11. Reducing Compilation Time

QII52022-13.0.0

The Quartus® II software offers several features and techniques to help reduce compilation time.

This chapter describes techniques to reduce compilation time when designing for Altera® devices, and includes the following topics:

Compilation Time Optimization Techniques

The Analysis and Synthesis and Fitter modules consume the majority of time in a compilation. The Analysis and Synthesis module includes physical synthesis optimizations performed during synthesis, if you have turned on physical synthesis optimizations. The Fitter includes two steps, placement and routing, and also includes physical synthesis if you turned on the physical synthesis option with **Normal** or **Extra** effort levels. The **Flow Elapsed Time** section of the Compilation Report shows the duration of the Analysis and Synthesis and Fitter modules. The Fitter Messages report in the **Fitter** section of the Compilation Report displays the elapsed time for placement and routing processes.

Placement is the process of finding optimum locations for the logic in your design. Placement includes Quartus II pre-Fitter operations, which place dedicated logic such as clocks, PLLs, and transceiver blocks. Routing is the process of connecting the nets between the logic in your design. Finding better placement for the logic in your design requires more compilation time. Good logic placement allows you to more easily meet your timing requirements and makes your design easier to route.

Example 11–1 shows examples of messages with each time component in two-digit format, and days shown only if applicable:

Example 11–1.

```
Info: Fitter placement operations ending: elapsed time =
<days:hours:minutes:seconds>
Info: Fitter routing operations ending: elapsed time =
<days:hours:minutes:seconds>
```

Example 11–2 shows an info message displayed while the Fitter is running (including Placement and Routing). The Message window displays this message every hour to indicate Fitter operations are progressing normally.

Example 11–2.

```
Info: Placement optimizations have been running for 4 hour(s)
```

Compilation Time Advisor

A Compilation Time Advisor is available to help you to reduce compilation time. 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.

Strategies to Reduce the Overall Compilation Time

This section discusses strategies to reduce overall compilation time, including the following topics:

- "Using Parallel Compilation with Multiple Processors"
- "Using Incremental Compilation" on page 11–4
- "Using the Smart Compilation Setting" on page 11–4

Using Parallel Compilation with Multiple Processors

The Quartus II software can detect the number of processors available on a computer and use multiple processors to reduce compilation time. You can also control the number of processors used during a compilation on a per user basis. The Quartus II software can use up to 16 processors to run algorithms in parallel and reduce compilation time. The Quartus II software turns on parallel compilation by default to enable the software to detect available multiple processors. You can specify the maximum number of processors that the software can use if you want to reserve some of the available processors for other tasks.

Do not consider processors with Intel Hyper-Threading as 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, because it can increase runtimes.

The software does not necessarily use all the processors that you specify during a given compilation. Additionally, the software never uses more than the specified number of processors, enabling you to work on other tasks on your computer without it becoming slow or less responsive.

If you have partitioned your design and enabled parallel compilation, the Quartus II software can use different processors to compile those partitions simultaneously during Analysis and Synthesis. This can cause higher peak memory usage during Analysis and Synthesis.

You can reduce the compilation time by up to 10% on systems with two processing cores and by up to 20% on systems with four cores. With certain design flows in which timing analysis runs alone, multiple processors can reduce the time required for timing analysis by an average of 10% when using two processors. This reduction can reach an average of 15% when using four processors.

The actual reduction in compilation time when using incremental compilation partitions depends on your design and on the specific compilation settings. For example, compilations with multi-corner optimization turned on benefit more from using multiple processors than do compilations without multi-corner optimization. The runtime requirement is not reduced for some other compilation goals, such as Analysis and Synthesis. The Fitter (`quartus_fit`) and the Quartus II TimeQuest Timing Analyzer (`quartus_sta`) stages in the compilation can, in certain cases, benefit from the use of multiple processors. The **Flow Elapsed Time** panel of the Compilation Report shows the average number of processors for these stages. The Parallel Compilation panel of the appropriate report, such as the Fitter report, shows a more detailed breakdown of processor usage. This panel is displayed only if parallel compilation is enabled.

Parallel compilation is available for Arria® series, Cyclone®, MAX® II, MAX V (limited support), and Stratix® series devices.

For more information, refer to *Processing Page (Options Dialog Box)* in Quartus II Help.

For more information about how to control the number of processors used during compilation for a specific project, refer to *Compilation Process Settings Page (Settings Dialog Box)* in Quartus II Help.

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 `16`.

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

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS ALL ↵
```

The use of 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 compilations using a different number of processors is the compilation time.

Using Incremental Compilation

The incremental compilation feature can accelerate design iteration time by up to 70% for small design changes, and helps you reach design timing closure more efficiently. You can speed up design iterations 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 parts of your design untouched to preserve performance.

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

In the standard incremental compilation design flow, you can divide the top-level design into partitions, which the software can compile and optimize in the top-level Quartus II project. You can preserve fitting results and performance for completed partitions while other parts of your design are changing. Incremental compilation reduces the compilation time for each design iteration because the software does not recompile the unchanged partitions in your design.

The incremental compilation feature facilitates team-based design flows by enabling designers to create and optimize design blocks independently, when necessary, and supports third-party IP integration.

For more 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 more information about creating multiple netlist files in third-party tools for use with incremental compilation, refer to the appropriate chapter in *Section IV. Synthesis* in volume 1 of the *Quartus II Handbook*.

For more information about incremental compilation, refer to *About Incremental Compilation* in Quartus II Help.

Using the Smart Compilation Setting

Not all compilation processes are required for when recompiling your design. Smart compilation skips unnecessary Compiler stages, such as Analysis and Synthesis. This feature is different from incremental compilation, which can compile parts of your design while preserving results for unchanged parts.

This setting is especially useful when you perform multiple compilations during the optimization phase of your design process. Smart compilation requires more disk space than regular compilation. 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**.

For more information on how to use smart compilation, refer to *Compilation Process Settings Page (Settings Dialog Box)* in Quartus II Help.

Reducing Synthesis Time and Synthesis Netlist Optimization Time

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

Settings to Reduce Synthesis Time and Synthesis Netlist Optimization Time

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. When using Quartus II Integrated Synthesis, you can also enable specific options in the Physical Synthesis Optimizations window before performing Analysis and Synthesis. Netlist optimizations can cause the Analysis and Synthesis module to take much longer to run. Read the Analysis and Synthesis messages to determine how much time these optimizations take. The compilation time spent in Analysis and Synthesis is usually short 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.

Turn off settings that are not useful. In general, if you carry over compilation settings from a previous project, evaluate all settings and keep only those that you need.

Use Appropriate Coding Style to Reduce Synthesis Time

Your HDL coding style can also affect the synthesis time. For example, if you want to infer RAM blocks from your code, you must follow the guidelines for inferring RAMs. If RAM blocks are not inferred properly, the software implements those blocks as registers.

If you are trying to infer a large memory block, the software consumes more resources in the FPGA. This can cause routing congestion and increasing compilation time significantly. If you see high routing utilizations in certain blocks, it is a good idea to review the code for such blocks.

For more information about coding guidelines, refer to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook.*

Reducing Placement Time

The time required to place a design depends on two factors: the number of ways the logic in your 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 the unchanged parts of your 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, ensure that it does not increase routing time and negate the overall time reduction.

Fitter Effort Setting

The highest Fitter effort setting, **Standard Fit**, requires the most runtime, but does not always yield a better result than using the default **Auto Fit**. For designs with very tight timing requirements, both **Auto Fit** and **Standard Fit** use the maximum effort during optimization. Altera recommends using **Auto Fit** for reducing compilation time. If you are certain that your design has only easy-to-meet timing constraints, you can select **Fast Fit** for an even greater runtime savings.

Placement Effort Multiplier Settings

You can control the amount of time the Fitter spends in placement by reducing 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.

Physical Synthesis Effort Settings

Physical synthesis options enable you 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 reduce compilation time. For example, if some or all of the physical synthesis algorithm information messages display an improvement of 0 ps, turning off physical synthesis can reduce compilation time.

You also can use the **Physical synthesis effort** setting on the **Physical Synthesis Optimizations** page to reduce the amount of extra compilation time used by these optimizations.

The **Fast** setting directs the Quartus II software to use a lower level of physical synthesis optimization. Compared to the **Normal** physical synthesis effort level, using the **Fast** setting can cause a smaller increase in compilation time. However, the lower level of optimization can result in a smaller increase in design performance.

Preserving Placement with Incremental Compilation

Preserving information about previous placements can make future placements faster. The incremental compilation feature provides an easy-to-use method for preserving placement results. For more information, refer to "Using Incremental Compilation" on page 11–4.

Reducing Routing Time

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

- Check for routing congestion.
- Turn off **Fitter Aggressive Routability Optimization**.
- Use incremental compilation to preserve routing information for parts of your design.

Identifying Routing Congestion in the Chip Planner

To identify areas of routing congestion in your design, open the Chip Planner from the Tools menu. To view the routing congestion in the Chip Planner, double-click the **Report Routing Utilization** command in the **Tasks** list. Click **Preview** in the **Report Routing Utilization** dialog box to preview the default congestion display. Change the **Routing utilization type** to display congestion for specific resources. The default display uses dark blue for 0% congestion and red for 100%. Adjust the slider for **Threshold percentage** to change the congestion threshold level.

Even if average congestion is not very high, your design may have areas where congestion is very high in a specific type of routing. You can use the Chip Planner to identify areas of high congestion for specific interconnect types. 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 your design. If the routing time remains the same, the time is a characteristic of your 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.

Sometimes, routing congestion may be a result of the HDL coding style used in your design. After you identity congested areas using the Chip Planner, review the HDL code for the blocks placed in those areas to determine whether you can reduce interconnect usage by code changes.

The Quartus II compilation messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, could be an indication that it might be difficult to fit your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, are likely to have increased chances of not getting a valid fit.

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

Preserving Routing with Incremental Compilation

Preserving the previous routing results for part of your design can reduce future routing time. Incremental compilation provides an easy-to-use methodology that preserves placement and routing results. For more information, refer to "Using Incremental Compilation" on page 11–4 and the references listed in the section.

Reducing Static Timing Analysis Time

If you are performing timing-driven synthesis, the Quartus II software runs the TimeQuest analyzer during Analysis and Synthesis. The Quartus II Fitter also runs the TimeQuest analyzer during placement and routing. If there are incorrect constraints in the Synopsys Design Constraints File (**.sdc**), the Quartus II software may spend unnecessary time processing constraints several times.

- If you do not specify false paths and multicycle paths in your design, the TimeQuest analyzer may analyze paths that are not relevant to your design.
- If you redefine constraints in the **.sdc** files, the TimeQuest analyzer may spend additional time processing them. To avoid this situation, look for indications that Synopsis design constraints are being redefined in the compilation messages, and update the **.sdc** file.
- Ensure that you provide the correct timing constraints to your design, because the software cannot assume design intent, such as which paths to consider as false paths or multicycle paths. When you specify these assignments correctly, the TimeQuest analyzer skips analysis for those paths, and the Fitter does not spend additional time optimizing those paths.

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 must run other tasks at the same time.

For more information about setting process priority, refer to *Processing Page (Options Dialog Box)* in Quartus II Help.

Document Revision History

Table 11–1 shows the revision history for this chapter.

Table 11–1. Document Revision History (Part 1 of 2)

Date	Version	Changes
May 2013	13.0.0	■ Removed the “Limit to One Fitting Attempt”, “Using Early Timing Estimation”, “Final Placement Optimizations”, and “Using Rapid Recompile” sections. ■ Updated “Placement Effort Multiplier Settings” section. ■ Updated “Identifying Routing Congestion in the Chip Planner” section. ■ General editorial changes throughout the chapter.
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	■ Updated “Using Parallel Compilation with Multiple Processors”. ■ Updated “Identifying Routing Congestion in the Chip Planner”. ■ General editorial changes throughout the chapter.

Table 11–1. Document Revision History (Part 2 of 2)

Date	Version	Changes
December 2010	10.1.0	■ Template update. ■ Added details about peak and average interconnect usage. ■ Added new section “Reducing Static Timing Analysis Time”. ■ Minor changes throughout chapter.
July 2010	10.0.0	Initial release.

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