dissipate significant static (frequency-independent) power, because the I/O buffer is constantly driving current into the resistive termination network. However, the lower dynamic power of these I/O standards means they often have lower total power than LVCMOS or LVTTL for high-frequency applications. Use the lowest drive strength I/O setting that meets your speed and waveform requirements to minimize I/O power when using resistively terminated standards.

You can save a small amount of static power by connecting unused I/O banks to the lowest possible $V_{CCIO}$ voltage of 1.2 V.

Table 11-4 shows the total supply and thermal power consumed by outputs using different I/O standards for Stratix II devices. The numbers are for an I/O pin transmitting random data clocked at 200 MHz with a 10 pF capacitive load.

For this configuration, non-terminated standards generally use less power, but this is not always the case. If the frequency or the capacitive load is increased, the power consumed by non-terminated outputs increases faster than the power of terminated outputs.
For more information about I/O Standards, refer to the *Selectable I/O Standards in Stratix II Devices and Stratix II GX Devices* chapter in volume 2 of the *Stratix II Device Handbook* or the *Selectable I/O Standards in Cyclone II Devices* chapter in the *Cyclone II Device Handbook*, or the *Cyclone III Device Handbook*, or the *Cyclone IV GX Handbook*.

When calculating I/O power, the PowerPlay Power Analyzer uses the default capacitive load set for the I/O standard in the *Capacitive Loading* tab of the *Device & Pin Options* dialog box. For Stratix II devices, if *Enable Advanced I/O Timing* is turned on, I/O power is measured using an equivalent load calculated as the sum of the near capacitance, the transmission line distributed capacitance, and the far-end capacitance as defined in the *Board Trace Model* tab of the *Device & Pin Options* dialog box or the Board Trace Model view in the Pin Planner. Any other components defined in the board trace model are not taken into account for the power measurement.

For Stratix IV, Stratix III, Cyclone IV GX, and Cyclone III devices, advanced I/O power, which uses the full board trace model, is always used.

For information about using Advanced I/O Timing and configuring a board trace model, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Handbook*.

### Dynamically-Controlled On-Chip Terminations

Stratix IV and Stratix III FPGAs offer dynamic on-chip termination (OCT). Dynamic OCT enables series termination (RS) and parallel termination (RT) to dynamically turn on/off during the data transfer. This feature is especially useful when Stratix IV and Stratix III FPGAs are used with external memory interfaces, such as interfacing with DDR memories.
Compared to conventional termination, dynamic OCT reduces power consumption significantly as it eliminates the constant DC power consumed by parallel termination when transmitting data. Parallel termination is extremely useful for applications that interface with external memories where I/O standards, such as HSTL and SSTL, are used. Parallel termination supports dynamic OCT, which is useful for bidirectional interfaces (see Figure 11–17).

The following is an example of power saving for a DDR3 interface using on-chip parallel termination.

The static current consumed by parallel OCT is equal to the $V_{CCIO}$ voltage divided by 100 $\Omega$. For DDR3 interfaces that use SSTL-15, the static current is $1.5 \, V / 100 \, \Omega = 15 \, mA$ per pin. Therefore, the static power is $1.5 \, V \times 15 \, mA = 22.5 \, mW$. For an interface with 72 DQ and 18 DQS pins, the static power is $90 \, pins \times 22.5 \, mW = 2.025 \, W$.

Dynamic parallel OCT disables parallel termination during write operations, so if writing occurs 50% of the time, the power saved by dynamic parallel OCT is $50% \times 2.025 \, W = 1.0125 \, W$.

For more information about dynamic OCT in Stratix IV and Stratix III devices, refer to the Stratix III Device I/O Features chapter in the Stratix III Device Handbook and the Stratix IV Device I/O Features chapter in the Stratix IV Device Handbook, respectively.

**Power Optimization Advisor**

The Quartus II software includes the Power Optimization Advisor, which provides specific power optimization advice and recommendations based on the current design project settings and assignments. The advisor covers many of the suggestions listed in this chapter. The following example shows how to reduce your design power with the Power Optimization Advisor.

**Power Optimization Advisor Example**

After compiling your design, run the PowerPlay Power Analyzer to determine your design power and to see where power is dissipated in your design. Based on this information, you can run the Power Optimization Advisor to implement recommendations that can reduce design power. Figure 11–18 shows the Power Optimization Advisor after compiling a design that is not fully optimized for power.
The Power Optimization Advisor shows the recommendations that can reduce power in your design. The recommendations are split into stages to show the order in which you should apply the recommended settings. The first stage shows mostly CAS setting options that are easy to implement and highly effective in reducing design power. An icon indicates whether each recommended setting is made in the current project. In Figure 11–18, the checkmark icon for Stage 1 shows the recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icon shows the general suggestions. Each recommendation includes the description, summary of the affect of the recommendation, and the action required to make the appropriate setting.

There is a link from each recommendation to the appropriate location in the Quartus II user interface where you can change the setting. You can change the Power-Driven Synthesis setting by clicking Open Settings dialog box - Analysis & Synthesis Settings page (Figure 11–19). The Settings dialog box is shown with the Analysis & Synthesis Settings page selected, where you can change the PowerPlay power optimization settings.
After making the recommended changes, recompile your design. The Power Optimization Advisor indicates with green check marks that the recommendations were implemented successfully (Figure 11–20). You can use the PowerPlay Power Analyzer to verify your design power results.
The recommendations listed in Stage 2 generally involve design changes, rather than CAD settings changes as in Stage 1. You can use these recommendations to further reduce your design power consumption. Altera recommends that you implement Stage 1 recommendations first, then the Stage 2 recommendations.

**Conclusion**

The combination of a smaller process technology, the use of low-k dielectric material, and reduced supply voltage significantly reduces dynamic power consumption in the latest FPGAs. To further reduce your dynamic power, use the design recommendations presented in this chapter to optimize resource utilization and minimize power consumption.

**Referenced Documents**

This chapter references the following documents:

- **Area and Timing Optimization** chapter in volume 2 of the Quartus II Handbook
- **AN 437: Power Optimization in Stratix III FPGAs**
- **AN 514: Power Optimization in Stratix IV FPGAs**
- **Clock Control Block Megafunction User Guide (ALTCLKCTRL)**
- **Cyclone III Device Family Overview** chapter in the Cyclone III Device Handbook
- **Cyclone II Architecture** chapter in the Cyclone II Device Handbook
- **Design Space Explorer** chapter in volume 2 of the Quartus II Handbook
- **I/O Management** chapter in volume 2 of the Quartus II Handbook
- **Netlist Optimizations and Physical Synthesis** chapter in volume 2 of the Quartus II Handbook
- **PowerPlay Power Analysis** chapter in volume 3 of the Quartus II Handbook
- **Stratix II Architecture** chapter in volume 1 in the Stratix II Device Handbook
Chapter 11: Power Optimization

Document Revision History

Table 11–5 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
</table>
| November 2009 v.9.1.0      | ■ Updated Figure 11-1 and associated references.  
■ Updated device support.  
■ Minor editorial updates. | Updated for the Quartus II 9.1 software release. |
| March 2009 v9.0.0          | ■ Was chapter 9 in the 8.1.0 release.  
■ Updated for the Quartus II software release.  
■ Added benchmark results.  
■ Removed several sections.  
■ Updated Figure 11–1, Figure 11–18, Figure 11–19, and Figure 11–20. | Updated for the Quartus II 9.0 software release. |
| November 2008 v8.1.0       | ■ Changed to 8½” × 11” page size.  
■ Changed references to altsyncram to RAM.  
■ Minor editorial updates | Updated for the Quartus II 8.1 software release. |
| May 2008 v8.0.0            | ■ Updated Table 9–1 and 9–9.  
■ Updated “Architectural Optimization” on page 9–22  
■ Added “Dynamically-Controlled On-Chip Terminations” on page 9–26  
■ Updated “Referenced Documents” on page 9–29  
■ Updated references | Added support for Stratix IV devices. |

For previous versions of the Quartus II Handbook, refer to the Quartus II Handbook Archive.
12. Analyzing and Optimizing the Design Floorplan

You can use the Chip Planner to perform design analysis and create a design floorplan. With some of the older device families, you must use the Timing Closure Floorplan to analyze the device floorplan. To make I/O assignments, use the Pin Planner.

Introduction

As FPGA designs grow larger in density, analyzing the design for performance, routing congestion, and logic placement to meet the design requirements becomes critical.

This chapter discusses how to analyze the design floorplan with the Chip Planner and the Timing Closure Floorplan (for supported devices only).

You can use the Design Partition Planner along with the Chip Planner to customize the floorplan for your design. For more information, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design and the Best Practices for Incremental Compilation Partition and Floorplan Assignments chapters in volume 1 of the Quartus II Handbook.

This chapter includes the following topics:

- “Chip Planner Overview” on page 12–2
- “LogicLock Regions” on page 12–6
- “Using LogicLock Regions in the Chip Planner” on page 12–20
- “Design Floorplan Analysis Using the Chip Planner” on page 12–20
- “Design Analysis Using the Timing Closure Floorplan” on page 12–42
- “Scripting Support” on page 12–48

For more information about the Pin Planner, refer to the I/O Management chapter in volume 2 of the Quartus II Handbook.

Table 12–1 lists the device families supported by the Chip Planner and the Timing Closure Floorplan.

Table 12–1. Chip Planner and Timing Closure Floorplan Device Support (Part 1 of 2)

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Timing Closure Floorplan</th>
<th>Chip Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arria® series</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>Cyclone series</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>HardCopy series</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>Stratix series</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>MAX® IIZ</td>
<td>—</td>
<td>✓</td>
</tr>
<tr>
<td>MAX II</td>
<td>—</td>
<td>✓</td>
</tr>
</tbody>
</table>
Chip Planner Overview

The Chip Planner provides a visual display of chip resources. It can show logic placement, LogicLock regions, relative resource usage, detailed routing information, fan-in and fan-out connections between nodes, timing paths between registers, and delay estimates for paths. With the Chip Planner, you can view critical path information, physical timing estimates, and routing congestion.

You can also perform assignment changes with the Chip Planner, such as creating and deleting resource assignments, and post-compilation changes such as creating, moving, and deleting logic cells and I/O atoms. With the Chip Planner and Resource Property Editor, you can change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, and RAM and digital signal processing (DSP) blocks. With the Chip Planner, you can view and create assignments for a design floorplan, perform power and design analyses, and implement ECOs.

For details about how to implement ECOs in your design using the Chip Planner in the Quartus II software, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

Table 12–1. Chip Planner and Timing Closure Floorplan Device Support (Part 2 of 2)

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Timing Closure Floorplan</th>
<th>Chip Planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX 3000</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>MAX 7000</td>
<td>✓</td>
<td>—</td>
</tr>
</tbody>
</table>
Starting the Chip Planner

To start the Chip Planner, on the Tools menu, click Chip Planner (Floorplan & Chip Editor). You can also start the Chip Planner by the following methods:

- Click the Chip Planner icon on the Quartus II software toolbar
- On the Shortcut menu in the following tools, click Locate and then click Chip Planner:
  - Design Partition Planner
  - Compilation Report
  - LogicLock Regions window
  - Technology Map Viewer
  - Project Navigator window
  - RTL source code
  - Node Finder
  - Simulation Report
  - RTL Viewer
  - Report Timing panel of the TimeQuest Timing Analyzer

If the device in your project is not supported by the Chip Planner and you attempt to start the Chip Planner, the following message appears:

Can’t display Chip Planner: the current device family is unsupported.
Use the Timing Closure Floorplan for devices not supported by the Chip Planner.

Chip Planner Toolbar

The Chip Planner gives you powerful capabilities for design analysis with a user-friendly GUI. Many Chip Planner functions are available from the menu items or by clicking the icons on the toolbar. Figure 12–1 shows an example of the Chip Planner toolbar and provides descriptions for commonly used icons located on the Chip Planner toolbar.
You can customize the icons on the Chip Planner toolbar by clicking Customize Chip Planner on the Tools menu (if the Chip Planner window is attached), or by clicking Customize on the Tools menu (if the Chip Planner window is detached).

**Chip Planner Tasks and Layers**

The Chip Planner has predefined tasks that enable you to quickly implement ECO changes or manipulate assignments for the floorplan of the device. To select a task, click on the task name in the Task menu. The predefined tasks in the Chip Planner are:

- **Floorplan Editing (Assignment)**
- **Post-Compilation Editing (ECO)**
- **Partition Display (Assignment)**
- **Partition Planner**
- **Routing Congestion (ECO)**
- **Clock Regions (Assignment)**—available for Arria GX, Arria II GX, Cyclone II, Cyclone III, HardCopy II, HardCopy III, Stratix II, Stratix II GX, Stratix III, and Stratix IV devices only
- **Power Analysis (Assignment)**—available for Stratix III and Stratix IV devices only

In the Chip Planner, layers allow you to specify the graphic elements that are displayed for a given task. You can turn off the display of specific graphic elements to increase the window refresh speed and reduce visual clutter when viewing complex designs. The **Background Color Map** can indicate the **Block Utilization**, **Routing Utilization**, **Physical Timing Estimate**, **I/O Banks**, or the High speed-Low power Tiles. When you select **Design Partition Planner** in the **Background Color Map** settings, the resources used by each partition are displayed in the Chip Planner with the same colors used for these partitions in the Design Partition Planner. For example, **Routing Utilization** indicates the relative routing utilization, and **Physical Timing Estimate** indicates the relative physical timing.

Each predefined task in the Chip Planner has a **Background Color Map**, a set of displayed layers, and an editing mode associated with the task. Click the Layers icon (shown in Figure 12–1) to display the Layers Settings window (Figure 12–2). In this window you can select the layers and background color map for each task.

**Figure 12–2. Layers in the Chip Planner**
The Chip Planner operates in either Assignment or ECO mode. You can perform design analyses in either of these modes. Use the Floorplan Editing (Assignment) task in the Assignment mode to manipulate LogicLock regions and location assignments in your design. The Post Compilation Editing (ECO) task in ECO mode allows you to implement ECO changes in your design. The Partition Display (Assignment) task allows you to view the placement of nodes and color codes the nodes based on their partition. When you select the Clock Regions (Assignment) task, you can see the regions in your device that are driven by global clock networks. The Power Analysis (Assignment) task allows you to view high and low power resources in Stratix III and Stratix IV devices.

For more information about the ECO mode of operation, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

You can also create and save your own custom tasks. When you create a custom task, you can turn on or off any layer by checking the appropriate box located next to each layer. You can also select different Background Color Maps for your custom task. After selecting the required settings, click Save Task As to save your custom task.

### LogicLock Regions

LogicLock regions are regions you define on the device. You can use LogicLock regions to create a floorplan for your design. Your floorplan can contain several LogicLock regions. A LogicLock region is defined by its height, width, and location. You can specify the size or location of a region, or both, or the Quartus II software can generate these properties automatically. The Quartus II software bases the size and location of a region on the contents of the region and the timing requirements of the module. Table 12–2 describes the options for creating LogicLock regions.

**Table 12–2. Types of LogicLock Regions**

<table>
<thead>
<tr>
<th>Properties</th>
<th>Values</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Floating (default), Locked</td>
<td>Floating regions allow the Quartus II software to determine the location of the region on the device. Locked regions are areas that you define and are shown with a solid boundary in the floorplan. A locked region must have a fixed size.</td>
</tr>
<tr>
<td>Size</td>
<td>Auto (default), Fixed</td>
<td>Auto-sized regions allow the Quartus II software to determine the appropriate size of a region given its contents. Fixed regions have a shape and size that you define.</td>
</tr>
<tr>
<td>Reserved</td>
<td>Off (default), On, Limited</td>
<td>The reserved property allows you to define whether the Fitter can use the resources within a region for entities that are not assigned to the region. If the reserved property is turned on, only items assigned to the region can be placed within its boundaries. When you set it to limited, the Fitter does not place any logic from the parent region.</td>
</tr>
<tr>
<td>Origin</td>
<td>Any Floorplan Location</td>
<td>The origin is the origin of the LogicLock region’s placement on the floorplan. For Arria GX, Stratix, and Cyclone series devices, and MAX II devices, the origin is located in the lower left corner. For other Altera® device families, the origin is located in the upper left corner.</td>
</tr>
</tbody>
</table>

The Quartus II software cannot automatically define the size of a region if the location is locked. Therefore, if you want to specify the exact location of the region, you must also specify the size.
You can use the Design Partition Planner in conjunction with LogicLock regions to create a floorplan for your design. For more information about using the Design Partition Planner, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Designs* and the *Best Practices for Incremental Compilation Partition and Floorplan Assignments* chapters in volume 1 of the *Quartus II Handbook*.

**Creating LogicLock Regions**

You can create LogicLock Regions from the Project Navigator, the LogicLock Regions window, or the Chip Planner.

**Creating LogicLock Regions from the Quartus II User Interface**

After you perform either a full compilation or analysis and elaboration on the design, the Quartus II software displays the hierarchy of the design. On the View menu, click **Project Navigator**. With the hierarchy of the design fully expanded, as shown in *Figure 12–3*, right-click on any design entity in the design, and click **Create New LogicLock Region** to create a LogicLock region.

**Figure 12–3. Using the Project Navigator to Create LogicLock Regions**

---

**Placing LogicLock Regions**

A fixed region must contain all resources required for the design block for which you define the region. Although the Quartus II software can automatically place and size LogicLock regions to meet resource and timing requirements, you can manually place and size regions to meet your design requirements. To do so, follow these guidelines:

- Place LogicLock regions with pin assignments on the periphery of the device, adjacent to the pins. For the Arria GX, Stratix, and Cyclone series of devices and MAX II devices, you must also include the I/O block within the LogicLock Region.
Floating LogicLock regions can overlap with their ancestors or descendants, but not with other floating LogicLock regions.

Avoid creating fixed and locked regions that overlap.

If you want to import multiple instances of a module into a top-level design, you must ensure that the device has two or more locations with exactly the same device resources. (You can determine this from the applicable device handbook.) If the device does not have another area with exactly the same resources, the Quartus II software generates a fitting error during compilation of the top-level design.

When you import a LogicLock region, the Quartus II software changes the property to floating and assigns a new unique name. You can change the property to fixed to guarantee the same placement achieved previously. You can import or export LogicLock regions across devices within a family, but not between families.

**Placing Device Features into LogicLock Regions**

A LogicLock region includes all device resources within its boundaries, including memory and pins. You can assign pins to LogicLock regions; however, this placement puts location constraints on the region. When the Quartus II software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the LogicLock region.

Pin assignments to LogicLock regions are effective only in fixed and locked regions. Pin assignments to floating regions do not influence the placement of the region.

Only one LogicLock region can claim a device resource. If the boundary includes part of a device resource, the Quartus II software allocates the entire resource to the LogicLock region.

**LogicLock Regions Window**

The LogicLock window consists of the LogicLock Regions window (Figure 12–4) and the LogicLock Region Properties dialog box. Use the LogicLock Regions window to create LogicLock regions and assign nodes and entities to them. The dialog box provides a summary of all LogicLock regions in your design. In the LogicLock Regions window, you can modify the properties of a LogicLock region such as size, state, width, height, origin, and whether the region is a reserved region. The LogicLock Regions window also has a recommendations toolbar at the bottom. Select a LogicLock region from the drop-down list in the recommendations toolbar to display the relevant suggestions to optimize that LogicLock region.

The origin location varies, depending on the device family. For Arria GX, Cyclone, Stratix, and MAX II devices, the origin of the LogicLock region is located at the lower-left corner of the region. For all other supported devices, the origin is located at the upper-left corner of the region.
You can customize the LogicLock Regions window by dragging and dropping the columns to change their order. Columns can also be hidden.

For designs that target Arria GX, Cyclone, Stratix, and MAX II devices, the Quartus II software automatically creates a LogicLock region that encompasses the entire device. This default region is labelled `Root_region`, and is locked and fixed.

Use the LogicLock Region Properties dialog box to obtain detailed information about your LogicLock region, such as which entities and nodes are assigned to your region and which resources are required. The LogicLock Region Properties dialog box shows the properties of the current selected regions. You can also modify the settings for LogicLock regions in the LogicLock Region Properties dialog box.

To open the LogicLock Region Properties dialog box, double-click any region in the LogicLock Regions window, or right-click the region and click Properties.

**Creating LogicLock Regions with the Chip Planner**

In the View menu of the Chip Planner, click Create LogicLock Region. In the Chip Planner, click and drag to create a region of your preferred location and size.

**Assigning LogicLock Region Content**

After you have created a LogicLock region, you must assign resources to it using the Chip Planner, the LogicLock Regions dialog box, or a Tcl script.

You can drag selected logic displayed in the Hierarchy tab of the Project Navigator, in the Node Finder, or in a schematic design file, and drop it into the Chip Planner or the LogicLock Regions dialog box. Figure 12–5 shows logic that has been dragged from the Hierarchy tab of the Project Navigator and dropped into a LogicLock region in the Chip Planner.
You can also drag logic from the **Hierarchy** tab of the Project Navigator and drop it in the LogicLock Regions **Properties** dialog box. Logic can also be dropped into the **Design Element Assigned** column of the **Contents** tab of the LogicLock Region **Properties** box.

You must assign pins to a LogicLock region manually. The Quartus II software does not include pins automatically when you assign an entity to a region. The software only obeys pin assignments to locked regions that border the periphery of the device. For the Cyclone, Stratix, and MAX II series of devices, the locked regions must include the I/O pins as resources.

**Hierarchical (Parent and Child) LogicLock Regions**

You can define a hierarchy for a group of regions by declaring parent and child regions. The Quartus II software places a child region completely within the boundaries of its parent region, allowing you to further constrain module locations. Additionally, parent and child regions allow you to further improve the performance of a module by constraining the nodes in the critical path of the module.

To make one LogicLock region a child of another LogicLock region, in the LogicLock Regions window, select the new child region and drag and drop it inside its new parent region.

The LogicLock region hierarchy does not have to be the same as the design hierarchy.
You can create both fixed and floating LogicLock regions within a fixed parent LogicLock region. The location of a floating child region can float within its parent. If a child region is fixed, its location remains locked relative to its parent’s origin. A locked parent region’s location is locked relative to the device. If the child’s location is locked and the parent’s location is changed, the child’s origin changes, but maintains the same placement relative to the origin of its parent. Either you or the Quartus II software can determine a child region’s size; however, the child region must fit entirely within the parent region. The levels of hierarchy in LogicLock regions are unlimited, but complicated hierarchical regions might result in some LABs not being utilized; thus, effectively increasing the resource utilization in the device.

**Reserved LogicLock Region**

The Quartus II software honors all entity and node assignments to LogicLock 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, the Quartus II software’s default behavior fills the unoccupied resources with other nodes and entities that have not been assigned to another region. You can prevent this behavior by turning on Reserved on the General tab of the LogicLock Region Properties dialog box. When you turn on this option, your LogicLock region contains only the entities and nodes that you specifically assigned to your LogicLock region. When you set the reserved property for a LogicLock region, the Fitter does not place logic from the immediate parent LogicLock region in the assigned LogicLock area, but it might place logic from other parts of your design in that area.

In a team-based design environment, the Limited option helps you create a device floorplan. When this option is turned on, each team can be assigned a portion of the device floorplan where placement and optimization of each submodule occurs. Device resources can be distributed to each module without affecting the performance of other modules.

**Creating Non-Rectangular LogicLock Regions**

When you create a floorplan for your design, you may want to create non-rectangular LogicLock regions to make some device resources accessible to design blocks outside a LogicLock region. You might also create a non-rectangular LogicLock region to place certain parts of your design around specific device resources to improve performance. You can create non-rectangular LogicLock regions in two ways: with the Merge command in the Chip Planner, or with the reserved property of LogicLock regions.

**Creating Non-Rectangular LogicLock Regions Using the Merge Command**

The Merge command is available for Arria II GX, Cyclone III series, Cyclone IV, HardCopy III, HardCopy IV, Stratix III, and Stratix IV series device families. To create a non-rectangular region with the Merge command, follow these steps:

1. In the Chip Planner, create two or more contiguous or non-contiguous rectangular regions as described in “Creating LogicLock Regions” on page 12–7.
2. Arrange the regions that you have created into the locations where you want the non-rectangular region to be.
3. Select all the individual regions to be merged by clicking each of them while holding the Shift key.

4. Right-click the title bar of any of the LogicLock regions that you want to merge, point to LogicLock regions, and then click Merge. The individual regions that you have selected are now merged to create a single new region.

By default, the new LogicLock region bears the name of the component region containing the greatest number of resources; however, you can rename the new region. In the LogicLock Regions window, the new region is shown as having a custom shape.

Figure 12–6 illustrates two autonomous LogicLock regions combined using the Merge command to form a new non-rectangular region.

**Figure 12–6.** Using the Merge command to create a non-rectangular region

---

**Creating Non-Rectangular Regions Using Reserved LogicLock Regions**

For all devices not supported by the Merge command, you can use the reserved property of LogicLock regions to create regions that are non-rectangular or non-contiguous.

For example, consider a case in which there is one LogicLock region under the Root region and two child regions under this region (Figure 12–7).

**Figure 12–7.** Example 1
You can set the **Reserved** property of a LogicLock region to **On**, **Off**, or **Limited**. If you create a LogicLock region for `Child_Region_1` with its **Reserved** property set to **Limited**, the Fitter does not place nodes that are members of `Parent_Region_1` or `Child_Region_2` into the boundary of `Child_Region_1`. However, if `Child_Region_2` overlaps `Child_Region_1`, then logic can be placed in the overlapping area. The Fitter can also place nodes that are not members of `Parent_Region_1` or `Child_Region_1` (such as members of the `Root_Region`) into `Child_Region_1`. On the other hand, if `Child_Region_1` is set to exclude all non-members, the Fitter can only place nodes that are members of `Child_Region_1` into the region.

If the Parent Region’s reserved property is turned off, then the Fitter might place other logic in the allocated region.

If you want to create a non-rectangular region as shown in Figure 12–8, you can create two rectangular hierarchical LogicLock regions. Turn off the reserved property on the parent LogicLock region and set the reserved property on the child LogicLock region to **Limited** to prevent the Fitter from placing any logic of the module assigned to the parent LogicLock region. Logic that is external to the parent LogicLock region might be placed in the area allocated to the child region. This produces a non-rectangular LogicLock region.

![Non-Rectangular Region](image)

**Examples of Non-Rectangular LogicLock Regions Using Reserved Property**

The following examples use the design hierarchy shown in Figure 12–9.

![An Example Design Hierarchy](image)

**Example 1: Creating an L-Shaped Region**

In the design hierarchy example in Figure 12–9, suppose you want to create an L-shaped region, such that the `Alu1` module is placed completely inside the region, and the non-`Alu1` nodes can be placed anywhere on the chip (as shown in Figure 12–10).
The L-shaped region defines a rectangular region that is carved out by a child LogicLock region (Special) to achieve the L-shape effect. The Reserved property of this child LogicLock region is set to Limited, such that the Fitter does not require logic from members of Alu1 (which is the parent region of the region named Special) inside it while letting other nodes in. Not displayed in Figure 12–10, the Alu1 entity instance is assigned as a member to the L_Shaped region. This effect can be achieved by creating a hierarchical LogicLock region as shown in Figure 12–11.

Figure 12–12 illustrates the expected fitting results with these LogicLock regions. Nodes from the Alu1 entity instance are colored blue, while nodes from the rest of the design are colored red.
Figure 12–12. Expected Fitting Results with LogicLock Regions

Example 2: Region with Disjoint Areas
Suppose you want to create a region consisting of two disjoint rectangles (or any number of disjoint areas), such that the Alu1 module is placed completely inside the region, and the non-Alu1 nodes can be placed anywhere on the chip as shown in Figure 12–13.

Figure 12–13. Region Consisting of Two Disjointed Rectangles
You can achieve a region with disjoint areas using the region hierarchy example in Figure 12–14.

**Figure 12–14. Region with Disjoint Areas**

![Region with Disjoint Areas](image)

The disjoint region defines a rectangular region that is carved out by the Special child region to achieve the disjoint effect. Notice that the Special region is set to reserved “from members of parent region hierarchy” to prevent the Alu1 nodes from being placed inside it, while letting other nodes in. The Alu1 entity instance should be assigned to the Disjoint LogicLock region.

**Figure 12–15 shows the expected fitting results with the LogicLock regions. Nodes from the Alu1 entity instance are colored blue, while nodes from the rest of the design are colored red and brown.**

**Figure 12–15. Expected Fitting Results with the LogicLock Regions**

![Expected Fitting Results with the LogicLock Regions](image)

Hierarchial LogicLock assignments can increase resource usage in the device, because some design blocks might not have access to resources inside the LogicLock regions. When you create hierarchial LogicLock regions to create non-rectangular regions, keep the hierarchy assignments simple, to minimize increase in resource usage.
Excluded Resources

The Excluded Resources feature allows you to easily exclude specific device resources such as DSP blocks or M4K memory blocks from a LogicLock region. For example, you can specify resources that belong to a specific entity that are assigned to a LogicLock region, and specify that these resources be included with the exception of the DSP blocks. Use the Excluded Resources feature on a per-LogicLock region member basis.

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

F The Excluded Resources feature prevents certain resource types from being included in a region, but it does not prevent the resources from being placed inside the region unless the region’s Reserved property is set to On. To indicate to the Fitter that certain resources are not required inside a LogicLock region, define a resource filter.

Additional Quartus II LogicLock Design Features

To complement the LogicLock Regions dialog box, the Quartus II software has additional features to help you design with LogicLock regions.

Tooltips

When you move the mouse pointer over a LogicLock region name on the LogicLock Regions dialog box, or over the top bar of the LogicLock region in the Chip Planner, the Quartus II software displays a tooltip with information about the properties of the LogicLock region.

Analysis and Synthesis Resource Utilization by Entity

The Compilation Report contains an Analysis and Synthesis Resource Utilization by Entity section, which reports accurate resource usage statistics, including entity-level information. You can use this feature when you manually create LogicLock regions.

Path-Based Assignments

You can assign paths to LogicLock regions based on source and destination nodes, allowing you to easily group critical design nodes into a LogicLock region. Any of the following types of nodes can be the source and destination nodes:

- Valid register-to-register path—the source and destination nodes must be registers
- Valid pin-to-register path—the source node is a pin and the destination node is a register
- Valid register-to-pin path—the source node is a register and the destination node is a pin
- Valid pin-to-pin path—both the source and destination nodes are pins
To open the **Paths** dialog box, on the **General** tab of the **Logic Lock Regions** dialog box, click **Add Path**.

Both “*” and “?” wildcard characters are allowed for the source and destination nodes. When creating path-based assignments, you can exclude specific nodes using the **Name exclude** field in the **Paths** dialog box. The Quartus II software ignores all paths passing through the nodes that match the setting in the **Name exclude** field. For example, consider a case with two paths between the source and destination—one passing through node A and the other passing through node B. If you specify node B in the **Name exclude** field, only the path assignment through node A is valid.

You can also use the Quartus II Timing Analysis Report to create path-based assignments by following these steps:

1. Expand the **Timing Analyzer** section in the Compilation Report.
2. Select any of the clocks in the section labeled “Clock Setup:<clock name>.”
3. Locate a path that you want to assign to a LogicLock region. Drag this path from the Report window and drop it in the appropriate row in the LogicLock Region pane in the Quartus II GUI.

This operation creates a path-based assignment from the source register to the destination register, as shown in the Timing Analysis Report.

**Quartus II Revisions Feature**

When you evaluate different LogicLock regions in your design, you might want to experiment with different configurations to achieve your desired results. The Quartus II Revisions feature provides a convenient way to organize the same project with different settings until you find an optimum configuration.

To use the Revisions feature, on the Project menu, click **Revisions**. In the **Revisions** dialog box, you can create and specify revisions. Revision can be based on the current design or any previously created revisions. Each revision can have an associated description. Revisions are a convenient way to organize the placement constraints created for your LogicLock regions.

**LogicLock Assignment Precedence**

Conflicts can arise during the assignment of entities and nodes to LogicLock regions. For example, an entire top-level entity might be assigned to one region and a node within this top-level entity assigned to another region. To resolve conflicting assignments, the Quartus II software maintains an order of precedence for LogicLock assignments. The following order of precedence, from highest to lowest, applies:

- Exact node-level assignments
- Path-based and wildcard assignments
- Hierarchical assignments
Conflicts can arise within path-based and wildcard assignments when one path-based or wildcard assignment contradicts another path-based or wildcard assignment. For example, a path-based assignment is made containing a node labeled X and assigned to LogicLock region PATH_REGION. A second assignment is made using wildcard assignment X* with node X being placed into region WILDCARD_REGION. As a result of these two assignments, node X is assigned to two regions: PATH_REGION and WILDCARD_REGION.

To resolve this type of conflict, the Quartus II software maintains the order in which the assignments were made and grants the higher priority to the most recently created assignment.

Open the Priority dialog box by selecting Priority on the General tab of the LogicLock properties dialog box. You can change the priority of path-based and wildcard assignments with the Up and Down buttons in the Priority dialog box. To prioritize assignments between regions, you must select multiple LogicLock regions and then open the Priority dialog box from the LogicLock Properties window.

Normally, all nodes assigned to a particular LogicLock region reside within the boundaries of that region.

Virtual Pins

Usually, when you compile a design in the Quartus II software, all I/O ports are directly mapped to pins on the targeted device. However, there may be situations where you do not want to map all I/O ports to the device pins; use the Virtual Pin assignment in such cases.

A virtual pin is an I/O element which you do not intend to bring to the chip pins. You can create a virtual pin by assigning the Virtual Pin logic option to an I/O element. When you compile a design with some I/O elements assigned as virtual pins, those I/O elements are mapped to a logic element and not to a pin during compilation, and are then implemented as a LUT. You might use virtual pin assignments when you compile a partial design, because not all the I/Os from a partial design may drive chip pins at the top level.

The Virtual Pin assignment communicates to the Quartus II software which I/O ports of the design module are internal nodes in the top-level design. These assignments prevent the number of I/O ports in the lower-level modules from exceeding the total number of available device pins. Every I/O port that is designated a virtual pin is mapped to either an LCELL or an adaptive logic module (ALM), depending on the target device.

Bidirectional, registered I/O pins, and I/O pins with output enable signals cannot be virtual pins.

In the top-level design, these virtual pins are connected 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. The Virtual Pin option can be useful when compiling a LogicLock module with more pins than the target device allows. The Virtual Pin option can enable timing analyses that more closely match the performance of the LogicLock module when it is integrated into the top-level design.
Apply the following guidelines when creating virtual pins in the Quartus II software:

- Do not declare clock pins as virtual pins
- Nodes or signals that drive physical device pins in the top-level design should not be declared as virtual pins

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

**Using LogicLock Regions in the Chip Planner**

You can easily edit properties of existing LogicLock regions or assign resources to them in the Chip Planner. You can also create new LogicLock regions using the Chip Planner.

**Viewing Connections Between LogicLock Regions in the Chip Planner**

You can view and edit LogicLock regions using the Chip Planner. Select the Floorplan Editing (Assignment) task or any task with the User-assigned LogicLock regions setting enabled to manipulate LogicLock regions.

The Chip Planner shows the connections between LogicLock regions. By default, each connection is represented as an individual line drawn between LogicLock regions. You can choose to display connections between LogicLock regions as a single bundled connection rather than as individual connection lines. To use this option, open the Chip Planner floorplan and on the View menu, click Generate Inter-region Bundles.

In the Generate Inter-region Bundles dialog box, specify the Source node to region fanout less than and the Bundle width greater than values.

For more information about the Generate Inter-region Bundles dialog box, refer to the Quartus II Help.

**Design Floorplan Analysis Using the Chip Planner**

The Chip Planner helps you visually analyze the floorplan of your design at any stage of your design cycle. With the Chip Planner, you can view post-compilation placement, connections, and routing paths. You can also create LogicLock regions and location assignments. The Chip Planner allows you to create new logic cells and I/O atoms and to move existing logic cells and I/O atoms using the architectural floorplan of your design. You can also see global and regional clock regions within the device, and the connections between both I/O atoms and PLLs and the different clock regions.

From the Chip Planner, you can launch the Resource Property Editor, which you can use to change the properties and parameters of device resources, and modify connectivity between certain types of device resources. The Change Manager records any changes that you make to your design floorplan, so that you can selectively undo changes if necessary.
For more information about the Resource Property Editor and the Change Manager, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

The following sections present Chip Planner floorplan views and design analysis procedures which you can use with any predefined task—unless explicitly stated that a given procedure requires a specific task or editing mode).

**Chip Planner Floorplan Views**

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

**First-Level View**

The first level provides a high-level view (LAB level view) of the entire device floorplan. You can locate a node and view the placement of that node in your design. Figure 12–16 shows the Chip Planner’s Floorplan first-level view of a Stratix device.

*Figure 12–16. Chip Planner’s First-Level Floorplan View*
Each resource is shown in a different color. The Chip Planner floorplan uses a gradient color scheme in which the color becomes darker as the utilization of a resource increases. For example, as more LEs are used in the logic array block (LAB), the color of the LAB becomes darker.

When you place the mouse pointer over a resource at this level, a tooltip appears that briefly describes the utilization of the resource (Figure 12–17).

**Figure 12–17.** Tooltip Message: First-Level View

---

**Second-Level View**

As you zoom in, the level of detail increases. Figure 12–18 shows the second-level view of the Chip Planner Floorplan for a Stratix device.

**Figure 12–18.** Chip Planner’s Second-Level Floorplan View

At this zoom level, the contents of LABs and I/O banks and the routing channels that connect resources are all visible.

When you place the mouse pointer over an LE or ALM at this level, a tooltip is displayed (Figure 12–19) that shows the name of the LE/ALM, the location of the LE/ALM, and the number of resources that are used with that LAB. When you place the mouse pointer over an interconnect, the tooltip shows the routing channels that are used by that interconnect. At this zoom level, you can move LEs, ALMs, and I/Os from one physical location to another.
Third-Level View

The third level provides a more detailed view, displaying each routing resource that is used within a LAB in the FPGA. Figure 12–20 shows the level of detail at the third-level view for a Stratix device.

From the third level, you can move LEs, ALMs, and I/Os from one physical location to another. You can move a resource by selecting, dragging, and dropping it into the desired location. At this level, you can also create new LEs and I/Os when you are in the post-compilation (ECO) mode.

You can delete a resource only after all of its fan-out connections are removed. Moving nodes in the Floorplan Editing (Assignment) task creates an assignment. However, if you move logic nodes in the Post-Compilation Editing (ECO) task, that change is considered an ECO change. For more information about Floorplan Assignments, refer to “Viewing Assignments in the Chip Planner” on page 12–39.

For more information about performing ECOs, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.
Bird's Eye View

The Bird’s Eye View (Figure 12–21) 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.

Figure 12–21. Bird’s Eye View

The Bird’s Eye View is a separate window that is linked to the Chip Planner floorplan. When you select an area of interest in the Bird’s Eye View, the Chip Planner floorplan automatically refreshes to show that region of the device. As you change the size of the main-view rectangle in the Bird’s Eye View window, the main Chip Planner floorplan window also zooms in (or zooms out). You can make the main-view rectangle smaller in the Bird’s Eye View to see more detail on the Chip Planner floorplan window by right-clicking and dragging inside the Bird’s Eye View.

You can use the Bird’s Eye View when you are interested in resources at opposite ends of the chip, and you want to quickly navigate between resource elements without losing your frame of reference.
**Selected Elements Window**

The Selected Elements Window lists the objects (such as atoms, paths, LogicLock regions, or routing elements) currently selected in the Chip Planner. To display the Selected Elements Window, click **Selected Elements Window** on the **View** menu in the Chip Planner.

![Selected Elements Window](image)

**Figure 12–22. Selected Elements Window**

---

**Viewing Architecture-Specific Design Information**

With the Chip Planner, you can view the following architecture-specific information related to your design:

- **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 how a logic element (LE) is configured within your design. For example, you can view which LE inputs are used; if the LE utilizes the register, the look-up table (LUT), or both; as well as the signal flow through the LE.

- **ALM configuration**—View how an ALM is configured within your design. For example, you can view which ALM inputs are used, if 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 how the device I/O resources are used. For example, you can view which components of the I/O resources are used, if the delay chain settings are enabled, which I/O standards are set, and the signal flow through the I/O.

- **PLL configuration**—View how a phase-locked loop (PLL) is configured within 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 DATA input to the COMBOUT output.
In addition, you can modify the following properties of an Altera device with the Chip Planner:

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

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

Viewing Available Clock Networks in the Device

When you select Clock Regions (Assignment) from the Task list, you can display the areas of the chip that are driven by global and regional clock networks. This global clock display feature is available for Arria GX, Arria II GX, Cyclone II, Cyclone III, HardCopy II, HardCopy III, Stratix II, Stratix II GX, Stratix III, and Stratix IV device families.

When you select the Clock Regions task, the Chip Planner displays various types of regional and global clocks and the regions they cover in the device. The connectivity between clock regions, pins, and PLLs is also shown. Clock regions are shown with rectangular overlay boxes with name labels of clock type and index. You can select each clock network region by clicking on it. The clock-shaped icon at the top-left corner indicates that the region represents a clock network region.

Clock types are listed in the Layer Settings window. You can change the color of the clock network in the Chip Planner on the Options page of the Tools menu.

You can customize your view of the global clock networks by using the layers setting in the Chip Planner. You can turn on or off the display of all clock regions with the All types option. When the selected device does not contain a specific clock region, the option for that category is turned off in the dialog box. Figure 12–23 shows the potential fan-in in the Chip Planner.
To trace the possible connectivity to each clock network region, select the clock network region and use the Generate Potential Fan-In and Generate Potential Fan-Out commands.

If you are interested in locating the clock regions that a pin or PLL can feed, select the pin or the PLL, then use the Generate Fan-Out Connections command. Connection arrows are drawn from the selected pins or PLLs to their clock regions.

When you use the Generate Fan-In Connections and Generate Fan-Out Connections commands, the Chip Planner shows connections that are actually used in the netlist for the selected clock region.

**Viewing 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 registers-to-devices I/Os. The View Critical Paths feature displays routing paths in the Chip Planner, as shown in Figure 12–24. The criticality of a path is determined by its slack and is shown in the timing analysis report. Design analysis for timing closure is a fundamental requirement for optimal performance in highly complex designs. The Chip Planner helps you close timing on complex designs with its analytical capability.
Viewing critical paths in the Chip Planner helps you analyze why a specific path is failing. You can see if any modification in the placement can potentially reduce the negative slack. You can display details of a path (to expand/collapse the path to/from the connections in the path) by clicking Expand Connections/Paths in the toolbar, or by clicking on the “+/-” on the label.

To view critical paths in the Chip Planner, on the View menu, click Critical Path Settings. In the Critical Path Settings dialog box, click Show Path (refer to Figure 12–25 on page 12–29).

If you are using the TimeQuest Timing Analyzer, you can locate the failing paths starting from the timing report. To locate the critical paths, run the Report Timing task from the Custom Reports group in the Tasks pane of the TimeQuest Timing Analyzer. From the View pane, which lists the failing paths, right-click on any failing path or node, and select Locate Path. From the Locate dialog box, select Chip Planner to see the failing path in the Chip Planner.
When viewing critical paths, you can specify the clock in the design you want to view. You determine the paths to be displayed by specifying the slack threshold in the slack field of the Critical Path Settings for Chip Planner dialog box. This dialog box also helps you to filter specific paths based on the source and destination registers.

Timing settings must be made and a timing analysis performed for paths to be displayed in the floorplan.

For more information about performing static timing analysis with the Quartus II Classic Timing Analyzer, refer to the Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook. For more information about performing static timing analysis with the Quartus II TimeQuest Timing Analyzer, refer to the Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook.

**Viewing Physical Timing Estimates**

In the Chip Planner, you can select a resource and see the approximate delay to any other resource on the device. After you select a resource, the delay is represented by the color of potential destination resources. The lighter the color of the resource, the longer the delay.

To see the physical timing map of the device, in the Chip Planner, click the Layers icon located next to the Task menu. Under Background Color Map, select Physical Timing Estimate. Select a source and move your cursor to a destination resource. The Chip Planner displays the approximate routing delay between your selected source and destination register (Figure 12–26).
You can use the physical timing estimate information when attempting to improve the Fitter results by manually moving logic in a device or when creating LogicLock regions to group logic together. This feature allows you to estimate the physical routing delay between different nodes so that you can place critical nodes and modules closer together, and move non-critical or unrelated nodes and modules further apart.

In addition to reducing delay between critical nodes, you can make placement assignments to reduce the routing congestion between critical and noncritical entities and modules. This allows the Fitter to meet the design timing requirements.

Moving logic and creating manual placements is an advanced technique to meet timing requirements and must be done after careful analysis of the design. Moving nodes in the Floorplan Editing (Assignment) task creates an assignment. However, if you move logic nodes in the Post-Compilation Editing (ECO) task, that change is considered an ECO change.

For more information about Floorplan Assignments, refer to “Viewing Assignments in the Chip Planner” on page 12–39.

For more information about performing ECOs, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

Figure 12–26. Chip Planner Displaying Routing Delay
Viewing Routing Congestion

The Routing Congestion view allows you to determine the percentage of routing resources used after a compilation. This feature identifies where there is a lack of routing resources. This information helps you make design changes that might ease routing congestion and thus meet design requirements. Congestion is represented visually by the color and shading of logic resources; darker shading represents a greater utilization of routing resources.

You can set a routing congestion threshold to identify areas of high routing congestion with the Routing Congestion Settings dialog box by selecting the Routing Congestion (ECO) task from the drop-down task list or by selecting Routing Utilization from the layers settings. In the Routing Congestion Settings dialog box, set the threshold level for congestion indication and click Apply. You can also select the interconnect type. All areas that exceed the specified threshold appear in red (Figure 12–27).

Figure 12–27. Areas Exceeding Threshold

If you are using a HardCopy II device, turn on Routing Congestion to see the routing congestion in the device by selecting Routing Utilization from the Layers Settings window.

To view the routing congestion in the Chip Planner, click the Layers icon located next to the Task menu. Under Background Color Map, select the Routing Utilization map (Figure 12–28). Any areas that exceed the threshold appear red. Use this congestion information to evaluate if you could modify the floorplan, or make changes to the RTL to reduce routing congestion.
Viewing I/O Banks

The Chip Planner can show all of the I/O banks of the device. To see the I/O bank map of the device, click the Layers icon located next to the Task menu. Under Background Color Map, select I/O Banks. Refer to Figure 12–29.
Generating Fan-In and Fan-Out Connections

The ability to display fan-in and fan-out connections enables you to view the atoms that fan-in to or fan-out from the selected atom. To remove the connections displayed, use the Clear Unselected Connections/Paths icon in the Chip Planner toolbar. Figure 12–30 shows the fan-in connections for the selected resource.
Generating Immediate Fan-In and Fan-Out Connections

The ability to display immediate fan-in and fan-out connections enables you to view the immediate resource that is the fan-in or fan-out connection for the selected atom. For example, selecting a logic resource and choosing to view the immediate fan-in enables you to see the routing resource that drives the logic resource. You can generate immediate fan-in and fan-outs for all logic resources and routing resources. To remove the connections that are displayed, click the Clear Connections icon in the toolbar. Figure 12–31 shows the immediate fan-out connections for the selected resource.

Figure 12–30. Generated Fan-In
Highlight Routing

The **Highlight Routing** command enables you to highlight the routing resources used by a selected path or connection. Figure 12–32 shows the routing resources used between two logic elements.
Show Delays

You can view the timing delays for the highlighted connections when generating connections between elements. For example, you can view the delay between two logic resources or between a logic resource and a routing resource. Figure 12–33 shows the delays between several logic elements.
Exploring Paths in the Chip Planner

You can use the Chip Planner to explore paths between logic elements. The following example uses the Chip Planner to traverse paths from the Timing Analysis report.

**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.
2. Right-click the path in the Timing Analysis report, point to **Locate**, and click **Locate in Chip Planner (Floorplan & Chip Editor)**.

Figure 12–34 shows the path that is displayed in the Chip Planner.
To view the routing resources taken for a path you have located in the Chip Planner, click the Highlight Routing icon in the Chip Planner toolbar, or from the View menu, click Highlight Routing.

**Analyzing Connections for a Path**

To determine the connections between items in the Chip Planner, click the Expand Connections/Paths icon on the toolbar. To add the timing delays between each connection, click the Show Delays icon on the toolbar. **Figure 12–35** shows the connections for the selected path that are displayed in the Chip Planner. To see the constituent delays on the selected path, click on the “+” sign next to the path delay displayed in the Chip Planner.
Viewing Assignments in the Chip Planner

You can view location assignments by selecting the appropriate layer set in the Chip Planner. To view location assignments in the Chip Planner, select the Floorplan Editing (Assignment) task or any custom task with Assignment editing mode. See Figure 12–36.

The Chip Planner shows location assignments graphically, by displaying assigned resources in a particular color (gray, by default). You can create or move an assignment by dragging the selected resource to a new location.

Figure 12–36. Viewing Assignments in the Chip Planner

You can make node and pin location assignments and assignments to LogicLock regions and custom regions using the drag-and-drop method in the Chip Planner. The assignments that you create are applied by the Fitter during the next place-and-route operation.

To learn more about working with location assignments, refer to the Quartus II Help.

Viewing Routing Channels for a Path in the Chip Planner

To determine the routing channels between connections, click the Highlight Routing icon on the toolbar. Figure 12–37 shows the routing channels used for the selected path in the Chip Planner.
You can view and edit resources in the FPGA using the Resource Property Editor mode of the Chip Planner. For more information, refer to the Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook.

**Cell Delay Table**

You can view the propagation delay from all inputs to all outputs for any LE in your design. To see the Cell Delay Table for an atom, select the atom in the Chip Planner and right-click. From the pop-up menu, click **Locate** and then click **Locate in Resource Property Editor**. The Resource Property window shows you the atom properties along with the Cell Delay Table, indicating the propagation delay from all inputs to all outputs. **Figure 12–38** shows the Cell Delay Table.
Timing numbers are displayed only when there is a direct path between the source input port and the destination output port. In cases where there is no path, or the path requires an intermediate buried timing node, the displayed cell delay is given as “N/A.”

**Viewing High-Speed and Low-Power Tiles in Stratix III Devices in the Chip Planner**

The Chip Planner has a predefined task, **Power Analysis (Assignment)**, which shows the power map of a Stratix III device. Stratix III 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 Quartus II software. These ALMs are grouped together to form larger blocks, called “tiles.”

To learn more about power analyses and optimizations in Stratix III devices, refer to *AN 437: Power Optimization in Stratix III FPGAs*. To learn more about power analyses and optimizations in Stratix IV devices, refer to *AN 514: Power Optimization in Stratix IV FPGAs*.

When the **Power Analysis (Assignment)** task is selected in the Chip Planner for Stratix III devices, low-power and high-speed tiles are displayed in different colors; yellow tiles operate in a high-speed mode, while blue tiles operate in a low-power mode (see Figure 12–39). When you select the Power Analysis task, you can perform all floorplanner-related functions for this task, however you cannot edit tiles to change the power mode.
For older device families not supported by the Chip Planner, you can perform floorplan analysis using the Timing Closure Floorplan. Table 12–1 on page 12–1 lists the device families supported by the Timing Closure Floorplan Editor and the Chip Planner.

The Timing Closure Floorplan Editor allows you to analyze your design visually before and after performing a full design compilation in the Quartus II software. This floorplan editor, used in conjunction with the Classic Timing Analyzer, provides a method for performing design analysis.

To start the Timing Closure Floorplan Editor, on the Assignments menu, click Timing Closure Floorplan.

If the device in your project is not supported by the Timing Closure Floorplan, the following message appears:

Can’t display a floorplan: the current device family is only supported by Chip Planner.

If your target device is supported by the , you can also start the Timing Closure Floorplan by right-clicking any of the following sources, pointing to Locate, and clicking Locate in Timing Closure Floorplan:

- Compilation Report
- Node Finder
Chapter 12: Analyzing and Optimizing the Design Floorplan

Design Analysis Using the Timing Closure Floorplan

- Project Navigator
- RTL source code
- RTL Viewer
- Simulation Report
- Timing Report

Figure 12–40 shows the icons in the Timing Closure Floorplan toolbar.

Figure 12–40. Timing Closure Floorplan Icons

Timing Closure Floorplan Views

The Timing Closure Floorplan Editor provides the following views of your design:

- Field view
- Interior Cells view
- Interior LAB view

The following two views open the Pin Planner:

- Package Top view
- Package Bottom view

Field View

The Field view provides a color-coded, high-level view of the resources used in the device floorplan. All device resources, such as embedded system blocks (ESBs) and MegaLAB blocks, are outlined.
To view the details of a resource in the Field view, select the resource, right-click, and click Show Details. To hide the details, select all the resources, right-click, and click Hide Details (Figure 12–41).

**Figure 12–41.** Show and Hide Details of a Logic Array Block in Field View

---

**Other Views**

You can view your design in the Timing Closure Floorplan Editor with the Interior Cells, Interior LABs, Package Top, and Package Bottom views. Use the View menu to display the various floorplan views. The Interior Cells view provides a detailed view of device resources, including device pins and individual logic elements within a MegaLAB.

**Viewing Assignments**

The Timing Closure Floorplan Editor differentiates between user assignments and Fitter placements. If the device is changed after a compilation, the user assignment and Fitter placement options cannot be used together. When this situation occurs, the Fitter placement displays the last compilation result and the user assignment displays the floorplan of the newly selected device.

To see the user assignments, click the Show User Assignments icon in the Floorplan Editor toolbar, or, on the View menu, point to Assignments and click Show User Assignments. To see the Fitter placements, click the Show Fitter Placements icon in the Floorplan Editor toolbar, or, on the View menu, point to Assignments and click Show Fitter Placements. Figure 12–42 shows the Fitter placements.
Figure 12–42. Fitter Placements

Viewing Critical Paths

The View Critical Paths feature displays routing paths in the floorplan, as shown in Figure 12–43. The criticality of a path is determined by its slack and is also shown in the Timing Analysis report.
Figure 12–43. Critical Paths

To view critical paths in the Timing Closure Floorplan, click the Critical Path Settings icon on the toolbar, or, on the View menu, point to Routing and click Critical Path Settings.

When viewing critical paths, you can specify the clock in the design to be viewed. You can determine which paths to display by specifying the slack threshold in the slack field.

You must make timing settings and perform timing analysis to view paths in the floorplan.

For more information about performing static timing analyses of your design with a timing analyzer, refer to the Quartus II Classic Timing Analyzer and the Quartus II TimeQuest Timing Analyzer chapters in volume 3 of the Quartus II Handbook.

You can view critical paths to determine the criticality of nodes based on placement. You can view the details of the critical path in a number of ways.

The default view in the Timing Closure Floorplan shows the path with the source and destination registers displayed. You can also view all the combinational nodes along the worst-case path between the source and destination nodes. To view the full path, click on the delay label to select the path, right-click, and select Show Path Edges. Figure 12–44 shows the critical path through combinational nodes. To hide the combinational nodes, select the path, right-click, and select Hide Path Edges.
You must view the routing delays to select a path.

Figure 12–44. Worst-Case Combinational Path Showing Path Edges

After running timing analysis, you can locate timing paths from the timing reports file produced. Right-click on any row in the report file, point to Locate, and click Locate in Timing Closure Floorplan. The Timing Closure Floorplan window opens with the timing path highlighted.

For more information about optimizing your design in the Quartus II software, refer to the Area and Timing Optimization chapter in volume 2 of the Quartus II Handbook. With the options and tools available in the Timing Closure Floorplan and the techniques described in that chapter, the Quartus II software can help you achieve timing closure in a more time-efficient manner.

Viewing Routing Congestion

The View Routing Congestion feature allows you to determine the percentage of routing resources used after a compilation. This feature identifies where there is a lack of routing resources.

The congestion is shown by the color and shading of logic resources. The darker shading represents a greater routing resource utilization. Logic resources that are red have routing resource utilization greater than the specified threshold.
The routing congestion view is only available from the View menu when you enable the Field view. To view routing congestion in the floorplan, click the Show Routing Congestion icon, or on the View menu, point to Routing and click Show Routing Congestion. To set the criteria for the critical path you want to view, click the Routing Congestion Settings icon, or on the View menu, point to Routing and click Routing Congestion Settings.

In the Routing Congestion Settings dialog box, you can choose the routing resource (interconnect type) you want to examine and set the congestion threshold. Routing congestion is calculated based on the total resource usage divided by the total available resources.

If you use the routing congestion viewer to determine where there is a lack of routing resources, examine each routing resource individually to determine which ones use close to 100% of the available resources (Figure 12–45). Use this congestion information to evaluate whether you should modify the floorplan, or make changes to the RTL to reduce routing congestion.

**Figure 12–45.** Routing Congestion of a Sample Design in a MAX3000A series Device

---

**Scripting Support**

You can run procedures and create the 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 Quartus II command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

The same information is available in the Quartus II Help, and in the Quartus II Scripting Reference Manual.
For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

For information about all settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Reference Manual*.

**Initializing and Uninitializing a LogicLock Region**

You must initialize the LogicLock data structures before creating or modifying any LogicLock regions and before executing any of the Tcl commands listed below.

Use the following Tcl command to initialize the LogicLock data structures:

```tcl
initialize_logiclock
```

Use the following Tcl command to uninitialize the LogicLock data structures before closing your project:

```tcl
uninitialize_logiclock
```

**Creating or Modifying LogicLock Regions**

Use the following Tcl command to create or modify a LogicLock region:

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

In the above example, the size of the region is set to auto and the state is set to floating.

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

For more information about creating LogicLock regions, refer to the sections “Creating LogicLock Regions” on page 12–7 and “Creating LogicLock Regions with the Chip Planner” on page 12–9.

**Obtaining LogicLock Region Properties**

Use the following Tcl command to obtain LogicLock region properties. This example returns the height of the region named *my_region*:

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

**Assigning LogicLock Region Content**

Use the following Tcl commands to assign or change nodes and entities in a LogicLock region. This example assigns all nodes with names matching *fifo* to the region named *my_region*.

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

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

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

For more information about assigning LogicLock Region Content, refer to “Assigning LogicLock Region Content” on page 12–9.
Save a Node-Level Netlist for the Entire Design into a Persistent Source File

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

```tcl
set_global_assignment -name LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT ON
set_global_assignment -name LOGICLOCK_INCREMENTAL_COMPILE_FILE <filename>
```

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

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

The saving of a node-level netlist to a persistent source file is not supported for designs targeting newer devices such as the Stratix IV, Stratix III, Cyclone III, Arria II GX, or Arria GX.

Setting LogicLock Assignment Priority

Use the following Tcl code to set the priority for a LogicLock region’s members. This example reverses the priorities of the LogicLock region in your design.

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

Assigning Virtual Pins

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

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

For more information about assigning virtual pins, refer to “Virtual Pins” on page 12–19.

For more information about Tcl scripting, refer to the Tcl Scripting chapter in volume 2 of the Quartus II Handbook.

Conclusion

Design floorplan analysis is a valuable method for achieving timing closure and timing closure optimal performance in highly complex designs. With their analysis capability, the Quartus II Chip Planner and the Timing Closure Floorplan help you close timing quickly on your designs. Using these tools together with LogicLock and Incremental Compilation enables you to compile your designs hierarchically, preserving the timing results from individual compilation runs. You can use LogicLock regions as part of an incremental compilation methodology to improve your productivity. You can also include a module in one or more projects while maintaining performance and reducing development costs and time-to-market. LogicLock region assignments give you complete control over logic and memory placement to improve the performance of non-hierarchical designs as well.
Referenced Documents

This chapter references the following documents:

- AN 437: Power Optimization in Stratix III FPGAs
- Area and Timing Optimization chapter in volume 2 of the Quartus II Handbook
- Best Practices for Incremental Compilation Partition and Floorplan Assignments chapters in volume 1 of the Quartus II Handbook
- Command-Line Scripting chapter in volume 2 of the Quartus II Handbook
- Engineering Change Management with the Chip Planner chapter in volume 2 of the Quartus II Handbook
- I/O Management chapter in volume 2 of the Quartus II Handbook
- Quartus II Classic Timing Analyzer chapter in volume 3 of the Quartus II Handbook
- Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook
- Quartus II Scripting Reference Manual
- Quartus II Settings File Manual
- The Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook
- Tcl Scripting chapter in volume 2 of the Quartus II Handbook

Document Revision History

Table 12–3 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2009 v9.1.0</td>
<td>Updated supported device information throughout</td>
<td>Updated for the Quartus II 9.1 software release.</td>
</tr>
<tr>
<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 II Handbook, available in the Quartus II Handbook Archive.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated “Creating Non-Rectangular LogicLock Regions” section</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Added “Selected Elements Window” section</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated table 12-1</td>
<td></td>
</tr>
<tr>
<td>March 2009 v9.0.0</td>
<td>Was chapter 10 in the 8.1.0 release.</td>
<td>Updated for the Quartus II 9.0 software release.</td>
</tr>
</tbody>
</table>
## Table 12–3. Document Revision History  (Part 2 of 2)

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2008 v8.1.0</td>
<td><img src="#" alt="List of changes" /></td>
<td>Updated for the Quartus II 8.1 software release.</td>
</tr>
<tr>
<td>May 2008 v8.0.0</td>
<td><img src="#" alt="List of changes" /></td>
<td>Updated for the Quartus II 8.0 software release.</td>
</tr>
</tbody>
</table>

For previous versions of the Quartus II Handbook, refer to the [Quartus II Handbook Archive](#).

---

For previous versions of the Quartus II Handbook, refer to the [Quartus II Handbook Archive](#).
The Quartus® II software offers physical synthesis optimizations to improve your design beyond the optimization performed in the normal course of the Quartus II compilation flow.

### Introduction

Physical synthesis optimizations can help improve the performance of your design regardless of the synthesis tool used, although the effect of physical synthesis optimizations depends on the structure of your design.

Netlist optimization options work with the atom netlist of your design, which describes a design in terms of Altera®-specific primitives. An atom netlist file can be an Electronic Design Interchange Format (.edf) file or a Verilog Quartus Mapping (.vqm) file generated by a third-party synthesis tool, or a netlist used internally by the Quartus II software. Physical synthesis optimizations are applied at different stages of the Quartus II compilation flow, either during synthesis, fitting, or both.

This chapter explains how the physical synthesis optimizations in the Quartus II software can modify your design’s netlist to improve your quality of results. This chapter also provides information about preserving compilation results through back-annotation and writing out a new netlist, and provides guidelines for applying the various options.

Because the node names for primitives in the design can change when you use physical synthesis optimizations, you should evaluate whether your design flow requires fixed node names. If you use a verification flow that might require fixed node names, such as the SignalTap® II Logic Analyzer, formal verification, or the LogicLock based optimization flow (for legacy devices), you must turn off the synthesis netlist optimization and physical synthesis options.

### WYSIWYG Primitive Resynthesis

If you use a third-party tool to synthesize your design, use the Perform WYSIWYG primitive resynthesis option to apply optimizations to the synthesized netlist.

The Perform WYSIWYG primitive resynthesis option directs the Quartus II software to un-map the logic elements (LEs) in an atom netlist to logic gates, and then re-map the gates back to Altera-specific primitives. Third-party synthesis tools generate an atom netlist file that specifies Altera-specific primitives. Atom netlist files can be either an .edf or .vqm file generated by the third-party synthesis tool. When you turn on the Perform WYSIWYG primitive resynthesis option, the Quartus II software can work on different techniques specific to the device architecture 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 has no effect if you are using Quartus II integrated synthesis to synthesize your design.
To turn on the **Perform WYSIWYG primitive resynthesis** option, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Analysis and Synthesis Settings**. The **Analysis & Synthesis Settings** page appears.
3. Turn on **Perform WYSIWYG Primitive Resynthesis**, and click **OK**.

If you want to perform WYSIWYG resynthesis on only a portion of your design, you can use the Assignment Editor to assign the **Perform WYSIWYG primitive resynthesis** logic option to a lower-level entity in your design. This logic option can be used with Arria® II GX, Arria GX, Cyclone® series, HardCopy® series, MAX® II series, or Stratix® series device families.

The results of the remapping depend on the **Optimization Technique** you choose. To select an **Optimization Technique**, perform the following steps:

1. In the **Category** list, select **Analysis & Synthesis Settings**. The **Analysis & Synthesis Settings** page appears.
2. Under **Optimization Technique**, select **Speed**, **Area**, or **Balanced** to specify how the Quartus II technology mapper optimizes the design. The **Balanced** setting is the default for many Altera device families; this setting optimizes the timing critical parts of the design for speed and the rest of the design for area.
3. Click **OK**.

Refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook* for details on the Optimization Technique option.

*Figure 13–1* shows the Quartus II software flow for the WYSIWYG primitive resynthesis feature.

**Figure 13–1. WYSIWYG Primitive Resynthesis**

The **Perform WYSIWYG primitive resynthesis** option is not beneficial if you are using Quartus II integrated synthesis; it is intended for optimization of projects that use other EDA synthesis tools.
Performing Physical Synthesis Optimizations

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

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 remapped by the Quartus II software. The remapping process removes duplicate registers, but registers that are not removed retain the same name after remapping.

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.

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 read by the Quartus II software.

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

For more information about using these attributes during synthesis in the Quartus II software, refer to the **Quartus II Integrated Synthesis** chapter in volume 1 of the **Quartus II Handbook**.

**Performing Physical Synthesis Optimizations**

The Quartus II design flow involves separate steps of synthesis and fitting. The synthesis step optimizes the logical structure of a circuit for area, speed, or both. The Fitter then places and routes the logic cells to ensure critical portions of logic are close together and use the fastest possible routing resources. While you are using this push-button flow, the synthesis stage is unable to anticipate the routing delays seen in the Fitter. Because routing delays are a significant part of the typical critical path delay, the physical synthesis optimizations available in the Quartus II software take those routing delays into consideration and focus timing-driven optimizations at those parts of the design. This tight integration of the fitting and synthesis processes is known as physical synthesis.

The following sections describe the physical synthesis optimizations available in the Quartus II software, and how they can help improve your performance results. Physical synthesis optimization options can be used with Arria GX, Arria II GX, Cyclone, HardCopy, and Stratix series device families.
If you are migrating your design to a HardCopy II device, you can target physical synthesis optimizations to the FPGA architecture in the FPGA-first flow or to the HardCopy II architecture in the HardCopy-first flow. The optimizations are mapped to the other device architecture during the migration process.

You cannot target optimizations to optimize for both device architectures individually because doing so results in a different post-fitting netlist for each device.

For more information about using physical synthesis with HardCopy devices, refer to the *Quartus II Support of HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*.

You can choose the physical synthesis optimization options you want for your design during synthesis and fitting in the Physical Synthesis Optimizations page under the Compilation Process Settings page in the Settings dialog box. The settings include optimizations for improving performance and fitting in the selected device.

You can also set the effort level for physical synthesis optimizations. Normally, physical synthesis optimizations increase the compilation time; however, you can select the Fast effort level if you want to limit the increase in compilation time. When you select the Fast effort level, the Quartus II software performs limited register retiming operations during fitting. The Extra effort level runs additional algorithms to get the best circuit performance, but results in increased compilation time.

To optimize performance, the following options are available:

- Perform physical synthesis for combinational logic
- Perform register retiming
- Perform automatic asynchronous signal pipelining
- Perform register duplication

To optimize for better fitting, you can choose from the following options:

- Perform physical synthesis for combinational logic
- Perform logic to memory mapping

To view and modify the physical synthesis optimization options, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
3. Specify the options for performing physical synthesis optimizations.

Some physical synthesis options affect only registered logic and some options affect only combinational logic. Select options based on whether you want to keep the registers intact or not. For example, if your verification flow involves formal verification, you might have to keep the registers intact.
Performing Physical Synthesis Optimizations

All Physical Synthesis optimizations write results to the Netlist Optimizations report, which provides a list of atom netlist files that were modified, created, and deleted during physical synthesis. To access the Netlist Optimizations report, perform the following steps:

2. In the Compilation Report list, select Netlist Optimizations under Fitter.

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

2. In the Compilation Report list, select Analysis & Synthesis.

Nodes or entities that have the Netlist Optimizations logic option set to Never Allow are not affected by the physical synthesis algorithms. 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.

Automatic Asynchronous Signal Pipelining

The Perform automatic asynchronous signal pipelining option on the Physical Synthesis Optimizations page in the Compilation Process Settings section of the Settings dialog box allows the Quartus II Fitter to perform automatic insertion of pipeline stages for asynchronous clear and asynchronous load signals during fitting when these signals negatively affect performance. You can use this option if asynchronous control signal recovery and removal times are not achieving their requirements.

The Perform automatic asynchronous signal pipelining option improves performance for designs in which asynchronous signals in very fast clock domains cannot be distributed across the chip fast enough due to long global network delays. This optimization performs automatic pipelining of these signals, while attempting to minimize the total number of registers inserted.

The Perform automatic asynchronous signal pipelining option adds registers to nets driving the asynchronous clear or asynchronous load ports of registers. These additional registers add register delays (adds latency) to the reset, adding the same number of register delays for each destination using the reset. The additional register delays can change the behavior of the signal in the design; therefore, you should use this option only if additional latency on the reset signals does not violate any design requirements. This option also prevents the promotion of signals to global routing resources.

The Quartus II software performs automatic asynchronous signal pipelining only if Enable Recovery/Removal analysis is turned on. If you use the TimeQuest Timing Analyzer, Enable Recovery/Removal analysis is turned on by default. Pipelining is allowed only on asynchronous signals that have the following properties:

- The asynchronous signal is synchronized to a clock (a synchronization register drives the signal)
- The asynchronous signal fans-out only to asynchronous control ports of registers
To use **Enable Recovery/Removal analysis** with the Classic Timing Analyzer, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Classic Timing Analyzer Settings** under **Timing Analysis Settings**.
3. Click **More Settings**. The **More Timing Settings** dialog box appears.
4. In the **Name** list, select **Enable Recovery/Removal analysis**. In the **Setting** list, select **On**.
5. Click **OK**.
6. Click **OK**.

The Quartus II software does not perform automatic asynchronous signal pipelining on asynchronous signals that have the **Netlist Optimization** logic option set to **Never Allow**.

**Physical Synthesis for Combinational Logic**

To optimize the design and reduce delay along critical paths, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Physical Synthesis Optimizations** under **Compilation Process Settings**.
3. Turn on **Perform physical synthesis for combinational logic**.

The software performs this optimization by swapping the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. See **Figure 13–2** for an example. The **Perform physical synthesis for combinational logic** option also allows the duplication of LUTs to enable further optimizations on the critical path.

**Figure 13–2. Physical Synthesis for Combinational Logic**

In **Figure 13–2**, the critical input feeds through the first LUT to the second LUT. The Quartus II software swaps the critical input to the first LUT with an input feeding the second LUT, thus reducing the number of LUTs contained in the critical path. The synthesis information for each LUT is altered to maintain design functionality.

The **Perform physical synthesis for combinational logic** option affects only combinational logic in the form of LUTs. These transformations might occur during the synthesis stage or the Fitter stage during compilation. The registers contained in the affected logic cells are not modified. Inputs into memory blocks, DSP blocks, and I/O elements (IOEs) are not swapped.
The Quartus II software does not perform combinational optimization on logic cells that have the following properties:

- Are part of a chain
- Drive global signals
- Are constrained to a single logic array block (LAB) location
- Have the Netlist Optimizations option set to Never Allow

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

**Physical Synthesis for Registers—Register Duplication**

The Perform register duplication option on the Physical Synthesis Optimizations page in the Compilation Process Settings section of the Settings dialog box allows the Quartus II Fitter to duplicate registers based on Fitter placement information. You can also duplicate combinational logic when this option is enabled. A logic cell that fans out to multiple locations can be duplicated to reduce the delay of one path without degrading the delay of another. The new logic cell can be placed closer to critical logic without affecting the other fan-out paths of the original logic cell. Figure 13–3 shows an example of register duplication.

**Figure 13–3. Register Duplication**
The Quartus II software does not perform register duplication on logic cells that have the following properties:

- Are part of a chain
- Contain registers that drive asynchronous control signals on another register
- Contain registers that drive the clock of another register
- Contain registers that drive global signals
- Contain registers that are constrained to a single LAB location
- Contain registers that are driven by input pins without a $t_{SU}$ constraint
- Contain registers that are driven by a register in another clock domain
- Are considered virtual I/O pins
- Have the Netlist Optimizations option set to Never Allow

For more information about virtual I/O pins, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

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

**Physical Synthesis for Registers—Register Retiming**

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

Figure 13–4 shows an example of register retiming in which the 10-ns critical delay is reduced by moving the register relative to the combinational logic.

**Figure 13–4. Register Retiming Diagram**

Retiming can create multiple registers at the input of a combinational block from a register at the output of a combinational block. In this case, the new registers have the same clock and clock enable. The asynchronous control signals and power-up level are derived from previous registers to provide equivalent functionality. Retiming can also combine multiple registers at the input of a combinational block to a single register (Figure 13–5).
To move registers across combinational logic to balance timing, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
3. Specify your preferred option under Physical synthesis for performance and Effort level.
4. Click OK.

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

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

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

- Both registers in a direct connection from input pin-to-register-to-register if both registers have the same clock and the first register does not fan-out to anywhere else. These registers are considered synchronization registers.

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

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

1. On the Assignments menu, click Settings. The Settings dialog box appears.
2. In the Category list, select Analysis & Synthesis Settings. The Analysis & Synthesis Setting page appears.
4. In the Name list, select Synchronization Register Chain Length and modify the setting to match the synchronization register length used in your design. If you set a value of 1 for the Synchronization Register Chain Length, it means that any registers connected to the first register in a register-to-register connection can be moved during retiming. A value of $n > 1$ means that any registers in a sequence of length $1, 2, \ldots n$ are not moved during register retiming.

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

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

The Quartus II software does not usually retime registers across different clock domains; however, if you are using the Classic Timing Analyzer and have specified a global $f_{\text{MAX}}$ requirement, the Quartus II software interprets all clocks as being related to one another. Consequently, the Quartus II software might try to retime register-to-register paths associated with different clocks.

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

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

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

For more information about virtual I/O pins, refer to the Analyzing and Optimizing the Design Floorplan chapter in volume 2 of the Quartus II Handbook.

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

The Quartus II software generates the same results on every compilation for the same source code and settings on a given system, hence you do not need to preserve your results from compilation to compilation. When you make changes to the source code or to the settings, you usually get the best results by allowing the software to compile without using previous compilation results or location assignments. In some cases, if you avoid performing analysis and synthesis or `quartus_map`, and run the Fitter or another desired Quartus II executable instead, you can skip the synthesis stage of the compilation.

When you use the Quartus II incremental compilation flow, you can preserve synthesis results for a particular partition of your design by choosing a netlist type of post-synthesis. If you want to preserve fitting results between compilation runs, choose a netlist type of post-fit during incremental compilation.

The rest of this section is relevant only for those designs using older devices that do not support incremental compilation.

For information about the incremental compilation design methodology, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

You can preserve the resulting nodes from physical synthesis in older devices that do not support incremental compilation. You might need to preserve nodes if you use the LogicLock flow to back-annotate placement, import one design into another, or both. For all device families that support incremental compilation, use that feature to preserve results.

To preserve the nodes from Quartus II physical synthesis optimization options for older devices that do not support incremental compilation (such as Max II devices), perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Compilation Process Settings**. The **Compilation Process Settings** page appears.
3. Turn on **Save a node-level netlist of the entire design into a persistent source file**. This setting is not available for Cyclone III, Stratix III, and newer devices.
4. Click **OK**.

The **Save a node-level netlist of the entire design into a persistent source file** option saves your final results as an atom-based netlist in `.vqm` file format. By default, the Quartus II software places the `.vqm` file in the `atom_netlists` directory under the current project directory. To create a different `.vqm` file using different Quartus II settings, in the **Compilation Process Settings** page, change the **File name** setting.
If you use synthesis netlist optimizations (and not physical synthesis optimizations), generating a .vqm file is optional. To lock down the location of all logic and device resources in the design with or without a Quartus II-generated .vqm file, on the Assignments menu, click **Back-Annotate Assignments** and specify the desired options. You should use back-annotated location assignments unless you have finalized the design. Making any changes to the design invalidates your back-annotated location assignments. If you require changes later, use the new source HDL code as your input files, and remove the back-annotated assignments corresponding to the old code or netlist.

If you create a .vqm file to recompile the design, use the new .vqm file as the input source file and turn off the synthesis netlist optimizations for the new compilation.

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

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

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

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

---

**Physical Synthesis Options for Fitting**

The Quartus II software provides physical synthesis optimization options for improving fitting results. To access these options, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Physical Synthesis Optimizations** under **Compilation Process Settings**. The **Physical Synthesis Optimizations** page appears.
3. Under **Optimize for fitting** (physical synthesis for density), there are two physical synthesis options available to improve fitting your design in the target device: **Physical synthesis for combinational logic** and **Perform logic to memory mapping** (Table 13–1).
Applying Netlist Optimization Options

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 SignalTap II Logic Analyzer or formal verification that requires fixed or known node names.

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

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

For more information about using DSE, refer to the Design Space Explorer chapter in volume 2 of the Quartus II Handbook.

### 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 Quartus II Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

The Quartus II Scripting Reference Manual includes the same information in PDF form. For more information about Tcl scripting, refer to the Tcl Scripting chapter in volume 2 of the Quartus II Handbook. Refer to the Quartus II Settings File Manual for information about all settings and constraints in the Quartus II software. For more information about command-line scripting, refer to the Command-Line Scripting chapter in volume 2 of the Quartus II Handbook.

### Table 13–1. Physical Synthesis Optimizations Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Synthesis for Combinational Logic</td>
<td>When you select this option, the Fitter detects duplicate combinational logic and optimizes combinational logic to improve the fit.</td>
</tr>
<tr>
<td>Perform Logic to Memory Mapping</td>
<td>When you select this option, the Fitter can remap registers and combinational logic in your design into unused memory blocks and achieves a fit.</td>
</tr>
</tbody>
</table>
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>
```

### Synthesis Netlist Optimizations

Table 13–2 lists the Quartus II Settings File (.qsf) variable names and applicable values for the settings discussed in “WYSIWYG Primitive Resynthesis” on page 13–1. The .qsf file variable name is used 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>Quartus II 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 Technique</td>
<td>&lt;Device Family Name&gt;_OPTIMIZATION_TECHNIQUE</td>
<td>AREA, SPEED, BALANCED</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>
<tr>
<td>Save a node-level netlist into a persistent source file</td>
<td>LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Allow Netlist Optimizations</td>
<td>ADV_NETLIST_OPT_ALLOWED</td>
<td>&quot;ALWAYS ALLOW&quot;, DEFAULT, &quot;NEVER ALLOW&quot;</td>
<td>Instance</td>
</tr>
</tbody>
</table>

### Physical Synthesis Optimizations

Table 13–3 lists the .qsf file variable name and applicable values for the settings discussed in “Performing Physical Synthesis Optimizations” on page 13–3. The .qsf file variable name is used 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>Quartus II Settings File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Synthesis for Combinational Logic</td>
<td>PHYSICAL_SYNTHESIS_COMBO_LOGIC</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Automatic Asynchronous Signal Pipelining</td>
<td>PHYSICAL_SYNTHESISASYCHRONOUS_SIGNAL_PIPELINING</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
</tbody>
</table>
Table 13–3. Physical Synthesis Optimizations and Associated Settings (Part 2 of 2)

<table>
<thead>
<tr>
<th>Setting Name</th>
<th>Quartus II Settings File Variable Name</th>
<th>Values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform Register Duplication</td>
<td>PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Perform Register Retiming</td>
<td>PHYSICAL_SYNTHESIS_REGISTER_RETIMING</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>Power-Up Don’t Care</td>
<td>ALLOW_POWER_UP_DONT_CARE</td>
<td>ON, OFF</td>
<td>Global, Instance</td>
</tr>
<tr>
<td>Power-Up Level</td>
<td>POWER_UP_LEVEL</td>
<td>HIGH, LOW</td>
<td>Instance</td>
</tr>
<tr>
<td>Allow Netlist Optimizations</td>
<td>ADV_NETLIST_OPT_ALLOWED</td>
<td>&quot;ALWAYS ALLOW&quot;, DEFAULT,</td>
<td>Instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NEVER ALLOW&quot;</td>
<td></td>
</tr>
<tr>
<td>Save a node-level netlist into</td>
<td>LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT</td>
<td>ON, OFF</td>
<td>Global</td>
</tr>
<tr>
<td>a persistent source file</td>
<td>LOGICLOCK_INCREMENTAL_COMPILE_FILE</td>
<td>&lt;filename&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Incremental Compilation

For information about scripting and command line usage for incremental compilation as mentioned in “Preserving Your Physical Synthesis Results” on page 13–11, refer to the Quartus II Incremental Compilation for Hierarchical and Team-Based Design chapter in volume 1 of the Quartus II Handbook.

Back-Announcing Assignments

You can use the logiclock_back_annotate Tcl command to back-annotate resources in your design. This command can back-annotate resources in LogicLock regions, and resources in designs without LogicLock regions.

For more information about back-annotating assignments, refer to “Preserving Your Physical Synthesis Results” on page 13–11.

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

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

The logiclock_back_annotate command is in the backannotate package.

Conclusion

Physical synthesis optimizations restructure and optimize your design netlist. You can take advantage of these Quartus II netlist optimizations to help improve your quality of results.
Referenced Documents

This chapter references the following documents:

- *Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*
- *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*
- *Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*
- *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
- *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*
- *Quartus II Settings File Manual*
- *Quartus II Support for HardCopy Series Devices* chapter in volume 1 of the *Quartus II Handbook*
- *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*

Document Revision History

Table 13–4 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2009 v9.1.0</td>
<td>Added information to “Physical Synthesis for Registers—Register Retiming”</td>
<td>Updated for the Quartus II 9.1 software release.</td>
</tr>
<tr>
<td></td>
<td>Added information to “Applying Netlist Optimization Options”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Made minor editorial updates</td>
<td></td>
</tr>
<tr>
<td>March 2009 v9.0.0</td>
<td>Was chapter 11 in the 8.1.0 release.</td>
<td>Updated GUI references and procedure steps, and document structure for the Quartus II software 9.0 release.</td>
</tr>
<tr>
<td></td>
<td>Updated the “Physical Synthesis for Registers—Register Retiming” and “Physical Synthesis Options for Fitting”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated “Performing Physical Synthesis Optimizations”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deleted Gate-Level Register Retiming section.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Updated the referenced documents</td>
<td></td>
</tr>
<tr>
<td>November 2008 v8.1.0</td>
<td>Changed to 8½” × 11” page size. No change to content.</td>
<td>Updated for the Quartus II 8.1 software release.</td>
</tr>
<tr>
<td>May 2008 v8.0.0</td>
<td>Updated “Physical Synthesis Optimizations for Performance on page 11-9</td>
<td>Updated for Quartus II 8.0 version.</td>
</tr>
<tr>
<td></td>
<td>Added Physical Synthesis Options for Fitting on page 11-16</td>
<td></td>
</tr>
</tbody>
</table>

For previous versions of the *Quartus II Handbook*, refer to the *Quartus II Handbook Archive*. 
The Quartus® II software includes many advanced optimization algorithms to help you achieve timing closure, optimize area, and reduce dynamic power. Various settings and parameters control the behavior of the algorithms. These options provide complete control over the Quartus II software optimization and power techniques.

Each FPGA design is unique. There is no standard set of options that always results in the best performance or power utilization. Each design requires a unique set of options to achieve optimal performance. This chapter describes Design Space Explorer (DSE), a utility written in Tcl/Tk that automates finding the best set of options for your design. DSE explores the design space of your design by applying various optimization techniques and analyzing the results. The DSE Tcl script `dse.tcl` is located in the `<Quartus II installation directory>/common/tcl/apps/dse` directory on Windows and Linux operating systems.

DSE is a valuable tool to use in the late phases of your design cycle. You can take advantage of DSE’s capability to automatically sweep multiple options to close timing, minimize area, or reduce power consumption on a design that is nearing completion.

**DSE Concepts**

This section explains the concepts and terminology used with DSE.

**Exploration Space and Exploration Point**

Before DSE explores a design, DSE creates an exploration space, which consists of Analysis and Synthesis, and Fitter settings available in the Quartus II software. Each group of settings in an exploration space is referred to as a point. An exploration space contains one or more points. DSE traverses the points in the exploration space to determine optimal settings for your design.

**Seed and Seed Sweeping**

The Quartus II Fitter uses a seed to specify the starting value that randomly determines the initial placement for the current design. The seed value can be any non-negative integer value. Changing the starting value may or may not produce better fitting results. However, varying the value of the seed or seed sweeping allows the Quartus II software to determine an optimal value for the current design.

DSE extends Fitter seed sweeping in exploration spaces by providing a method for sweeping through compilation and Fitter parameters to find the best options for your design. You can run DSE in various exploration space modes, ranging from an exhaustive try-all-options-and-values mode to a mode that focuses on one parameter.
DSE Exploration

DSE compares all exploration point results with the results of a base compilation, generated from the initial settings that you specify in the original Quartus II project files. As DSE traverses all points in the exploration space, all settings not explicitly modified by DSE default to the base compilation setting. For example, if an exploration point turns on register retiming, but does not modify the Placement Effort Multiplier setting, the Placement Effort Multiplier setting defaults to the value you specified in the base compilation.

DSE performs the base compilation with the settings you specified in the original Quartus II project. These settings are restored after DSE traverses all points in the exploration space. DSE makes a copy of your base revision and uses this copy for changing the settings required to traverse through all other points in the chosen exploration space. Your base revision is not affected by DSE exploration.

DSE Support for Altera Device Families

DSE support varies across Altera device families. The Stratix® series of devices, the Cyclone® series of devices, and the Arria® series of devices can take advantage of all the available DSE optimization methods. The MAX® II device family supports a subset of DSE options.

Timing Analyzer Support

DSE supports both the Quartus II TimeQuest Timing Analyzer and the Quartus II Classic Timing Analyzer. You must set the timing analyzer with the Quartus II software prior to opening the project in DSE. After the timing analyzer is set, DSE performs the design exploration with the selected timing analyzer.

You can directly launch the TimeQuest Timing Analyzer from DSE if you have set the default timing analyzer to TimeQuest and have specified the timing constraints in a Synopsis Design Constraint File (.sdc).

Running DSE

You can use DSE in either the graphical user interface (GUI) or from a command line.

Using DSE from a Command Line

To run DSE from a command line, type the following command at the command prompt:

```
quartus_sh --dse --nogui [options]
```

You can run DSE with the following options:

- `archive`
- `concurrent-compiles [0..6]`
- `custom-file <filename>`
- `decision-column <"column name">`
- `exploration-space <"space">`
- `ignore-failed-base`
- `llr-restructuring`
- `lower-priority`
- `lsf-queue <queue name>`
For more information about DSE command line options, type the following command at the command prompt:

```
quartus_sh --help=dse
```

### Using the DSE Graphical User Interface

To run DSE with the GUI, either click **Launch Design Space Explorer** on the Tools menu in the Quartus II software, or type the following at the command prompt:

```
quartus_sh --dse
```

Figure 14–1 shows the DSE graphical user interface. The **Settings** tab is divided into two sections: **Project Settings** and **Exploration Settings**.
DSE Configuration File

Many options exist that allow you to customize the behavior of each DSE exploration. For example, you can specify seed values or a list of slave computers to be used for a Parallel DSE run. Each time you close the DSE GUI, it saves these values in a configuration file, dse.conf. The next time you launch the DSE GUI, it reads the values from dse.conf and restores the previous exploration settings.

Where the dse.conf file is stored varies depending on the operating system that launches DSE. Table 14–1 specifies the locations where dse.conf files are stored.

Table 14–1. DSE Configuration File Location

<table>
<thead>
<tr>
<th>OS</th>
<th>File Location (default)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%APPDATA%/Altera/dse.conf</td>
<td>If the variable %APPDATA% is not defined, the configuration file is saved to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>%HOME%\altera.quartus\dse.conf</td>
</tr>
<tr>
<td>Linux</td>
<td>~/.altera.quartus/dse.conf</td>
<td></td>
</tr>
</tbody>
</table>

Settings specified in the DSE command-line mode are not saved to a dse.conf configuration file.

For more information about the DSE GUI, launch the DSE GUI. On the Help menu, click Contents or press the F1 key.

DSE Flow

You can run DSE at any point in the design process. However, Altera recommends that you run DSE late in your design cycle when your focus is on optimizing performance and power. The results gained from different combinations of optimization options early in the design cycle may not persist over large changes in a design.

DSE runs the Quartus II software for every point in the exploration space. The Quartus II software always attempts to achieve all your timing requirements regardless of whether or not you are running DSE. The Exploration Settings you choose in DSE will determine the settings to be used for compilation. DSE does not change the behavior of the Quartus II software.

DSE provides a summary of results for all the compilations, and flags the best compilation run based on the exploration setting you have chosen. Specifying all timing requirements before you use DSE to explore your design is very important to ensure that DSE finds the optimal set of parameters for your design based on design criteria you set in your initial design.

You can change the initial placement configuration used by the Quartus II Fitter by varying the Fitter Seed value. You can enter seed values in the Seeds box of the DSE user interface.

To set the seed value in the Quartus II software, on the Assignments menu, click Settings and select Fitter Settings.

Compilation time increases as DSE exploration spaces become more comprehensive. Increased compilation time results from running several compilations and comparing the generated results with the original base compilation results.
For a typical design, varying only the seed value varies the \( f_{\text{MAX}} \) within a range of +/-5%. For example, when compiling with three different seeds, one-third of the time \( f_{\text{MAX}} \) does not change over the initial compilation, one-third of the time \( f_{\text{MAX}} \) improves by 5%, and one-third of the time \( f_{\text{MAX}} \) worsens by 5%.

**DSE Project Settings**

This section provides the following information about DSE project settings:

- Setting Up the DSE Work Environment
- Specifying the Revision
- Setting the Initial Seed
- Project Uses Quartus II Integrated Synthesis
- Restructuring LogicLock Regions

**Setting Up the DSE Work Environment**

From the DSE GUI, you can open a Quartus II project for a design exploration by clicking **Open Project** on the File menu and then browsing to your project. Clicking the Quartus II icon in the DSE GUI closes the DSE GUI and opens the project in the Quartus II software.

**Specifying the Revision**

You can specify the revision to be explored with the **Revision** field in the DSE GUI. The **Revision** field is populated after the Quartus II project has been opened.

If no revisions were created in the Quartus II project, the default revision, which is the top-level entity, is used. For more information, refer to the *Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*.

**Setting the Initial Seed**

To specify the seed that DSE uses for an exploration, specify a non-negative integer value in the **Seeds** box under **Project Settings** on the **Settings** tab. The seed value determines your design’s initial placement in a Quartus II compilation.

To specify a range of seeds, type the low end of the range followed by a hyphen, followed by the high end of the range. For example, 2-5 specifies that DSE uses the values 2, 3, 4, and 5 as seeds.

**Project Uses Quartus II Integrated Synthesis**

Ensure that you turn on the **Project Uses Quartus II Integrated Synthesis** option if you use Quartus II Integrated Synthesis to synthesize your design. The DSE explores several options that affect, and can help, the synthesis stage of compilation when this option is turned on.

For more information about integrated synthesis options, refer to the *Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*. 
Restructuring LogicLock Regions

The Allow LogicLock Region Restructuring option allows DSE to modify the properties of LogicLock regions in your design. DSE applies the Soft property to LogicLock regions to improve timing. In addition, DSE can remove LogicLock regions that negatively affect the performance of the design.

DSE makes a copy of your base revision and modifies the LogicLock region settings on the new copy to test whether timing improves with the Allow LogicLock Region Restructuring option. Your original revision remains intact.

DSE Exploration Settings

This section provides the following information about DSE exploration settings:

- Using DSE to Search for the Best Area
- Using DSE to Search for the Best Performance
- Using DSE to Search for the Lowest Power

Use the Exploration Settings list to select the type of exploration to perform.

Using DSE to Search for the Best Area

The Search for Best Area option uses a predefined exploration space that targets device utilization improvements for your design.

Using DSE to Search for the Best Performance

The Search for Best Performance option uses a predefined exploration space that targets performance improvements for your design. Depending on the device that your design targets, you can select up to five predefined exploration spaces: Low (Seed Sweep), Medium (Extra Effort Space), High (Physical Synthesis Space), Highest (Physical Synthesis with Retiming Space), and Selective (Selected Performance Optimizations). As you move from Low to Highest, the number of options explored by DSE increases, which causes compilation time to increase.

In an exploration for best performance, DSE works on reducing the worst magnitude slack, regardless of whether it is a hold slack or setup slack. During this process, DSE also takes into consideration all process corners before flagging one of the compilations as the best.

Effort Level

When you select Search for Best Performance under the Exploration Settings in the DSE GUI, you can select the effort level you wish to use to compile your design in DSE. The effort levels are Low (Seed Sweep), Medium (Extra Effort Space), High (Physical Synthesis Space), Highest (Physical Synthesis with Retiming Space) and Selective (Selected Performance Optimizations). DSE traverses the points in the exploration space, applies the settings to the design, and compares compilation results to determine the best settings for your design based on your chosen effort level. Search time increases proportionally with the breadth of the options being explored. The exploration space search time increases with the number, type, and combination of options DSE explores.
DSE offers the following exploration space types:

- **Seed Sweep**
- **Extra Effort Space**
- **Physical Synthesis Space**
- **Physical Synthesis with Retiming Space**
- **Selective (Selective Performance Optimizations)**

### Seed Sweep

Enter the seed values in the **Seeds** field in the DSE user interface. There are no “magic” seeds. The variation between seeds is truly random, any non-negative integer value is as likely to produce good results. DSE defaults to seeds 2, 3, 4, 5, and 6. The **Low (Seed Sweep)** option exploration space does not change your netlist.

The **Seeds** field accepts individual seed values, for example, 2, 3, 4, and 5, or seed ranges, for example, 2-5.

Each seed value you specify requires an additional compilation. For example, if you enter five seeds, the compilation time increases to 5 times the initial (or base) compilation time.

### Extra Effort Space

The **Extra Effort Space** effort level increases the Quartus II Fitter effort during placement and routing in addition to performing a seed sweep. The **Extra Effort Space** effort level does not change your netlist.

### Physical Synthesis Space

The **Physical Synthesis Space** effort level adds physical synthesis options such as register retiming and physical synthesis for combinational logic to the options included in the **Extra Effort Space** effort level. These netlist optimizations move registers in your design. Look-up tables (LUTs) are modified by these options. However, the design behavior is not affected by these options.

For more information about physical synthesis, refer to the **Netlist Optimizations and Physical Synthesis** chapter in volume 2 of the **Quartus II Handbook**.

### Physical Synthesis with Retiming Space

The **Physical Synthesis with Retiming Space** effort level includes all the options in the **Physical Synthesis Space** effort level, and it explores various Quartus II Integrated Synthesis optimization options and register retiming. Register retiming can move registers in your design.

The **Physical Synthesis with Retiming Space** effort level works only for designs that have been synthesized using Quartus II Integrated Synthesis.
Selective (Selective Performance Optimizations)

The Selective Performance Optimizations effort level combines a seed sweep with various performance Fitter settings to improve the timing of your design. The seed sweep is performed over a limited number of points in such a way that the base settings are not replicated. This is the recommended option for large designs where other spaces may be too large. Use this exploration space for first-time DSE searches on your designs to evaluate the range of results.

Table 14–2 shows the settings adjusted by each effort level.

<table>
<thead>
<tr>
<th>Optimization Options</th>
<th>Effort Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seed Sweep</td>
</tr>
<tr>
<td>Analysis and Synthesis Settings</td>
<td></td>
</tr>
<tr>
<td>Optimization Technique</td>
<td>—</td>
</tr>
<tr>
<td>Perform WYSIWYG primitive resynthesis</td>
<td>—</td>
</tr>
<tr>
<td>Fitter Settings</td>
<td>✓</td>
</tr>
<tr>
<td>Fitter seed</td>
<td>—</td>
</tr>
<tr>
<td>Increase PowerFit Fitter effort</td>
<td>—</td>
</tr>
<tr>
<td>Perform physical synthesis for combinational logic</td>
<td>—</td>
</tr>
<tr>
<td>Perform register retiming</td>
<td>—</td>
</tr>
</tbody>
</table>

Note to Table 14–2:
(1) For effort levels that include Quartus II Integrated Synthesis projects, DSE increases the synthesis effort.

Using DSE to Search for the Lowest Power

The Search for Lowest Power option uses a predefined exploration space that targets overall power improvements for your design. When Search for Lowest Power is selected, DSE automatically runs the PowerPlay Power Analyzer for each point in the space. You must ensure that the PowerPlay Power Analyzer is configured correctly to ensure accurate results. DSE issues a warning if the confidence level for any power estimate is low.

DSE Flow Options

You can control the configuration of DSE with the following options:

- Continue Exploration Even If Base Compilation Fails
- Skip Base Analysis and Compilation If Possible
- Stop Flow When Zero Failing Paths are Achieved
- Stop Flow After Time
- Report all Resource Usage Information
- Parallel DSE Information
- Create Revisions Without Compiling
Run Quartus II PowerPlay Power Analyzer During Exploration

Show Full Path to Project in Title Bar

**Continue Exploration Even If Base Compilation Fails**
With the **Continue Exploration Even If Base Compilation Fails** option turned on, DSE continues the exploration even when a design compilation error occurs. For example, if timing settings are not applied to your design, a DSE error occurs. To direct DSE to continue with the exploration instead of halting when an error occurs, turn on this option.

**Skip Base Analysis and Compilation If Possible**
The **Skip Base Analysis & Compilation if Possible** option allows DSE to skip the Analysis and Elaboration stage or the compilation of the base point if base point compilation results are available from a previous Quartus II compilation.

**Stop Flow When Zero Failing Paths are Achieved**
Instructs DSE to stop exploring the space after it encounters any point, including the base point, that has zero failing paths. DSE uses the failing path count reported in the **All Failing Paths report** column to make this decision.

**Stop Flow After Time**
Turn on **Stop Flow After Time** to stop further exploration after a specified number of days, hours, and/or minutes.

† Exploration time might exceed the specified value because DSE does not stop in the middle of a compilation.

**Report all Resource Usage Information**
Turn on **Report all Resource Usage Information** to include all resource information from the Quartus II Fitter reports in the DSE report. The **Report all Resource Usage Information** option allows you to compare resource utilization in one place, rather than comparing the Fitter report from multiple compilations. You may find this option useful if you are trying to optimize the design for the lowest use of a particular type of resource. Turn off **Report all Resource Usage Information** to include only logic elements and RAM blocks used.

**Archive All Compilations**
Turn on **Archive all Compilations** to create a Quartus II Archive File (.qar) for each compilation. These archive files are saved to the **dse** directory in the design’s working directory.

The result of each DSE run is saved as a .qar file in the **dse** subdirectory under your project directory. Each run is identified by a number. The best result of DSE run is saved with the name **best.qar**.

The **dse** directory also contains a spreadsheet (results.csv) that compares the results of all the individual runs in your DSE compilation.
Create Revisions Without Compiling
Turn on Create Revisions Without Compiling to create a Quartus II project revision for every combination of Quartus II software settings in the exploration space, without compiling the project. That is, DSE creates a revision for every combination of Analysis & Synthesis settings, Fitter settings, LogicLock region settings, and seed values in the exploration space. DSE creates $n_{\text{synthesis settings}} \times n_{\text{fitter settings}} \times n_{\text{LogicLock settings}} \times n_{\text{seeds}}$ revisions.

Run Quartus II PowerPlay Power Analyzer During Exploration
Turn on Run Quartus II PowerPlay Power Analyzer During Exploration to run the Quartus II PowerPlay Analyzer for every exploration performed by DSE. Using this option can help you debug your design and determine trade-offs between power requirements and performance optimization.

Show Full Path to Project in Title Bar
Shows, in the title bar of the DSE window, the full directory path to the project.

Changes to this option do not take effect until you restart DSE.

DSE Processing Commands
You can process design explorations with the following commands:

- Explore Space
- View Last DSE Report for Project
- Create a Revision from a DSE Point
- Open Project in TimeQuest Timing Analyzer
- Open Project in Quartus II

Explore Space
The Explore Space command directs DSE to begin the search of the exploration space.

View Last DSE Report for Project
The View Last DSE Report for Project command displays the DSE report generated by the most recent exploration of the project.

Create a Revision from a DSE Point
This command facilitates the creation of multiple revisions based on the same space point for further optimization within the Quartus II software. After you have performed a design exploration, you can use the Create a Revision from a DSE Point to create a new revision with the Quartus II settings of any exploration point. You can also use this command to merge the Quartus II settings of any exploration point with an existing revision.

Open Project in TimeQuest Timing Analyzer
The Open Project in TimeQuest Timing Analyzer command closes DSE and opens the current project revision in the TimeQuest Timing Analyzer.
Open Project in Quartus II

The Open Project in Quartus II command closes DSE and opens the current project revision in the Quartus II software.

Parallel DSE Information

This section covers the Parallel DSE option, which enables you to run an exploration on multiple computers concurrently. This feature increases the processing efficiency of design space exploration. You can access the settings for Parallel DSE from the Parallel DSE menu in the DSE GUI.

Computer Load Sharing Using Parallel DSE

DSE uses cluster computing technology to decrease exploration time when you click Distribute Compilations to Other Machines on the Parallel DSE menu. DSE uses multiple client computers to compile points in the specified exploration space.

Parallel DSE functions in one of the following modes:

- **Use LSF Resources**—DSE uses the PlatformLSF grid computing technology to distribute exploration space points to a computing network.

- **Use QSlave**—This function uses a Quartus II master process. DSE acts as a master and distributes exploration space points to client computers.

When you use the Distribute Compilations to Other Machines option, different exploration points in the exploration space are compiled on different slave client computers at the same time. Concurrent compilations requires a separate license for each instance of the Quartus II software being used to compile the design. Each compilation also might require licenses for any IP cores in the design. Therefore, the number of parallel distributed compilations can be limited to the number of licenses available for the Quartus II software or the IP core used in your design.

Parallel DSE Using LSF Resources

The easiest way to use distributed DSE technology is to submit the compilations to a preconfigured LSF cluster at your local site. For more information about LSF software, refer to www.platform.com, or contact your system administrator. To run Parallel DSE using LSF resources, on the Parallel DSE menu, click Configure Resources.

Parallel DSE Using a Quartus II Master Process

Before DSE can use computers in the local area network to compile points in the exploration space, you must create Quartus II software slave instances on the computers that will be used as clients. Type the following command at a command prompt on a client computer:

```
quartus_sh --qslave
```

Repeating this command on several computers creates a cluster of Quartus II software slaves for DSE to use. After you have created a set of Quartus II software slaves on the network, add the names of each slave computer in the QSlave tab of the Configure Resources dialog box.
To access the **Configure Resources** dialog box, on the Parallel DSE menu, click **Configure Resources**. To add resources, click the **QSlave** tab and click **Add** and type the client name. Click **OK**.

At the start of an exploration, DSE assumes the role of a Quartus II software master process and submits points to the slaves on the list to compile. If the list is empty, DSE issues an error and the search stops.

For more information about running and configuring Quartus II slaves, type the following command at the command prompt:

```
quartus_sh --help=qslave
```

Parallel DSE uses a protocol based on FTP to move files between the master and the slaves. By default, the qslave client listens to port number 1977 for communication with the master. If you are running a firewall on a computer that is running the qslave client, make sure you configure the firewall software such that it allows incoming and outgoing transmission control protocol (TCP) and user datagram protocol (UDP) packets on the port used by qslave.

You must set this configuration in every computer that is used as a slave in a distributed DSE environment.

You can change the default port number used by qslave by typing the following command at a command prompt:

```
quartus_sh --qslave port=<new_port_number>
```

You must use the same version of the Quartus II software to run the slave processes as you use to run DSE. To determine which Quartus II software version you are using to run DSE, select Help and click **About DSE**. Unexpected results can occur if you mix different Quartus II software versions when using the Parallel DSE feature.

When you are using ClearCase revision control software, Parallel DSE compilations launched within a ClearCase view might fail. ClearCase catches system I/O calls that can prevent communication between the DSE master and its slave computers. To avoid this problem, run Parallel DSE outside of the ClearCase environment.

**Concurrent Local Compilations**

To reduce compilation time, DSE can compile exploration points concurrently. The **Concurrent Local Compilations** option allows you to specify up to six concurrent compilations by choosing an integer value ranging from 0 through 6. You can use this option in conjunction with Parallel DSE. However, your system must have the appropriate resources and licenses to perform concurrent compilations, and distributed processing. Multiprocessor or multicore systems are recommended for concurrent local compilations.

**Concurrent Local Compilations** require a separate Quartus II software license for each concurrent compilation. For example, if you compile four concurrent compilations, you must have four licenses. Ensure that sufficient licenses are available before you choose a **Concurrent Local Compilations** value and start compilation.
You can use concurrent compilations and distributed compilations with other computer options at the same time if you use the QSlave approach for distributing compilations to other computers.

If you use LSF, all the jobs are submitted to the LSF system.

**Referenced Documents**

This chapter references the following documents:

- Managing Quartus II Projects chapter in volume 2 of the Quartus II Handbook
- Netlist Optimizations and Physical Synthesis chapter in volume 2 of the Quartus II Handbook
- Quartus II Integrated Synthesis chapter in volume 1 of the Quartus II Handbook
**Document Revision History**

Table 14–3 shows the revision history for this chapter.

### Table 14–3. Document Revision History  (Part 1 of 2)

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2009 v.9.1.0</td>
<td>Updated the following sections:</td>
<td>Updated for the Quartus II software version 9.1 release.</td>
</tr>
<tr>
<td></td>
<td>- Updated Figure 14–1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Added the following section:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “DSE Processing Commands” on page 14–10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Updated the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Introduction” on page 14–1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “DSE Support for Altera Device Families” on page 14–2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “DSE Exploration Settings” on page 14–6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “DSE Flow Options” on page 14–8</td>
<td></td>
</tr>
<tr>
<td>March 2009 v9.0.0</td>
<td>- Was chapter 12 in the 8.1.0 release.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Updated Table 14–2 and Table 14–1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Added the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Project Uses Quartus II Integrated Synthesis” on page 14–5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “DSE Exploration Settings” on page 14–6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Effort Level” on page 14–6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Updated the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Running DSE” on page 14–2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Setting Up the DSE Work Environment” on page 14–5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Seed Sweep” on page 14–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Physical Synthesis Space” on page 14–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Concurrent Local Compilations” on page 14–12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Deleted the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Ignore SignalTap and SignalProbe Settings</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Quartus II Integrated Synthesis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Search for Best Performance, Search for Best Area Options, or Search for Lowest Power Option</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated for the Quartus II software version 9.0 release.</td>
</tr>
</tbody>
</table>
### Table 14–3. Document Revision History  (Part 2 of 2)

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2008 v8.1.0</td>
<td>Changed to 8½” x 11” page size. No change to content.</td>
<td>Updated for the Quartus II software version 8.1 release.</td>
</tr>
<tr>
<td>May 2008 v8.0.0</td>
<td>Updated the following sections:</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>- “Search for Best Performance, Search for Best Area Options,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or Search for Lowest Power Option” on page 12–5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Using the DSE to Search for the Best Performance” on page 12–6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Physical Synthesis with Retiming Space” on page 12–7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- “Parallel DSE Information” on page 12–10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deleted the following sections:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Advanced Search Options</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Exploration Space</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Custom Space</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Area Optimization Space</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Change Decision Column</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Save Exploration Space to File</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Creating Custom Spaces for DSE</td>
<td></td>
</tr>
</tbody>
</table>

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