Introduction to the Quartus® II Software

Version 9.1

QUARTUS® II

Introduction to the
Quartus® II
Software

QUARTUS®II

Contents

Preface

The Altera® Quartus® II design software is the most comprehensive environment available for system-on-a-programmable-chip (SOPC) design. This manual is designed for the novice Quartus II software user and provides an overview of the capabilities of the Quartus II software in programmable logic design. It is not, however, intended to be an exhaustive reference manual for the Quartus II software. Instead, it is a guide that explains the features of the software and how these can assist you in FPGA and CPLD design. This manual is organized into a series of specific programmable logic design tasks. Whether you use the Quartus II graphical user interface, other EDA tools, or the Quartus II command-line interface, this manual guides you through the features that are best suited to your design flow.

The first two chapters give an overview of the major graphical user interface, EDA tool, and command-line interface design flows. Each subsequent chapter begins with an introduction to the specific purpose of the chapter, and leads you through an overview of each task flow. In addition, the manual refers you to other resources that are available to help you use the Quartus II software, such as Quartus II online Help and the Quartus II interactive tutorial, application notes, white papers, and other documents and resources that are available on the Altera website.

Use this manual to learn how the Quartus II software can help you increase productivity and shorten design cycles; integrate with existing programmable logic design flows; and achieve design, performance, and timing requirements quickly and efficiently.

Documentation Conventions

The *Introduction to the Quartus II Software* manual uses the following conventions to make it easy for you to find and interpret information.

Typographic Conventions

Quartus II documentation uses the typographic conventions shown in the following table:

Visual Cue	Meaning
Bold Initial Capitals	Command names; dialog box, page, and tab titles; and button names are shown in bold, with initial capital letters. For example: **Find Text** command, **Save As** dialog box, and **Start** button.
bold	Directory, project, disk drive, file names, file extensions, software utility and software executable names; file name extensions, and options in dialog boxes are shown in bold. Examples: **quartus** directory, **d:** drive, **license.dat** file.
Initial Capitals	Keyboard keys, user-editable application window fields, window names, view names, and menu names are shown with initial capital letters. For example: Delete key, the Options menu.
"Subheading Title"	Subheadings within a manual section are enclosed in quotation marks. In manuals, titles of Help topics are also shown in quotation marks.
Italic Initial Capitals	Help categories, manual titles, section titles in manuals, and application note and brief names are shown in italics with initial capital letters. For example: *Introduction to the Quartus II Software*.
italics	Variables are enclosed in angle brackets (< >) and shown in italics. For example: *<file name>*, *<DVD drive>*.
`Courier font`	Anything that must be typed exactly as it appears is shown in Courier. For example: `\quartus\bin\lmutil lmhostid`.
↵	Enter or return key.
■	Bullets are used in a list of items when the sequence of the items is not important.
👣	The feet show you where to go for more information on a particular topic.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.

Terminology

The following table shows terminology that is used throughout the *Introduction to the Quartus II Software* manual:

Term	Meaning
"click"	Indicates a quick press and release of the left mouse button. It also indicates that you need to use a mouse or key combination to start an action.
"double-click"	Indicates two clicks in rapid succession.
"select"	Indicates that you need to highlight text and/or objects or an option in a dialog box with a key combination or the mouse. A selection does not start an action. For example: Select **Chain Description File**, and then click **OK**.
"point"	Indicates that you need to position the mouse pointer, without clicking, at an appropriate location on the screen, such as a menu or submenu. For example: On the Help menu, point to **Altera on the Web**, and then click **Quartus II Service Request**.
turn on/turn off	Indicates that you must click a check box to turn a function on or off.

Chapter One

Design Flow
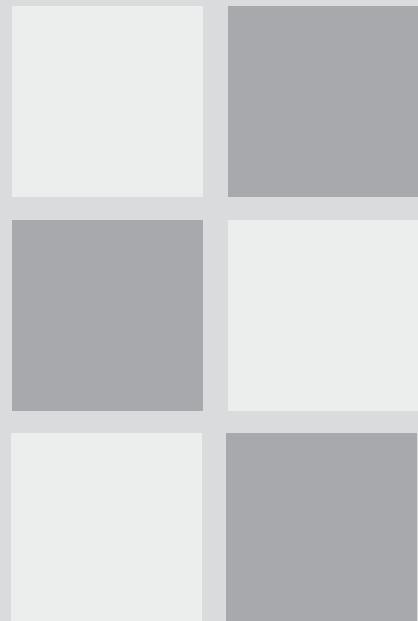
1

Introduction

The Altera Quartus II design software provides a complete, multiplatform design environment that easily adapts to your specific design needs. It is a comprehensive environment for system-on-a-programmable-chip (SOPC) design. The Quartus II software includes solutions for all phases of FPGA and CPLD design (Figure 1).

Figure 1. Quartus II Design Flow
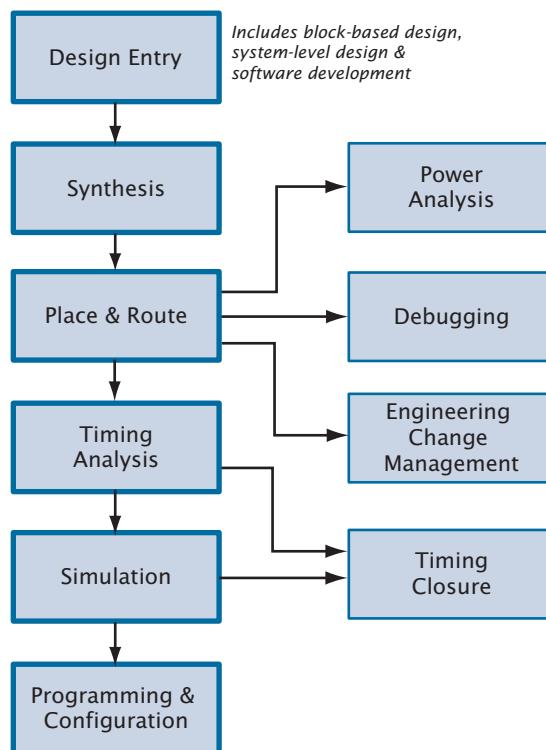
In addition, the Quartus II software allows you to use the Quartus II graphical user interface and command-line interface for each phase of the design flow. You can use one of these interfaces for the entire flow, or you can use different options at different phases.

Graphical User Interface Design Flow

You can use the Quartus II software graphical user interface to perform all stages of the design flow. Figure 2 shows the Quartus II GUI as it appears when you first start the software.

Figure 2. Quartus II Graphical User Interface
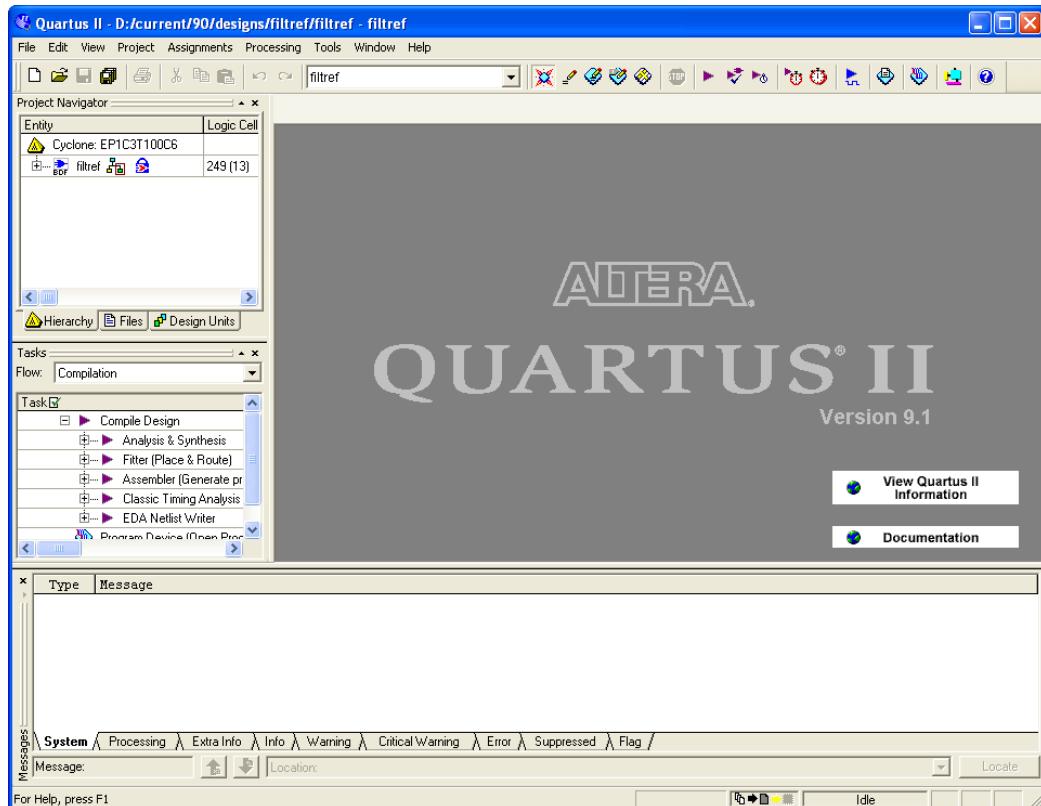
The Quartus II software includes a modular Compiler. The Compiler includes the following modules (modules marked with an asterisk are optional during a compilation, depending on your settings):

- Analysis & Synthesis
- Partition Merge*

- Fitter
- Assembler*
- TimeQuest Timing Analyzer*
- Design Assistant*
- EDA Netlist Writer*
- HardCopy® Netlist Writer*

To run all Compiler modules as part of a full compilation, on the Processing menu, click **Start Compilation**. You can also run each module individually by pointing to **Start** on the Processing menu, and then clicking the command for the module you want to start. You can also run some of the Compiler modules incrementally. See "Top-Down Incremental Compilation Flow" on page 12 for more information.

In addition, you can use the Tasks window to start Compiler modules individually (Figure 3). The Tasks window also allows you to open the settings file or report file for the module, or to start other tools related to each stage in a flow.

Figure 3. Tasks Window
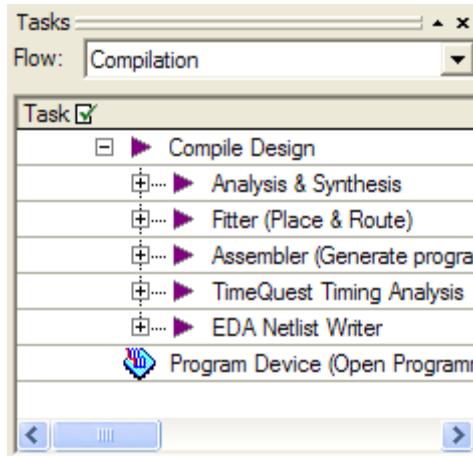
The Quartus II software also provides other predefined compilation flows that you can use with commands on the Processing menu. Table 1 lists the commands for some of the most common compilation flows.

Table 1. Commands for Common Compiler Flows

Flow	Description	Quartus II Command from Processing Menu
Full compilation flow	Performs a full compilation of the current design.	**Start Compilation** command
Compilation and simulation flow	If the simulation mode is timing, the flow performs a full compilation and then a simulation of the current design. If the simulation mode is functional, the flow runs only the **Generate Functional Simulation Netlist** command and then simulates the current design.	**Start Compilation and Simulation** command
SignalProbe™ flow	Routes user-specified signals to output pins without affecting the existing fitting in a design, so that you can debug signals without completing a full compilation.	**Start SignalProbe Compilation** command
Early timing estimate flow	Performs a partial compilation, but stops and generates early timing estimates before the Fitter is complete.	**Start Early Timing Estimate** command

For Information About	Refer To
Using compilation flows	"About Compilation Flows" in Quartus II Help

The following steps describe the basic design flow for using the Quartus II graphical user interface:

1. To create a new project and specify a target device or device family, on the File menu, click **New Project Wizard**.

2. Use the Text Editor to create a Verilog HDL, VHDL, or Altera Hardware Description Language (AHDL) design.

3. Use the Block Editor to create a block diagram with symbols that represent other design files, or to create a schematic.

4. Use the **MegaWizard® Plug-In Manager** to generate custom variations of megafunctions and IP functions to instantiate in your design, or create a system-level design by using SOPC Builder or DSP Builder.

5. Specify any initial design constraints using the Assignment Editor, the Pin Planner, the **Settings** dialog box, the Floorplan Editor, the Design Partitions window, or the Design Partition Planner.

6. (Optional) Perform an early timing estimate to generate early estimates of timing results before fitting.

7. Synthesize the design with Analysis & Synthesis.

8. (Optional) If your design contains partitions and you are not performing a full compilation, merge the partitions with partition merge.

9. (Optional) Generate a functional simulation netlist for your design and perform a functional simulation with the Simulator.

10. Place and route the design with the Fitter.

11. Perform a power estimation and analysis with the PowerPlay Power Analyzer.

12. Use the Simulator to perform timing simulation for the design.

13. Use the TimeQuest Timing Analyzer to analyze the timing of your design.

14. (Optional) Use physical synthesis, the Chip Planner, LogicLock™ regions, and the Assignment Editor to correct timing problems.

15. Create programming files for your design with the Assembler, and then program the device with the Programmer and Altera programming hardware.

16. (Optional) Debug the design with the SignalTap® II Logic Analyzer, an external logic analyzer, the SignalProbe feature, or the Chip Planner.

17. (Optional) Manage engineering changes with the Chip Planner, the Resource Property Editor, or the Change Manager.

EDA Tool Design Flow

The Quartus II software allows you to use the EDA tools you are familiar with for various stages of the design flow. Figure 4 shows the EDA tool design flow.

Figure 4. EDA Tool Design Flow
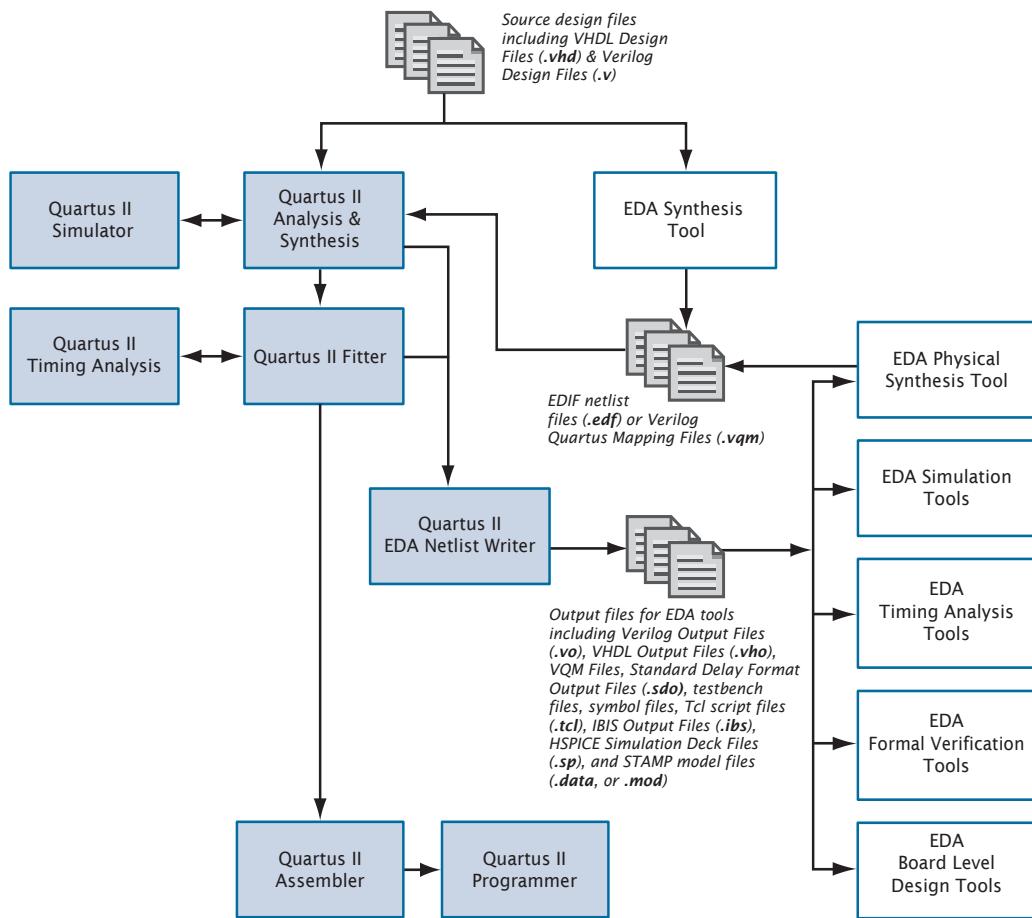
Table 2 shows the EDA tools that are supported by the Quartus II software, and indicates which EDA tools have NativeLink® support. NativeLink technology facilitates the seamless transfer of information between the Quartus II software and other EDA tools, and allows you to run the EDA tool automatically from within the Quartus II software.

Table 2. EDA Tools Supported by the Quartus II Software

Function	Supported EDA Tools	NativeLink Support
Design Entry & Synthesis	Mentor Graphics® LeonardoSpectrum	✓
	Mentor Graphics Precision RTL Synthesis	✓
	Mentor Graphics ViewDraw	
	Synopsys Design Compiler	
	Synopsys Synplify	✓
Simulation	Cadence NC-Verilog	✓
	Cadence NC-VHDL	✓
	Mentor Graphics ModelSim®	✓
	Mentor Graphics ModelSim-Altera	✓
	Synopsys VCS MX	✓
	Synopsys VCS	✓
	Active-HDL	✓
Timing Analysis	Mentor Graphics Tau (through Stamp)	
	Synopsys PrimeTime	✓
Board-Level Design	Hyperlynx (through Signal Integrity IBIS)	
	XTK (through Signal Integrity IBIS)	
	ICX (through Signal Integrity IBIS)	
	SpectraQuest (through Signal Integrity IBIS)	
	Mentor Graphics Symbol Generation (Viewdraw)	
Formal Verification	Cadence Encounter Conformal	
Physical Synthesis	Magma Design Automation PALACE	✓
	Synplicity Amplify	
	Precision Physical Synthesis	

To specify which EDA tools you want to use on the **EDA Tool Settings** page of the **Settings** dialog box, click **Settings** on the **Assignments** menu (Figure 5).

Figure 5. EDA Tool Settings Page of the Settings Dialog Box
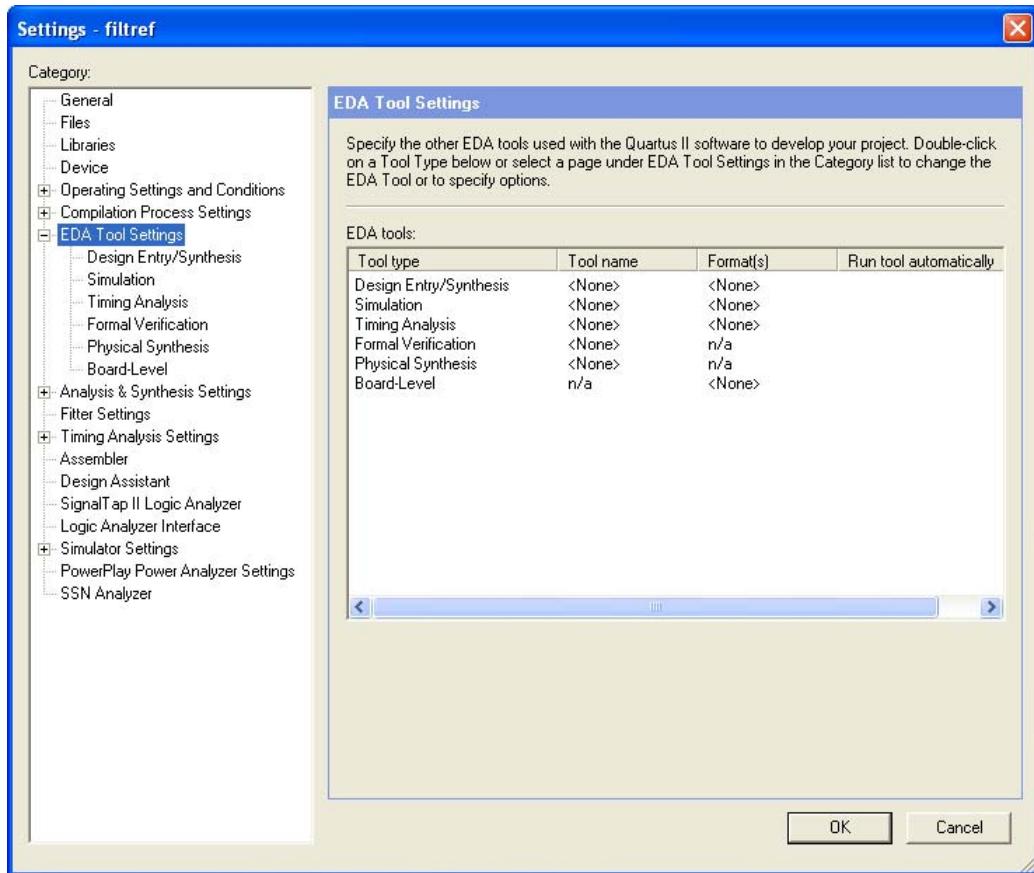
The pages under **EDA Tool Settings** provide additional options for each type of EDA tool.

The following steps describe the basic design flow for using other EDA tools with the Quartus II software. Refer to Table 2 on page 8 for a list of the supported EDA tools.

1. Create a new project and specify a target device or device family.

2. Specify which EDA design entry, synthesis, simulation, timing analysis, board-level verification, formal verification, and physical synthesis tools you are using with the Quartus II software, and specify additional options for those tools.

3. Create a Verilog HDL or VHDL design file with a standard text editor or use the **MegaWizard Plug-In Manager** to create custom variations of megafunctions.

4. Synthesize your design with one of the Quartus II-supported EDA synthesis tools, and generate an EDIF netlist file (**.edf**) or a Verilog Quartus Mapping File (**.vqm**).

5. (Optional) Perform functional simulation on your design with one of the Quartus II-supported simulation tools.

6. Compile your design with the Quartus II software. Run the EDA Netlist Writer to generate output files for use with other EDA tools.

7. (Optional) Perform timing analysis and simulation on your design with one of the Quartus II-supported EDA timing analysis or simulation tools.

8. (Optional) Perform formal verification with one of the Quartus II-supported EDA formal verification tools to make sure that Quartus post-fit netlist is equivalent to that of the synthesized netlist.

9. (Optional) Perform board-level verification with one of the Quartus II-supported EDA board-level verification tools.

10. (Optional) Perform physical synthesis with one of the Quartus II-supported EDA physical synthesis tools.

11. Program the device with the Programmer and Altera hardware.

For Information About	Refer To
Using the Quartus II software with Synopsys Synplify software	*Synopsys Synplify Support* chapter in volume 1 of the *Quartus II Handbook*
Using the Quartus II software with Mentor Graphics LeonardoSpectrum software	*Mentor Graphics LeonardoSpectrum Support* chapter in volume 1 of the *Quartus II Handbook*
Using the Quartus II software with Mentor Graphics Precision RTL Synthesis software	*Mentor Graphics Precision Synthesis Support* chapter in volume 1 of the *Quartus II Handbook*
Using the Quartus II software with Mentor Graphics ModelSim software	*Mentor Graphics ModelSim Support* chapter in volume 3 of the *Quartus II Handbook*

For Information About	Refer To
Using the Quartus II software with Synopsys VCS software	*Synopsys VCS Support* chapter in volume 3 of the *Quartus II Handbook*
Using the Quartus II software with Cadence NC-Sim software	*Cadence NC-Sim Support* chapter in volume 3 of the *Quartus II Handbook*
Using the Quartus II software with Aldec Active-HDL software	*Aldec Active-HDL Support* chapter in volume 3 of the *Quartus II Handbook*
Using the Quartus II software with Synopsys PrimeTime software	*Synopsys PrimeTime Support* chapter in volume 3 of the *Quartus II Handbook*
Using the Quartus II software with Cadence Encounter Conformal software	*Cadence Encounter Conformal Support* in volume 3 of the *Quartus II Handbook*

Design Methodologies and Planning

When you are creating a new design, it is important to consider the design methodologies the Quartus II software offers, including top-down or bottom-up incremental compilation design flows, and block-based design flows. You can use these design flows with or without EDA design entry and synthesis tools.

Top-Down and Bottom-Up Design Methodologies

The Quartus II software supports both top-down and bottom-up compilation flows. With top-down compilation, one designer or project lead compiles the entire design in the software. Different designers or IP providers can design and verify different parts of the design, and the project lead can add design entities to the project as they are completed. However, the project lead compiles and optimizes the top-level project as a whole. Completed parts of the design can have fitting results and performance fixed as other parts of the design change.

Bottom-up design flows allow individual designers to complete the optimization of their design in separate projects and then integrate each lower-level project into one top-level project. Designers of lower-level blocks can export the optimized netlist for their design, along with a set of assignments, such as LogicLock regions. Then the project lead imports each

design block as a design partition in a top-level project. In this case, the project lead must provide guidance to designers of lower-level blocks to ensure that each partition uses the appropriate device resources.

A top-down flow is generally simpler to perform than its bottom-up counterpart. For example, the need to import and export lower-level designs is eliminated. Second, a top-down approach provides the design software with information about the entire design so it can perform global optimizations. In the bottom-up design methodology, you must perform resource balancing and time budgeting because the software does not have any information about the other partitions in the top-level design when it compiles individual lower-level partitions.

Top-Down Incremental Compilation Flow

Incremental compilation allows you to preserve design performance and save compilation time by reusing previous compilation results and ensuring that only the parts of the design that have been modified are recompiled. The top-down incremental compilation flow can help you to improve timing by allowing you to change the placement of only the critical elements of the design while processing the other design partitions, or allowing you to specify the placement only for designated portions of the design so that the Compiler can automatically optimize the rest of the design.

In the incremental compilation flow, you assign an instance of a design entity to a design partition. You then assign the partitions to a physical location on the device by using the Chip Planner and the LogicLock feature, and perform a full compilation of the design. During compilation, the Compiler saves synthesis and fitting results in the project database. After the first compilation, if you make additional changes to the design, only the partitions that have changed require recompilation. When you finish making design changes and you perform a full incremental compilation, the Quartus II software merges all partitions together.

Because incremental compilation prevents the Compiler from optimizing across design partition boundaries, the Compiler may not be able to perform as many optimizations for area and timing as would be possible with standard compilation. To obtain best results for area and timing, register the inputs and outputs of design partitions, try to keep the number of design

partitions to a reasonable amount, avoid having too many critical paths that go beyond partition boundaries, and avoid having partitions that are smaller than 1,000 logic elements or Adaptive Logic Modules (ALMs).

For more information on assigning partitions and other stages of the incremental compilation flow, see the following sections:

■   "Creating Design Partitions" on page 57 in Chapter 4, "Constraint Entry."
■   "Using Incremental Compilation" on page 86 in Chapter 6, "Place and Route."
■   "Using LogicLock Regions in Top-Down Incremental Compilation Flows" on page 105 in Chapter 7, "Block-Based Design."
■   "Using Incremental Compilation to Achieve Timing Closure" on page 140 in Chapter 10, "Timing Closure."
■   "Logic Analysis with Incremental Compilation" on page 174 in Chapter 13, "Debugging."

Bottom-Up Incremental Compilation Flow

In a bottom-up incremental compilation design flow, you can design and optimize each module independently, integrate all optimized modules in a top-level design, and then verify the overall design. Each module has a separate netlist, which is incorporated after synthesis and optimization into the top-level design. Each module in the top-level design does not affect the performance of the other modules. The general block-based design flow concepts can be used in modular, hierarchical, incremental, and team-based design flows.

You can use EDA design entry and synthesis tools in the block-based design flow to design and synthesize individual modules, and then incorporate the modules into a top-level design in the Quartus II software, or completely design and synthesize a block-based design in EDA design entry and synthesis tools. For more information on the block-based design flow, refer to "Chapter 7: Block-Based Design" on page 99.

For Information About	Refer To
Using Quartus II incremental compilation	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" in Quartus II Help
	"Module 7: Incremental Compilation" in the Quartus II Interactive Tutorial

Chapter
Two

Design Entry
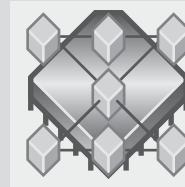
2

Introduction

A Quartus II project includes all of the design files, software source files, and other related files necessary for the eventual implementation of a design in a programmable logic device. You can use the Quartus II Block Editor, Text Editor, **MegaWizard Plug-In Manager**, and EDA design entry tools to create design files that include Altera megafunctions, library of parameterized modules (LPM) functions, and intellectual property (IP) functions. Figure 1 shows the design entry flow.

Figure 1. Design Entry Flow
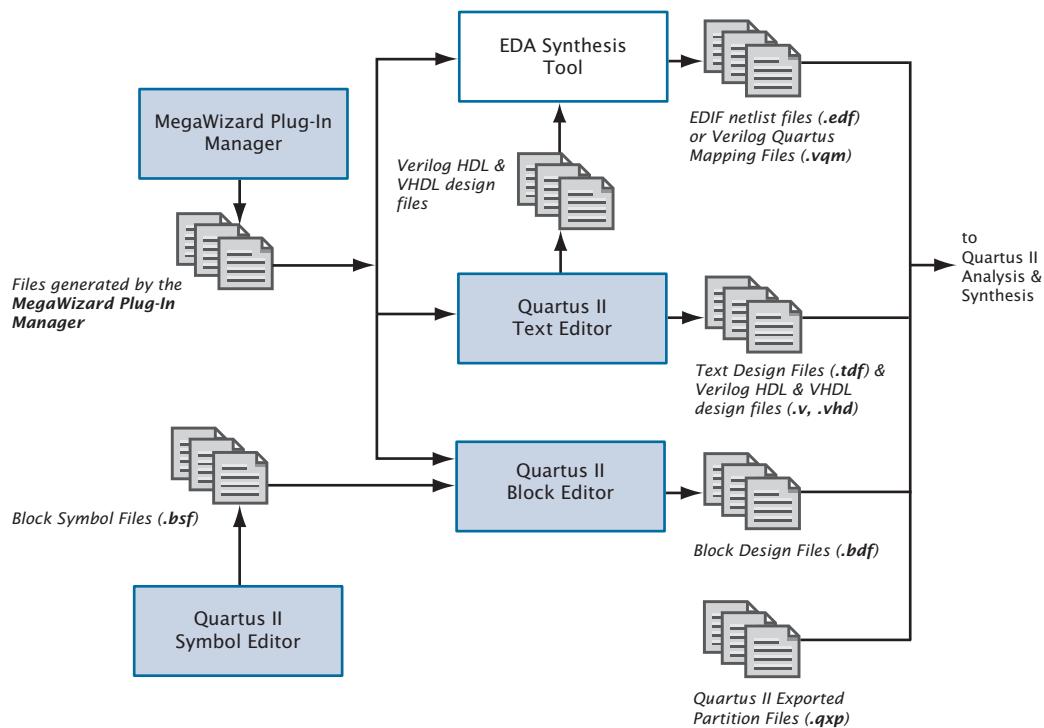
The Quartus II software also supports system-level design entry flows with the Altera SOPC Builder and DSP Builder software. For more information about these methods, refer to "Chapter 16: System-Level Design" on page 199.

Creating a Project

You can create a new project by clicking **New Project Wizard** on the File menu. When creating a new project, you specify the working directory for the project, assign the project name, and designate the name of the top-level design entity. You can also specify which design files, other source files, user libraries, and EDA tools you want to use in the project, as well as the target device. Table 1 lists the project and settings files for a Quartus II project.

Table 1. Quartus II Project Files

File Type	Description
Quartus II Project File (**.qpf**)	Specifies the version of the Quartus II software used to create the project and specifies all revisions of the project.
Quartus II Settings File (**.qsf**)	Contains all assignments you made with the Assignment Editor, Chip Planner, **Settings** dialog box, Tcl scripts, or Quartus II executables. There is one Quartus II Settings File for each revision of the project.
Quartus II IP File (**.qip**)	Contains a list of all of the files required for a project that includes an Altera MegaCore function. The Quartus II IP File allows you to add a custom MegaCore function variation to the project by adding only the Quartus II IP File, rather than adding all the necessary files individually. A separate Quartus II IP File exists for each individual custom MegaCore function variation.
Synopsys Design Constraints File (**.sdc**)	Contains timing constraints in the industry-standard Synopsys Design Constraints format required by the TimeQuest Timing Analyzer. The constraints in a Synopsys Design Constraints File are written in Tcl.
Quartus II Workspace File (**.qws**)	Contains user preferences and other information such as the positions of windows and open files.
Quartus II Default Settings File (**.qdf**)	Located in the **\\<*Quartus II system directory*>\\bin** directory and contains all the global default project settings. These settings are overridden by the settings in the Quartus II Settings File.

Once you have created a project, you can add and remove design files and other files from the project in the **Files** page of the **Settings** dialog box. During Analysis & Synthesis, the Quartus II software processes the files in the order they appear in the **Files** page.

You can also copy an entire project to a new directory by clicking **Copy Project** on the Project menu. This command causes the Quartus II software to copy the project design database files, design files, settings files, and report files to a new directory and then open the project in the new directory, creating the directory if it does not already exist.

The Project Navigator displays information related to the current revision and provides a graphical representation of the project hierarchy, files, and design units, and shortcuts to various menu commands. You can also customize the information shown in the Project Navigator by right-clicking the information and then clicking **Customize Columns**.

Figure 2. Project Navigator Window
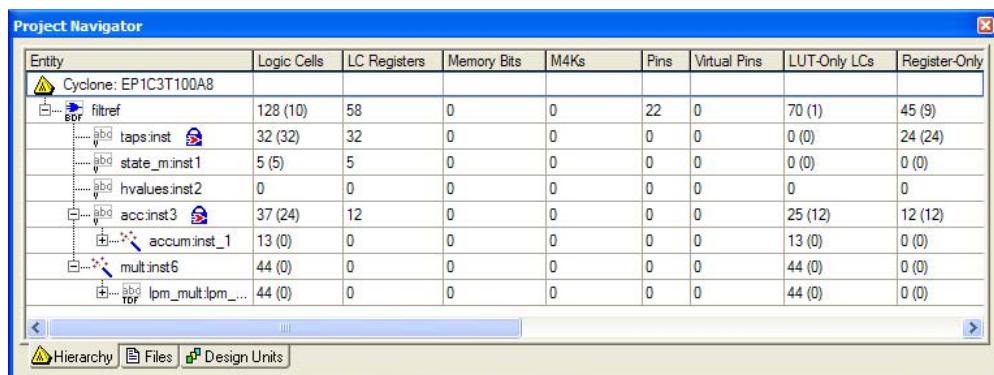
Entity	Logic Cells	LC Registers	Memory Bits	M4Ks	Pins	Virtual Pins	LUT-Only LCs	Register-Only
Cyclone: EP1C3T100A8								
filtref	128 (10)	58	0	0	22	0	70 (1)	45 (9)
taps:inst	32 (32)	32	0	0	0	0	0 (0)	24 (24)
state_m:inst1	5 (5)	5	0	0	0	0	0 (0)	0 (0)
hvalues:inst2	0	0	0	0	0	0	0	0
acc:inst3	37 (24)	12	0	0	0	0	25 (12)	12 (12)
accum:inst_1	13 (0)	0	0	0	0	0	13 (0)	0 (0)
mult:inst6	44 (0)	0	0	0	0	0	44 (0)	0 (0)
lpm_mult:lpm_...	44 (0)	0	0	0	0	0	44 (0)	0 (0)

Hierarchy | Files | Design Units

The Project Navigator also allows you to assign design partitions. For more information, see "Creating Design Partitions" on page 57.

For Information About	Refer To
Creating and working with Quartus II projects	"About the Project Navigator" in Quartus II Help
	"Module 2: Create a Design" in the Quartus II Interactive Tutorial
Managing Quartus II projects	*Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*
	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial

Using Revisions

You can use revisions to specify, save, and use different groups of settings and assignments for the design files in a design. Revisions allow you to compare results using different settings and assignments for the same design files in a design.

When you create a revision, the Quartus II software creates a separate Quartus II Settings File, which contains all the settings and assignments related to that revision, and places it in the top-level directory for the design. You can create a revision for any entity in a design. You can view the top-level entity for the any revision in the **Revisions** dialog box on the Project menu or the current top-level design entity in the **General** page of the **Settings** dialog box on the Assignments menu.

The **Revisions** dialog box allows you to view all the revisions for the current project, create a revision for a specific design entity, delete a revision, or set a particular revision as the current revision for compilation, simulation, or timing analysis. The information in the **Revisions** dialog box shows the top-level design entity for a particular revision and the family and device selected for the revision. A check mark icon indicates the current revision. With the **Create Revision** dialog box, you can create a new revision (based on an existing revision), enter a description for the revision, copy the database used to create the revision, and set a revision as the current revision. You can also select which columns appear in the **Revisions** dialog box (Figure 3).

Figure 3. Revisions Dialog Box
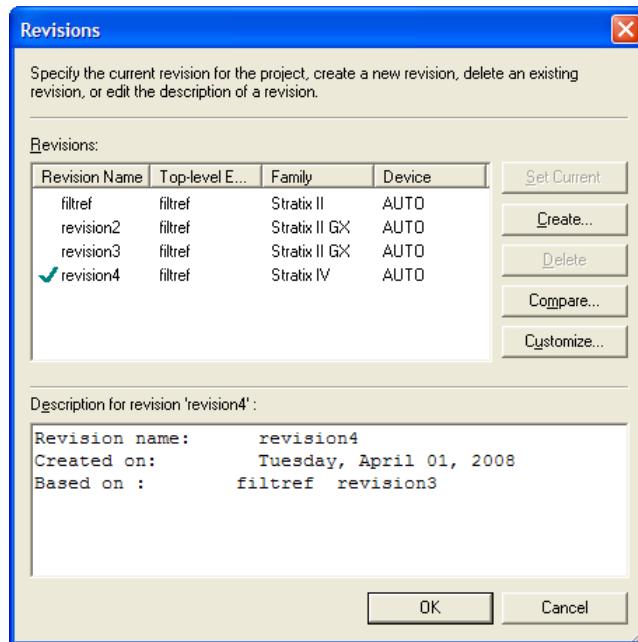
Creating a revision does not affect the source design files for the project. You can create a revision, set it as the current revision for the design, and then make assignments and settings for the entity. This feature allows you to create different settings and assignments for the same design entity and save those settings as different revisions for comparison. Each revision has a corresponding report file that you can open to view and compare the results of the effects of settings and assignments changes against other revisions.

You can use the **Compare Revisions** dialog box, which is available from the **Revisions** dialog box, to compare the results of compilations with different revisions. The **Compare Revisions** dialog box has a **Results** tab and an **Assignments** tab. By default, the comparison shows all revisions for the project, but you can also customize the comparison by selecting which revisions you want to display and adjusting the order. You can export a Comma-Separated Value File (**.csv**) from the comparison. Figure 4 shows the **Results** tab of the **Compare Revisions** dialog box, which allows you to compare the results of each revision.

Figure 4. Results Tab of Compare Revisions Dialog Box

Figure 5 shows the **Assignments** tab of the **Compare Revisions** dialog box, which allows you to compare the assignment settings for each revision.

Figure 5. Assignments Tab of Compare Revisions Dialog Box

Compare Revisions

Results | Assignments

	D:/current/80/tut... Revision filtref	D:/current/80/tut... Revision revision2	D:/current/80/tut... Revision revision3	D:/current/80/tut... Revision revision4
Pin & Location Assignments				
RESERVE_PIN	AS INPUT TRI-S...	AS INPUT TRI-S...	AS INPUT TRI-S...	AS INPUT TRI-S...
Classic Timing Assignments				
USE_TIMEQUEST_TIMING...	LOCKED		LOCKED	
TIMEQUEST_MULTICORN...	OFF		OFF	OFF
TIMEQUEST_DO_CCPP_R...			OFF	OFF
Analysis & Synthesis Assignments				
FAMILY	Stratix II	Stratix II GX	Stratix II GX	Stratix IV
TOP_LEVEL_ENTITY	filtref	LOCKED	filtref	LOCKED
STRATIXII_OPTIMIZATION...	SPEED	SPEED	AREA	
OPTIMIZATION_TECHNIQ...				AREA
Fitter Assignments				
FITTER_EFFORT	STANDARD FIT	STANDARD FIT	STANDARD FIT	STANDARD FIT
OPTIMIZE_TIMING	NORMAL_COMPI...	NORMAL_COMPI...	NORMAL_COMPI...	NORMAL_COMPI...
DEVICE	AUTO	AUTO	AUTO	AUTO
RESERVE_DATA0_AFTER...	USE AS REGULA...	USE AS REGULA...	USE AS REGULA...	USE AS REGULA...
EDA Netlist Writer Assignments				
Assembler Assignments				
APEX20K_CONFIGURATIO...	EPC2	EPC2	EPC2	EPC2
AUTO_RESTART_CONFIG...	OFF	OFF	OFF	OFF
EXCALIBUR_CONFIGURAT...	EPC2	EPC2	EPC2	EPC2
FLEX10K_CONFIGURATIO...	EPC2	EPC2	EPC2	EPC2
FLEX6K_CONFIGURATION...	EPC1	EPC1	EPC1	EPC1
MERCURY_CONFIGURATI...	EPC2	EPC2	EPC2	EPC2
STRATIX_CONFIGURATIO...	EPC2	EPC2	EPC2	EPC2
USE_CONFIGURATION_D...	ON	ON	ON	ON
Simulator Assignments				
VECTOR_INPUT_SOURCE	D:/current/80/tut...	D:/current/80/tut...	D:/current/80/tut...	D:/current/80/tut...
Design Assistant Assignments				
Programmer Assignments				
SignalProbe Assignments				
SignalTap II Assignments				
LogicLock Region Assignments				
Assignment Group Assignments				

Customize... | Close | Export...

For Information About	Refer To
Using revisions	*Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*
	"About Revisions" and "About Project Management" in Quartus II Help

Using Version-Compatible Databases

The Quartus II software allows you to export version-compatible database files for use in a later version of the Quartus II software, eliminating the need for a full compilation of the design in the later version of the Quartus II software. You can export a database at any stage in the design flow after running Analysis & Synthesis or the **quartus_map** command-line executable.

You can use this feature to create and optimize a design and then preserve the database for timing analysis in a later version of the Quartus II software to ensure that the design still meets the timing requirements when measured against the updated timing models in the later version.

To export a database for use in a future version of the Quartus II software, you can use the **Export Database** command on the Project menu to select the directory to export the database. The Quartus II software exports the design database. You can then use the **Import Database** command on the Project menu in a future version of the Quartus II software to select the project folder, import the design database, and perform timing analysis to verify the timing requirements of the design.

You can also use the **quartus_cdb** command-line executable to export or import design databases. Version-compatible databases are available in version 4.1 or later versions of the Quartus II software.

☞ Using the quartus_cdb executable

You can import or export version-compatible databases by using the **quartus_cdb** executable.

To use the **quartus_cdb** executable to import or export a database, type one of the following commands at a command prompt:

```
quartus_cdb <project> -c <revision> --import_database=<project directory> ↵
quartus_cdb <project> -c <revision> --export_database=<project directory> ↵
```

If you want to get help on the **quartus_cdb** executable, type one of the following commands at the command prompt:

```
quartus_cdb -h ↵
quartus_cdb --help ↵
quartus_cdb --help=<topic name> ↵
```

For Information About	Refer To
Using version-compatible databases	*Quartus II Project Management* chapter in volume 2 of the *Quartus II Handbook*
	"About Project Management" in Quartus II Help

Creating a Design

You can create designs in the Quartus II Block Editor or Text Editor. The Quartus II software also supports designs created from EDIF Input Files (**.edf**) or Verilog Quartus Mapping Files (**.vqm**) generated by EDA design entry and synthesis tools. You can also create Verilog HDL or VHDL designs in EDA design entry tools, and either generate EDIF Input Files and VQM Files, or use the Verilog HDL or VHDL design files directly in Quartus II projects. For more information on using EDA synthesis tools to generate EDIF Input Files or VQM Files, see "Using Other EDA Synthesis Tools" on page 68 in Chapter 5, "Synthesis."

The Quartus II software also supports use of Quartus II Exported Partition Files (**.qxp**) as source files containing entities you can add to your design.

You can use the design file types listed in Table 2 to create a design in the Quartus II software or in EDA design entry tools.

Table 2. Supported Design File Types (Part 1 of 2)

Type	Description	Extension
Block Design File	A schematic design file created with the Quartus II Block Editor.	.bdf
EDIF Input File	An EDIF netlist file, generated by any standard EDIF netlist writer.	.edf .edif
State Machine File	A state machine design file created with the State Machine Editor.	.smf
Text Design File	A design file written in the Altera Hardware Description Language (AHDL).	.tdf

Table 2. Supported Design File Types (Part 2 of 2)

Type	Description	Extension
Verilog Design File	A design file that contains design logic defined with Verilog HDL.	.v .vlg .verilog
VHDL Design File	A design file that contains design logic defined with VHDL.	.vh .vhd .vhdl
Verilog Quartus Mapping File	A Verilog HDL–format netlist file generated by the Synplicity Synplify software or the Quartus II software.	.vqm
Quartus II Exported Partition File	A binary file that contains compilation results for an exported design partition and includes a post-fit or post-synthesis netlist, LogicLock regions, and other assignments.	.qxp

Using the Quartus II Block Editor

The Quartus II Block Editor allows you to enter and edit graphic design information in the form of schematics and block diagrams. The Block Editor reads and edits Block Design Files.

Each Block Design File contains blocks and symbols that represent logic in the design. The Block Editor incorporates the design logic represented by each block diagram, schematic, or symbol into the project.

You can create new design files from blocks in a Block Design File, update the design files when you modify the blocks and the symbols, and generate Block Symbol Files (**.bsf**), AHDL Include Files (**.inc**), and HDL files from Block Design Files. You can also analyze the Block Design Files for errors before compilation. The Block Editor also provides a set of tools that help you connect blocks and primitives in a Block Design File, including bus and node connections and signal name mapping.

You can customize the Block Editor display to show options, such as guidelines and grid spacing, rubberbanding, colors and screen elements, zoom, and different block and primitive properties to suit your preferences.

You can use the following features of the Block Editor to create a Block Design File in the Quartus II software:

■ **Instantiate Altera-provided megafunctions:** The **MegaWizard Plug-In Manager** on the Tools menu allows you to create or modify design files that contain custom variations of megafunctions. These custom megafunction variations are based on Altera-provided megafunctions, including LPM functions. Megafunctions are represented by blocks in Block Design Files. See "Using the MegaWizard Plug-In Manager" on page 31.

■ **Insert block and primitive symbols:** Block diagrams use rectangular-shaped symbols, called blocks, to represent design entities and the corresponding assigned signals, and are useful in top-down design. Blocks are connected by conduits that represent the flow of the corresponding signals. You can use block diagrams exclusively to represent your design, or you can combine them with schematic elements. The Quartus II software provides symbols for a variety of logic functions—including primitives, library of parameterized modules (LPM) functions, and other megafunctions—that you can use in the Block Editor.

■ **Create files from blocks or Block Design Files**: To facilitate hierarchical projects, you can use the **Create/Update** command on the File menu in the Block Editor to create other Block Design Files, AHDL Include Files, Verilog HDL and VHDL design files, and Quartus II Block Symbol Files from blocks within a Block Design File. You can also create Verilog Design Files, VHDL Design Files, and Block Symbol Files from a Block Design File itself.

Using the Quartus II Text Editor

The Text Editor is a flexible tool for entering text-based designs in the AHDL, VHDL, and Verilog HDL languages, and the Tcl scripting language. You can also use the Text Editor to enter, edit, and view other ASCII text files, including those created for or by the Quartus II software.

The Text Editor also allows you to insert a template for any AHDL statement or section, Tcl command, or supported VHDL or Verilog HDL construct into the current file. AHDL, VHDL, and Verilog HDL templates provide an easy way for you to enter HDL syntax, increasing the speed and accuracy of design entry. You can also get context-sensitive help on all AHDL elements, keywords, statements, megafunctions, and primitives.

Using the Quartus II Symbol Editor

The Symbol Editor allows you to view and edit predefined symbols that represent macrofunctions, megafunctions, primitives, or design files. Each Symbol Editor file represents one symbol. For each symbol file, you can choose from libraries containing Altera megafunctions and LPM functions. You can customize these Block Symbol Files and then add the symbols to schematics created with the Block Editor.

Using Verilog HDL, VHDL, & AHDL

You can use the Quartus II Text Editor or another text editor to create Text Design Files, Verilog Design Files, and VHDL Design Files, and combine them with other types of design files in a hierarchical design.

Verilog Design Files and VHDL Design Files can contain any combination of Quartus II–supported constructs. They can also contain Altera-provided logic functions, including primitives and megafunctions, and user-defined logic functions.

In the Text Editor, you use the **Create/Update** command on the File menu to create a Block Symbol File from the current Verilog HDL or VHDL design file and then incorporate it into a Block Design File. Similarly, you can create an AHDL Include File that represents a Verilog HDL or VHDL design file and incorporate it into an Text Design File or another Verilog HDL or VHDL design file.

For VHDL designs, you can specify the name of a VHDL library for a design in the **Properties** dialog box, which is available from the **Files** page of the **Settings** dialog box on the Assignments menu.

For more information on using the Verilog HDL and VHDL languages in the Quartus II software, see "Using Quartus II Verilog HDL & VHDL Integrated Synthesis" on page 65 in Chapter 5, "Synthesis."

AHDL is a high-level, modular language that is completely integrated into the Quartus II system. AHDL supports Boolean equation, state machine, conditional, and decode logic. AHDL also allows you to create and use parameterized functions, and includes full support for LPM functions. AHDL is especially well suited for designing complex combinational logic, group operations, state machines, truth tables, and parameterized logic.

For Information About	Refer To
Using the Quartus II Block Editor and Symbol Editor	"About Design Entry" in Quartus II Help
Using the Quartus II Text Editor	"About the Quartus II Text Editor" in Quartus II Help
Creating designs in the Quartus II software	"Module 2: Create a Design" in the Quartus II Interactive Tutorial

Using the State Machine Editor

The State Machine Editor allows you to create graphic representations of state machines for use in your design. When you have fully described your state machine, you can generate a corresponding Verilog Design File or VHDL Design File.

The State Machine Editor provides a state machine diagram view where you can view the state diagram you created with the **State Machine** wizard or the drawing tools provided, and a ports list that lists all of the input and output ports of the state machine.

Using Altera Megafunctions

Altera megafunctions are complex or high-level building blocks that can be used together with gate and flipflop primitives in Quartus II design files. The parameterizable megafunctions and LPM functions provided by Altera are optimized for Altera device architectures. You must use megafunctions to access some Altera device-specific features, such as memory, DSP blocks, LVDS drivers, PLLs, and SERDES and DDIO circuitry.

You can use the **MegaWizard Plug-In Manager** on the Tools menu to create Altera megafunctions, LPM functions, and IP functions for use in designs in the Quartus II software and EDA design entry and synthesis tools. Table 3 shows the types of Altera-provided megafunctions and LPM functions that you can create with the **MegaWizard Plug-In Manager**.

Table 3. Altera-Provided Megafunctions & LPM Functions

Type	Description
Arithmetic Components	Includes accumulators, adders, multipliers, and LPM arithmetic functions.
Gates	Includes multiplexers and LPM gate functions.
I/O Components	Includes Clock Data Recovery (CDR), phase-locked loop (PLL), double data rate (DDR), gigabit transceiver block (GXB), LVDS receiver and transmitter, PLL reconfiguration, and remote update megafunctions.
Memory Compiler	Includes the FIFO Partitioner, RAM, and ROM megafunctions.
Storage Components	Memory and shift register megafunctions, and LPM memory functions.

To save valuable design time, Altera recommends using megafunctions instead of coding your own logic, where possible. These functions can offer more efficient logic synthesis and device implementation. It is easy to scale megafunctions to different sizes by simply setting parameters. Altera also provides AHDL Include Files and VHDL Component Declarations for both Altera-provided megafunctions and LPM functions.

For Information About	Refer To
Using the MegaWizard Plug-In Manager	"About the MegaWizard Plug-In Manager" in Quartus II Help
	"Module 2: Create a Design" in the Quartus II Interactive Tutorial

Using Intellectual Property (IP) Megafunctions

Altera provides several methods for obtaining both Altera Megafunction Partners Program (AMPP™) and MegaCore® megafunctions, functions that are rigorously tested and optimized for the highest performance in Altera device-specific architectures. You can use these parameterized blocks of intellectual property to reduce design and test time. MegaCore and AMPP

megafunctions include megafunctions for embedded processors, interfaces and peripherals, digital signal processing (DSP), and communications applications.

Altera provides the following programs, features, and functions to assist you in using IP functions in the Quartus II software and EDA design entry tools:

■   **AMPP Program:** The AMPP program offers support to third-party vendors to create and distribute megafunctions for use with the Quartus II software. AMPP partners offer a large selection of off-the-shelf megafunctions that are optimized for Altera devices.

   Evaluation periods for AMPP functions are determined by the individual vendors. You can download and evaluate AMPP functions through the IP MegaStore™ on the Altera website at **www.altera.com/ipmegastore**.

■   **MegaCore Functions:** MegaCore functions are predesigned and optimized design files for complex system-level functions, and are fully parameterizable using the **MegaWizard Plug-In Manager** and IP Toolbench. IP Toolbench is a toolbar that you can use to quickly and easily view documentation, specify parameters, set up other EDA tools, and generate all the files necessary for integrating a parameterized MegaCore function into your design.

   MegaCore functions are automatically installed when you install the Quartus II software. You can also download individual IP MegaCore functions from the Altera website, via the IP MegaStore, and install them separately. You can also access MegaCore functions though the MegaWizard Portal Extension to the **MegaWizard Plug-In Manager**.

■   **OpenCore Evaluation Feature:** The OpenCore® evaluation feature allows you to evaluate AMPP functions before purchase. You can use the OpenCore feature to compile, simulate, and verify the performance of a design, but it does not support programming file generation.

■   **OpenCore Plus Hardware Evaluation Feature:** The free OpenCore Plus hardware evaluation feature allows you to simulate the behavior of a MegaCore function within your system, verify the functionality of the design, and evaluate its size and speed quickly and easily. In addition, the Quartus II software generates time-limited programming files for designs containing MegaCore functions, allowing you to program devices and verify your design in hardware before purchasing a license for the IP megafunction.

When the OpenCore Plus hardware feature is turned on in the **More Compilation Process Settings** dialog box, the Quartus II software inserts a small amount of control logic in your design. This logic can have an adverse effect on fitting, especially with small devices. You can turn off the OpenCore Plus hardware evaluation feature to direct the Quartus II software to omit the additional logic.

Using the MegaWizard Plug-In Manager

The **MegaWizard Plug-In Manager** helps you create or modify design files that contain custom megafunction variations, which you can then instantiate in a design file. These custom megafunction variations are based on Altera-provided megafunctions, including LPM, MegaCore, and AMPP functions. The **MegaWizard Plug-In Manager** runs a wizard that helps you easily specify options for the custom megafunction variations. The wizard allows you to set values for parameters and optional ports. You can open the **MegaWizard Plug-In Manager** on the Tools menu or from within a Block Design File, or you can run it as a stand-alone utility. Table 4 lists the files generated by the **MegaWizard Plug-In Manager** for each custom megafunction variation you generate.

Table 4. Files Generated by the MegaWizard Plug-In Manager (Part 1 of 2)

File Name	Description
<output file>.bsf	Schematic design file for the megafunction used in the Block Editor
<output file>.cmp	Component Declaration File
<output file>.inc	AHDL Include File for the module in the megafunction wrapper file
<output file>.tdf	Megafunction wrapper file for instantiation in an AHDL design
<output file>.vhd	Megafunction wrapper file for instantiation in a VHDL design
<output file>.v	Megafunction wrapper file for instantiation in a Verilog HDL design
*<output file>*_bb.v	Hollow-body or black box declaration of the module in the megafunction wrapper file used in Verilog HDL designs to specify port directions when using EDA synthesis tools

Table 4. Files Generated by the MegaWizard Plug-In Manager (Part 2 of 2)

File Name	Description
<output file>_inst.tdf	Sample AHDL instantiation of the subdesign in the megafunction wrapper file
<output file>_inst.vhd	Sample VHDL instantiation of the entity in the megafunction wrapper file
<output file>_inst.v	Sample Verilog HDL instantiation of the module in the megafunction wrapper file

---

☞ **Using the Stand-Alone MegaWizard Plug-In Manager**

You can use the **MegaWizard Plug-In Manager** from outside the Quartus II software by typing the following command at a command prompt:

```
qmegawiz ↵
```

Instantiating Megafunctions in the Quartus II Software

You can instantiate Altera megafunctions and LPM functions in the Quartus II software through direct instantiation in the Block Editor, through instantiation in HDL code (either by instantiating through the port and parameter definition or by using the **MegaWizard Plug-In Manager** to parameterize the megafunction and create a wrapper file), or through inference.

Altera recommends that you use the **MegaWizard Plug-In Manager** to instantiate megafunctions and create custom megafunction variations. The wizard provides a GUI for customizing and parameterizing megafunctions, and ensures that you set all megafunction parameters correctly.

Instantiation in Verilog HDL & VHDL

You can use the **MegaWizard Plug-In Manager** to create a megafunction or a custom megafunction variation. The **MegaWizard Plug-In Manager** then creates a Verilog HDL or VHDL wrapper file that contains an instance of the

megafunction, which you can then use in your design. For VHDL megafunctions, the **MegaWizard Plug-In Manager** also creates a Component Declaration File.

Using the Port & Parameter Definition

You can instantiate the megafunction directly in your Verilog HDL or VHDL design by calling the function like any other module or component. In VHDL, you also must use a component declaration.

Inferring Megafunctions

Quartus II Analysis & Synthesis automatically recognizes certain types of HDL code and infers the appropriate megafunction. The Quartus II software uses inference because Altera megafunctions are optimized for Altera devices, and performance may be better than standard HDL code. For some architecture-specific features, such as RAM and DSP blocks, you must use Altera megafunctions.

The Quartus II software maps the following logic to megafunctions during synthesis:

- Counters
- Adders/Subtractors
- Multipliers
- Multiply-accumulators and multiply-adders
- RAM
- Shift registers

Instantiating Megafunctions in EDA Tools

You can use Altera-provided megafunctions, LPM functions, and IP functions in EDA design entry and synthesis tools. You can instantiate megafunctions in EDA tools by creating a black box for the function, by inference, or by using the clear box methodology.

Using the Black Box Methodology

You can use the **MegaWizard Plug-In Manager** to generate Verilog HDL or VHDL wrapper files for megafunctions. For Verilog HDL designs, the **MegaWizard Plug-In Manager** also generates a Verilog Design File that contains a hollow-body declaration of the module, used to specify port directions.

The Verilog HDL or VHDL wrapper file contains the ports and parameters for the megafunction, which you can use to instantiate the megafunction in the top-level design file as well as a sample instantiation file and then direct the EDA tool to treat the megafunction as a black box during synthesis.

The following steps describe the basic flow for using the **MegaWizard Plug-In Manager** to create a black box for an Altera megafunction or LPM function in EDA design entry and synthesis tools:

1. Create and parameterize the megafunction or LPM function using the **MegaWizard Plug-In Manager**.

2. Instantiate the function in the EDA synthesis tool with the black box file or component declaration (along with the sample instantiation file) generated by the **MegaWizard Plug-In Manager**.

3. Perform synthesis and optimization of the design in the EDA synthesis tool. The EDA synthesis tool treats the megafunction as a black box during synthesis.

Instantiation by Inference

EDA synthesis tools automatically recognize certain types of HDL code and infer the appropriate megafunction.You can directly instantiate memory blocks (RAM and ROM), DSP blocks, shift registers, and some arithmetic components in Verilog HDL or VHDL code. The EDA tool then maps the logic to the appropriate Altera megafunction during synthesis.

Using the Clear Box Methodology

In the black box flow, an EDA synthesis tool treats Altera megafunctions and LPM functions as black boxes. As a result, the EDA synthesis tool cannot fully synthesize and optimize designs with Altera megafunctions, because the tool does not have a full model or timing information for the function. Using the clear box flow, you can use the **MegaWizard Plug-In Manager** to create a fully synthesizeable Altera megafunction or LPM function for use with EDA synthesis tools.

The following steps describe the basic flow for using clear box megafunctions with EDA synthesis tools:

1. Create and parameterize the megafunction or LPM functions using the **MegaWizard Plug-In Manager**. Make sure you turn on **Generate clear box netlist file instead of a default wrapper file (for use with supported EDA synthesis tools only)** in the **MegaWizard Plug-In Manager**.

2. Instantiate the function in the EDA synthesis tool using the Verilog or VHDL design file generated by the **MegaWizard Plug-In Manager**.

3. Perform synthesis and optimization of the design in the EDA synthesis tool.

Use of the clear box methodology generally results in slower simulation times in EDA simulation tools (but not the Quartus II Simulator), due to the level of detail (timing information and device resources used) that is included with a clear box megafunction or LPM function. In addition, specific device details are included in the clear box megafunction or LPM function, so that to use a different device for the design, the clear box function needs to be regenerated for the new device.

For Information About	Refer To
Using Altera-provided megafunctions and LPM functions in EDA tools	"Creating and Instantiating Altera-Provided Functions in Other EDA Tools" in Quartus II Help
	Synopsys Synplify Support chapter in volume 1 of the *Quartus II Handbook*
	Mentor Graphics LeonardoSpectrum Support chapter in volume 1 of the *Quartus II Handbook*
	Mentor Graphics Precision Synthesis Support chapter in volume 1 of the *Quartus II Handbook*
Using Altera-provided megafunctions and LPM functions in the Quartus II software	"Module 2: Create a Design" in the Quartus II Interactive Tutorial
Using the **MegaWizard Plug-In Manager** and Altera-provided megafunctions and LPM functions	"About the MegaWizard Plug-In Manager" in Quartus II Help
MegaCore functions and OpenCore Plus hardware evaluation feature	*AN 343: OpenCore Evaluation of AMPP Megafunctions* on the Altera website
	AN 320: OpenCore Plus Evaluation of Megafunctions on the Altera website
	Simulating Altera IP in Third-Party Simulation Tools chapter in volume 3 of the *Quartus II Handbook*

Chapter Three

Command-Line And Tcl Design Flows
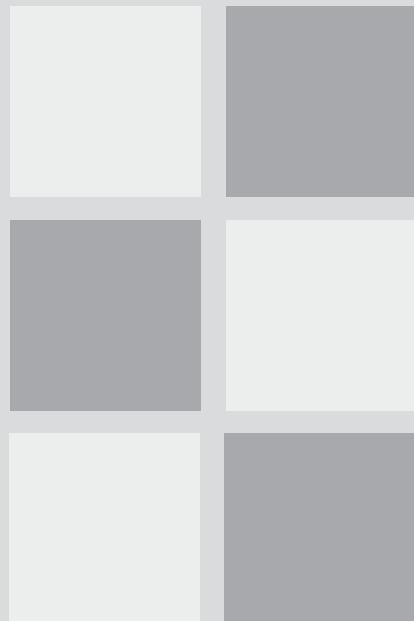
3

Introduction

The Quartus II software provides a complete command-line interface and Tcl scripting API. You can use command-line executables and scripts to perform every stage of the design flow. Using the command-line flow allows you to reduce memory requirements, control the Quartus II software with scripts or Tcl commands, and create makefiles (Figure 1).

Figure 1. Command-Line Design Flow
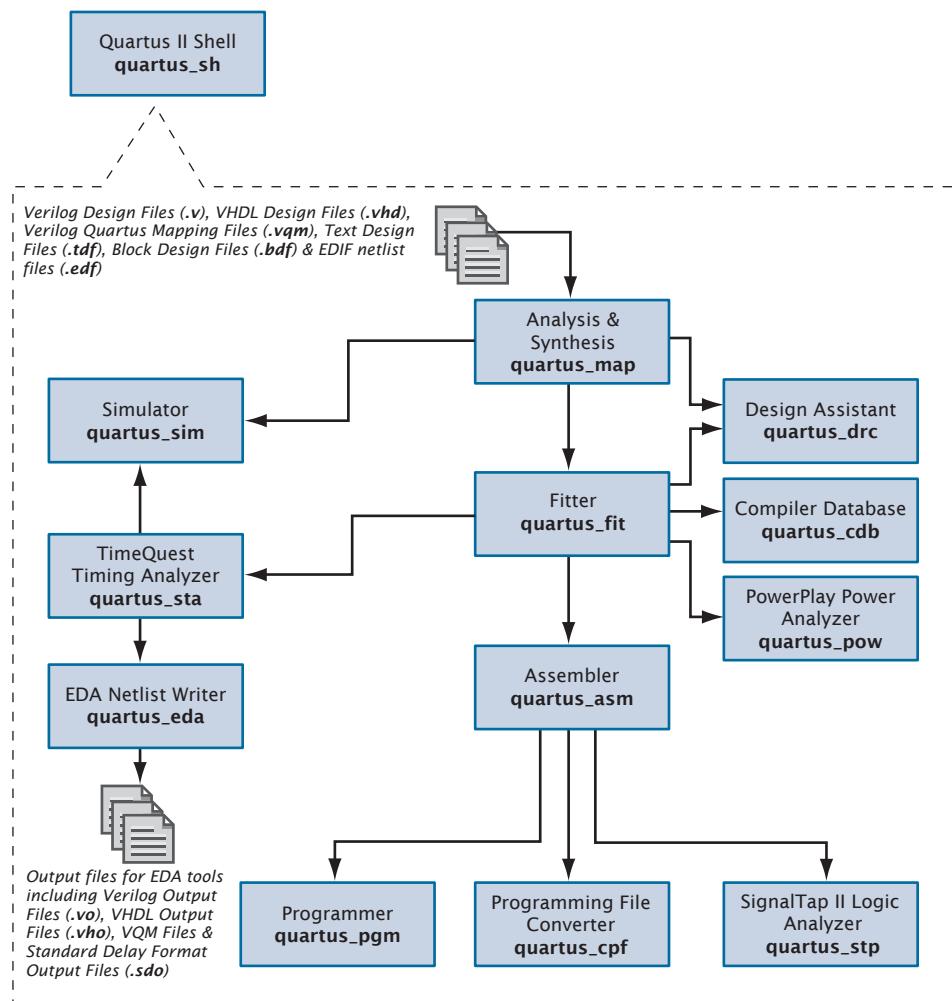
Command-Line Executables

The Quartus II software includes separate executables for each stage of the design flow. Each executable occupies memory only while it is being run. You can use these executables with standard command-line commands and scripts, with Tcl scripts, and in makefile scripts. See Table 1 for a list of all available command-line executables.

☞ Stand-Alone Graphical User Interface Executables

The Quartus II software also provides some stand-alone GUI executables available from the command prompt. The **qmegawiz** executable provides a command-line interface for the **MegaWizard Plug-In Manager**, as well as a way to start the GUI as a stand-alone application.

In addition, the **quartus_pgmw** executable provides the GUI for the Programmer as a stand-alone application. The **quartus_stpw** executable provides the GUI for the SignalTap II Logic Analyzer as a stand-alone application. Similarly, the **quartus_staw** executable provides the GUI for the TimeQuest timing analyzer as a stand-alone application.

Table 1. Command-Line Executables (Part 1 of 3)

Executable Name	Title	Function
quartus_map	Analysis & Synthesis	Creates a project if one does not already exist, and then creates the project database, synthesizes your design, and performs technology mapping on design files of the project.
quartus_fit	Fitter	Places and routes a design. Analysis & Synthesis must be run successfully before running the Fitter.
quartus_drc	Design Assistant	Checks the reliability of a design based on a set of design rules. Design Assistant is especially useful for checking the reliability of a design before migrating the design to HardCopy devices. Either Analysis & Synthesis or the Fitter must be run successfully before running the Design Assistant.
quartus_sta	TimeQuest Timing Analyzer	Performs ASIC-style timing analysis of the circuit using constraints entered in Synopsys Design Constraint format.

Table 1. Command-Line Executables (Part 2 of 3)

Executable Name	Title	Function
quartus_asm	Assembler	Creates one or more programming files for programming or configuring the target device. The Fitter must be run successfully before running the Assembler.
quartus_eda	EDA Netlist Writer	Generates netlist files and other output files for use with other EDA tools. Analysis & Synthesis, the Fitter, or the Timing Analyzer must be run successfully before running the EDA Netlist Writer, depending on the options used.
quartus_cdb	Compiler Database Interface (including VQM Writer)	Imports and exports version-compatible databases and merges partitions. Generates internal netlist files, including Verilog Quartus Mapping Files, for the Quartus II Compiler database so they can be used for back-annotation and for the LogicLock feature, and back-annotates device and resource assignments to preserve the fit for future compilations. Either the Fitter or Analysis & Synthesis must be run successfully before running the Compiler Database Interface.
quartus_jli	Jam STAPL Player	Reads and executes Jam Files (**.jam**) in the STAPL format. A single Jam File can perform several functions, such as programming, configuring, verifying, erasing, and blank-checking a programmable device in a JTAG chain.
quartus_jbcc	Jam Compiler	The Quartus II JAM Compiler converts Jam/STAPL files to Jam Byte Code Files (**.jbc**) which store data for programming, configuring, verifying, and blank-checking one or more devices in a JTAG chain.
quartus_sim	Simulator	Performs functional or timing simulation on your design. Analysis & Synthesis must be run before performing a functional simulation. Timing Analysis must be run before performing a timing simulation.
quartus_pow	Power Analyzer	Analyzes and estimates total dynamic and static power consumed by a design. Computes toggle rates and static probabilities for output signals. The Fitter must be run successfully before running the PowerPlay Power Analyzer.

Table 1. Command-Line Executables (Part 3 of 3)

Executable Name	Title	Function
quartus_pgm	Programmer	Programs Altera devices.
quartus_cpf	Programming File Converter	Converts programming files to secondary programming file formats.
quartus_stp	SignalTap II Logic Analyzer	Sets up your SignalTap II File (**.stp**). When it is run after the Assembler, the SignalTap II Logic Analyzer captures signals from internal device nodes while the device is running at speed.
quartus_si	SSN Analyzer	Estimates and reports the simultaneous switching noise contributions to voltage and timing noise for device pins.
quartus_sh	Tcl Shell	Provides overall control of Quartus II projects and compilation flows, as well as a Tcl shell.

☞ **Getting Help On the Quartus II Executables**

If you want to get help on the command-line options that are available for each of the Quartus II executables, type one of the following commands at the command prompt:

<executable name> `-h` ↵
<executable name> `--help` ↵
<executable name> `--help=`*<topic or option name>* ↵

You can also get help on command-line executables by using the Quartus II Command-Line Executable and Tcl API Help Browser, which is a Tcl- and Tk-based GUI that lets you browse the command-line and Tcl API help. To use this help, type the following command at the command prompt:

`quartus_sh --qhelp` ↵

You can perform a full compilation by using the following command:

`quartus_sh --flow compile` *<project name>* `[-c` *<revision name>*`]` ↵

This command runs the **quartus_map**, **quartus_fit**, **quartus_asm**, and **quartus_tan** executables. Depending on your settings, this command may also run the optional **quartus_drc**, **quartus_eda**, **quartus_cdb**, and **quartus_sta** executables.

Some of the executables create a separate text-based report file, named after the current project revision, that you can view with any text editor. The name of each report file uses the following format:

<revision name>.<abbreviated executable name>.**rpt**

For example, if you want to run the **quartus_map** executable for a project, you could type the following command at the command prompt:

```
quartus_map <project name> ⏎
```

The **quartus_map** executable analyzes and synthesizes the design and produces a report file with the name *<revision name>*.**map.rpt**.

☞ **Using Quartus II Settings Files with Quartus II Executables**

When you are using the Quartus II executables, by default the Quartus II software uses the revision that has the same name as the project name. If you want to use a revision with a name that is different from the project name, you can use the `-c` option to specify the name of the revision and its associated Quartus II Settings File (**.qsf**). For example, if you want to run the **quartus_map** executable for the **chiptrip** project with a revision named **speed_ch** and its associated **speed_ch.qsf** file, you could type the following command at the command prompt:

```
quartus_map chiptrip -c speed_ch ⏎
```

The **quartus_map** executable performs Analysis & Synthesis using that revision and its settings, and produces a report file with the name **speed_ch.map.rpt**.

The Quartus II software also offers several predefined compilation flows that use the Quartus II executables. You can use these commands with the `quartus_sh --flow` command, or with the Tcl `execute_flow` command. Table 2 shows some common Compiler flows.

Table 2. Command-Line Compiler Flows

Flow	Description	Command-Line Option for quartus_sh --flow or execute_flow
Full compilation flow	Performs a full compilation of the current design.	`compile`
Compilation and simulation flow	If the simulation mode is timing, performs a full compilation and then a simulation of the current design. If the simulation mode is functional, generates a functional simulation netlist and then performs a simulation of the current design.	`compile_and_simulate`
SignalProbe flow	Routes user-specified signals to output pins without affecting the existing fitting in a design, so that you can debug signals without completing a full compilation.	`signalprobe`
Early Timing Estimate flow	Performs a partial compilation, but stops and generates early timing estimates before the Fitter is complete.	`early_timing_estimate`

For Information About	Refer To
Using compilation flows	"About Compilation Flows" in Quartus II Help

Using Standard Command-Line Commands & Scripts

You can use the Quartus II executables with any command-line scripting method, such as Perl scripts, Tcl scripts, and batch files. You can design these scripts to create new projects or to compile existing projects. You can also run the executables from the command prompt or console.

Figure 2 shows an example of a standard batch file. The example demonstrates how to create a project, perform Analysis & Synthesis, perform place and route, perform timing analysis, and generate programming files for the **filtref** design that is included with the Quartus II software. If you have installed the **filtref** design, it is in the **/altera/**<version number>**/qdesigns/fir_filter** directory. You can run the four commands in Figure 2 from a command prompt in the new project directory, or you can store them in a batch file or shell script. These examples assume that the **/**<Quartus II system directory>**/bin** directory (or the **/**<Quartus II system directory>**/linux** directory on Linux workstations) is included in your PATH environment variable.

Figure 2. Example of a Command-Line Script

```
quartus_map filtref --family=Stratix
```
Creates a new Quartus II project targeting the Stratix device family

```
quartus_fit filtref --part=EP1S10F780C5 --fmax=80MHz --tsu=8ns
```
Performs fitting for the EP1S10F780C5 device and specifies global timing requirements

```
quartus_sta filtref
```
Performs timing analysis

```
quartus_asm filtref
```
Generates programming files

Figure 3 shows an excerpt from a command-line script for use on a Linux workstation. The script assumes that the Quartus II tutorial project called **fir_filter** exists in the current directory. The script analyzes every design file in the **fir_filter** project and reports any files that contain syntax errors.

Figure 3. Example of a Linux Command-Line Shell Script

```
#!/bin/sh
FILES_WITH_ERRORS=""
for filename in `ls *.bdf *.v`
do
        quartus_map fir_filter --analyze_file=$filename
        if [ $? -ne 0 ]
        then
                FILES_WITH_ERRORS="$FILES_WITH_ERRORS $filename"
        fi
done
if [ -z "$FILES_WITH_ERRORS" ]
then
        echo "All files passed the syntax check"
        exit 0
else
        echo "There were syntax errors in the following file(s)"
        echo $FILES_WITH_ERRORS
        exit 1
fi
```

For Information About	Refer To
Command-Line Scripting	"About Quartus II Scripting" in Quartus II Help
	Command-Line Scripting chapter in volume 2 of the *Quartus II Handbook*
	Quartus II Scripting Reference Manual

Using Tcl Commands

You can use Tcl commands and scripts with the Quartus II executables to perform the following functions in a Quartus II project:

- Project and assignment functions
- Device functions
- Advanced device functions
- Flow functions
- Timing functions
- Advanced timing functions
- Simulator functions
- Report functions
- Timing report functions
- Back-annotate functions
- LogicLock functions
- Chip Planner functions
- Miscellaneous functions

There are several ways to use Tcl scripts in the Quartus II software. You can create a Tcl script by using commands from the Quartus II API for Tcl. You should save a Tcl script as a Tcl Script File (**.tcl**).

The **Templates** command on the Edit menu in the Quartus II Text Editor allows you to insert Tcl templates and Quartus II Tcl templates (for Quartus II commands) into a text file to create Tcl scripts. Commands used in the Quartus II Tcl templates use the same syntax as the Tcl API commands.

If you want to use an existing project as a baseline for another project, you can click **Generate Tcl File for Project** on the Project menu to generate a Tcl Script File for the project. After editing this generated script to target your new project, run the script to apply all assignments from the previous project to the new project.

You can run Tcl scripts from the system command prompt with the **quartus_sh** executable, from the Quartus II Tcl Console window, or from the **Tcl Scripts** dialog box by clicking **Tcl Scripts** on the Tools menu.

☞ **Getting Help On Tcl Commands**

The Quartus II software includes a Quartus II command-line and Tcl API Help browser, which is a Tcl- and Tk-based GUI that lets you browse the command-line and Tcl API help. To use this help, type the following command at the command prompt:

```
quartus_sh --qhelp ↵
```

You can also view Tcl API Help in Quartus II Help that is available in the GUI. Refer to "About Quartus II Scripting" in Quartus II Help for more information.

Figure 4 shows an example of a Tcl script.

Figure 4. Example of a Tcl Script (Part 1 of 2)

```
## This script works with the quartus_sh executable
# Set the project name to filtref
set project_name filtref

# Open the Project. If it does not already exist, create it
if [catch {project_open $project_name}] {project_new \ $project_name}

# Set Family
set_global_assignment -name family APEX 20KE

# Set Device
set_global_assignment -name device ep20k100eqc208-1

# Optimize for speed
set_global_assignment -name optimization_technique speed

# Turn-on Fastfit fitter option to reduce compile times
set_global_assignment -name fast_fit_compilation on
```

Figure 4. Example of a Tcl Script (Part 2 of 2)

```
# Generate a NC-Sim Verilog simulation Netlist
set_global_assignment -name eda_simulation_tool "NcSim\
(Verilog HDL output from Quartus II)"

# Using the ::quartus::flow package, the execute_flow command
# exports assignments automatically

load_package flow
execute_flow -compile

# Close Project
project_close
```

For Information About	Refer To
Tcl Scripting	The *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*
	"About Quartus II Scripting" in Quartus II Help
	Quartus II Scripting Reference Manual

Creating Makefile Scripts

The Quartus II software supports makefile scripts that use the Quartus II executables, which allow you to integrate your scripts with a wide variety of scripting languages. Figure 5 shows an excerpt from a standard makefile script.

Figure 5. Excerpt from a Makefile Script (Part 1 of 3)

```
###################################################################
# Project Configuration:
#
# Specify the name of the design (project) and Quartus II Settings
# File (.qsf) and the list of source files used.
###################################################################
```

Figure 5. Excerpt from a Makefile Script (Part 2 of 3)

```
PROJECT = chiptrip
SOURCE_FILES = auto_max.v chiptrip.v speed_ch.v tick_cnt.v time_cnt.v
ASSIGNMENT_FILES = chiptrip.qpf chiptrip.qsf


####################################################################
# Main Targets
#
# all: build everything
# clean: remove output files and database
####################################################################


all: smart.log $(PROJECT).asm.rpt $(PROJECT).tan.rpt

clean:
      rm -rf *.rpt *.chg smart.log *.htm *.eqn *.pin *.sof *.pof db

map: smart.log $(PROJECT).map.rpt
fit: smart.log $(PROJECT).fit.rpt
asm: smart.log $(PROJECT).asm.rpt
sta: smart.log $(PROJECT).sta.rpt
smart: smart.log


####################################################################
# Executable Configuration
####################################################################


MAP_ARGS = --family=Stratix
FIT_ARGS = --part=EP1S20F484C6
ASM_ARGS =
STA_ARGS =


####################################################################
# Target implementations
####################################################################


STAMP = echo done >

$(PROJECT).map.rpt: map.chg $(SOURCE_FILES)
      quartus_map $(MAP_ARGS) $(PROJECT)
      $(STAMP) fit.chg

$(PROJECT).fit.rpt: fit.chg $(PROJECT).map.rpt
      quartus_fit $(FIT_ARGS) $(PROJECT)
      $(STAMP) asm.chg
      $(STAMP) tan.chg
```

Figure 5. Excerpt from a Makefile Script (Part 3 of 3)

```
$(PROJECT).asm.rpt: asm.chg $(PROJECT).fit.rpt
      quartus_asm $(ASM_ARGS) $(PROJECT)


$(PROJECT).sta.rpt: sta.chg $(PROJECT).fit.rpt
      quartus_sta $(TAN_ARGS) $(PROJECT)


smart.log: $(ASSIGNMENT_FILES)
      quartus_sh --determine_smart_action $(PROJECT) > smart.log


###################################################################
# Project initialization
###################################################################


$(ASSIGNMENT_FILES):
      quartus_sh --prepare $(PROJECT)


map.chg:
      $(STAMP) map.chg
fit.chg:
      $(STAMP) fit.chg
sta.chg:
      $(STAMP) sta.chg
asm.chg:
      $(STAMP) asm.chg
```

Chapter Four

Constraint Entry
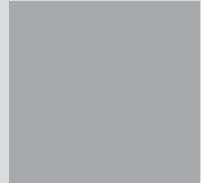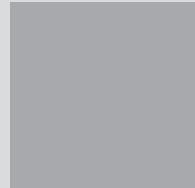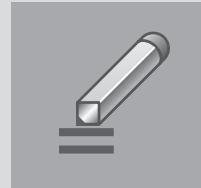
4

Introduction

Once you have created a project and your design, you can use the Assignment Editor, **Settings** dialog box, TimeQuest Timing Analyzer, Pin Planner, Design Partitions window, Design Partition Planner, and the Chip Planner to specify initial design constraints, such as pin assignments, device options, logic options, and timing constraints. You can import assignments by clicking **Import Assignments** on the Assignments menu and export assignments by clicking **Export** on the File menu. You can also import assignments from other EDA synthesis tools using Tcl commands or scripts. Figure 1 shows the constraint and assignment entry flow.

Figure 1. Constraint & Assignment Entry Flow
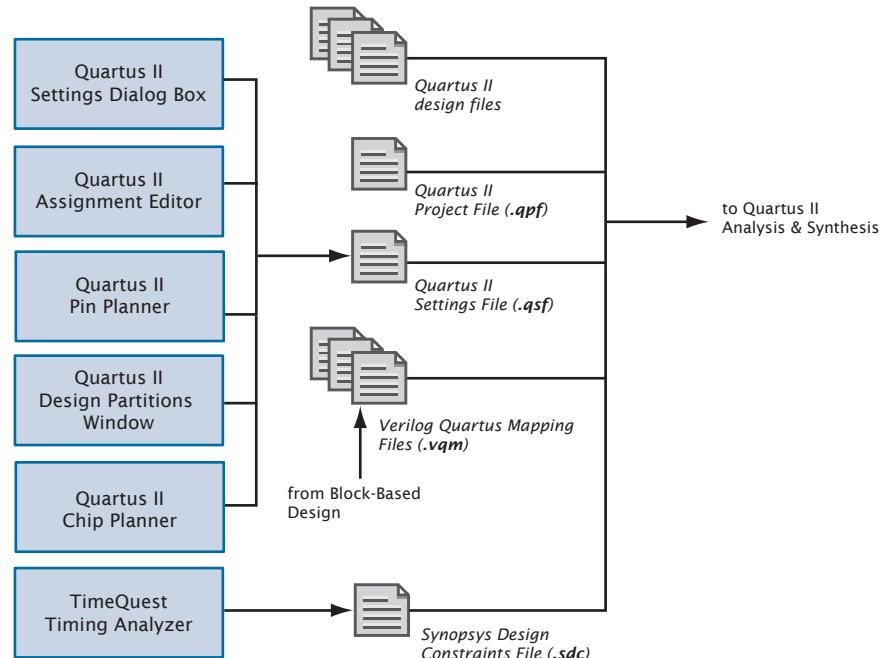
Using the Assignment Editor

The Assignment Editor is the interface for creating and editing node and entity-level assignments in the Quartus II software. Assignments allow you to specify various options and settings for the logic in your design. You can enable or disable individual assignments, and you can also add comments to an assignment.

The following steps illustrate the basic flow for using the Assignment Editor to make assignments:

1. Open the Assignment Editor.

2. Select the appropriate category assignment in the **Category** bar.

3. Specify the appropriate node or entity in the **Node Filter** bar, or use the **Node Finder** dialog box to find a specific node or entity.

4. In the spreadsheet that displays the assignments for the current design, add the appropriate assignment information.

The spreadsheet in the Assignment Editor provides applicable drop-down lists or allows you to type assignment information. As you add, edit, and remove assignments, the corresponding Tcl command appears in the Messages window.

To export the data from the Assignment Editor to a Tcl Script File (**.tcl**) or a Comma-Separated Value File (**.csv**), click **Export** on the File menu. To import assignments data from a Comma-Separated Value File or text file, click **Import Assignments** on the Assignments menu. For more information about importing assignments, see "Importing Assignments" on page 60.

When creating and editing assignments, the Quartus II software dynamically validates the assignment information where possible. If an assignment or assignment value is illegal, the Quartus II software does not add or update the value, and instead reverts to the current value or does not accept the value. When you view all assignments, the Assignment Editor shows all assignments created for the current project that are valid for the current device, but when you view individual assignment categories, the Assignment Editor displays only the assignments that are related to the specific category selected.

Figure 2. The Quartus II Assignment Editor
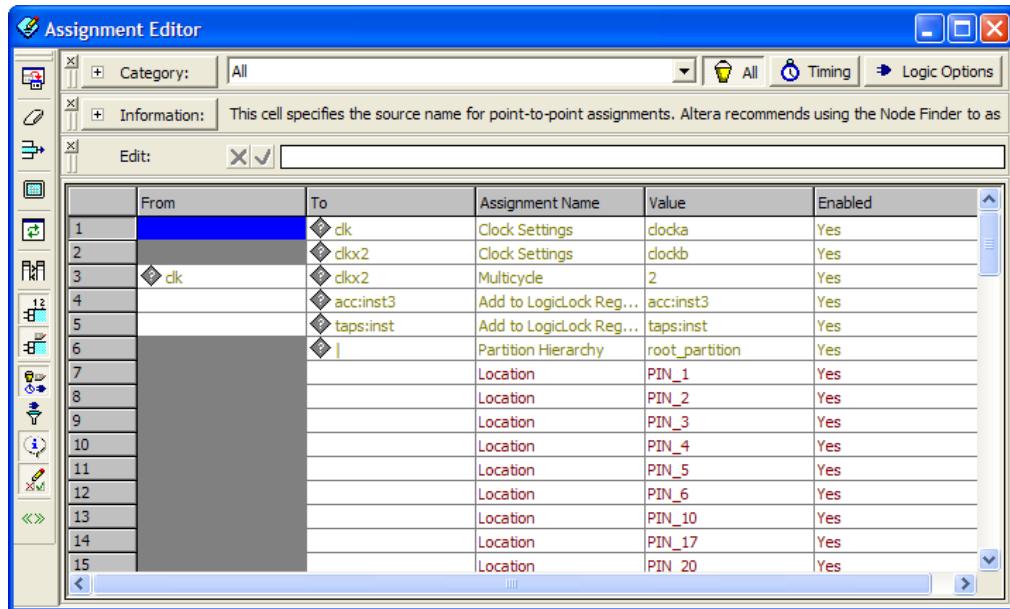
For Information About	Refer To
Using the Assignment Editor	*Assignment Editor* chapter in volume 2 of the *Quartus II Handbook*
	"About the Assignment Editor" and "Working with Assignments in the Assignment Editor" in Quartus II Help

Using the Pin Planner

The Pin Planner allows you to make assignments to pins and groups of pins. It includes a package view of the device with different colors and symbols that represent the different types of pins and additional symbols that represent I/O banks. The symbols used in the Pin Planner are very similar to the symbols used in device family data sheets. It also includes tables of pins and groups. Figure 3 shows the Pin Planner.

Figure 3. Pin Planner
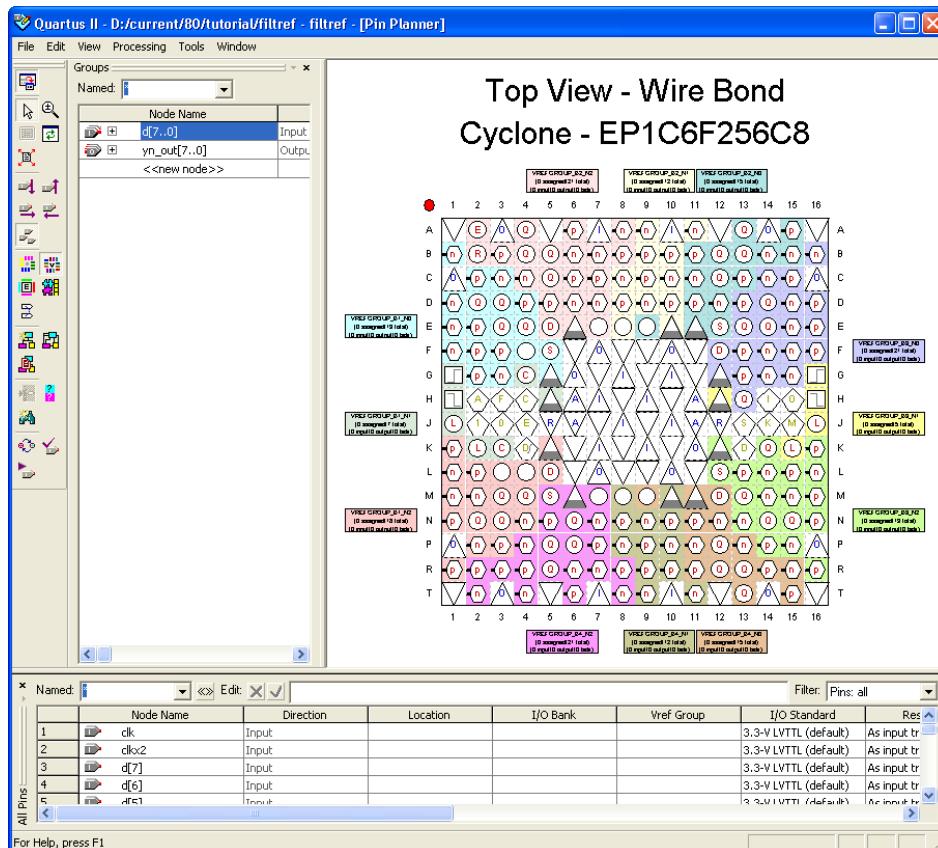
By default, the Pin Planner displays a **Groups** list, an **All Pins** list, and a package view diagram of the device. You can make pin assignments by dragging pins from the **Groups** list and **All Pins** list to available pin or I/O bank locations in the package diagram. In the **All Pins** list, you can filter the node names, change the I/O standards, and specify options for reserved pins. You can also filter the **All Pins** list to display only unassigned pins, so you can change the node name and direction for user-added nodes. You can also specify options for reserved pins.

You can also display the properties and available resources for the selected pin, and can display a legend that explains the different colors and symbols that are used in the Pin Planner.

For Information About	Refer To
Using the Pin Planner to assign pins	*I/O Management* chapter in volume 2 of the *Quartus II Handbook*
	"Assigning Pins" in Quartus II Help

The Settings Dialog Box

You can use the **Settings** dialog box to specify general project-wide options and synthesis, fitting, simulation, timing analysis, power analysis, and debugging options for a project.

You can perform the following types of tasks in the **Settings** dialog box:

■ **Modify project settings**: specify and view the current top-level entity for project and revision information; add and remove files from the project; specify custom user libraries; specify device options for package, pin count, and speed grade; and specify migration devices.

■ **Specify EDA tool settings**: specify EDA tools for design entry/synthesis, simulation, timing analysis, board-level verification, formal verification, physical synthesis, and related tool options.

■ **Specify Analysis & Synthesis settings**: project-wide settings for Analysis & Synthesis, Verilog HDL and VHDL input settings, default design parameters, and synthesis netlist optimizations options.

■ **Specify compilation process settings**: options for smart compilation, parallel compilation, incremental compilation, saving node-level netlists, and enabling or disabling the OpenCore Plus evaluation feature.

■ **Specify Fitter settings**: timing-driven compilation options, Fitter effort, project-wide Fitter logic options assignments, and physical synthesis netlist optimizations.

■ **Specify Simulator settings**: mode (functional or timing), source vector file, simulation period, and simulation detection options.

■ **Specify PowerPlay Power Analyzer settings:** input file type, output file type, and default toggle rates, as well as operating conditions such as junction temperature, cooling solution requirements, and device characteristics.

■ **Specify Design Assistant and SignalTap II settings**: enable the Design Assistant and enable the SignalTap II Logic Analyzer; specify a SignalTap II File (**.stp**) name.

For Information About	Refer To
Assigning project-wide settings with the Settings dialog box	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial

Making Timing Constraints

The TimeQuest Timing Analyzer accepts constraints in Synopsys Design Constraint format to define the parameters for analysis. You can make timing constraints using the commands in the TimeQuest Timing Analyzer GUI or equivalent Tcl commands. For more information on the TimeQuest Timing Analyzer, refer to "Chapter 9: Timing Analysis" on page 123.

Creating Design Partitions

You can designate separate hierarchical sections of your design as design partitions to compile incrementally, without affecting the rest of the project. For more information about incremental compilation, see the following sections:

■ "Top-Down Incremental Compilation Flow" on page 12 in Chapter 1, "Design Flow."
■ "Using Incremental Compilation" on page 86 in Chapter 6, "Place and Route."

The Project Navigator, the Design Partition Planner, and the Design Partitions window allow you to assign design partitions.

# Creating Design Partitions in the Project Navigator

To specify a selected instance of an entity as a design partition in the **Hierarchy** tab of the Project Navigator, right-click the entity, point to **Design Partition**, and then click **Set as Design Partition**.

To make a LogicLock assignment for a partition, drag the partition from the Project Navigator window directly to the LogicLock Regions window or to a LogicLock region in the Timing Closure Floorplan. You can also right-click the partition/entity in the Project Navigator, point to **LogicLock Region**, and then click **Create New LogicLock Region**.

Creating Design Partitions in the Design Partition Planner

The Design Partition Planner allows you to view a graphical representation of the entities in a design, and specify entities as design partitions. To create a design partition in the Design Partition Planner, on the Tools menu, click **Design Partition Planner**. Right-click the entity and click **Set as Design Partition**.

Figure 4. Design Partition Planner
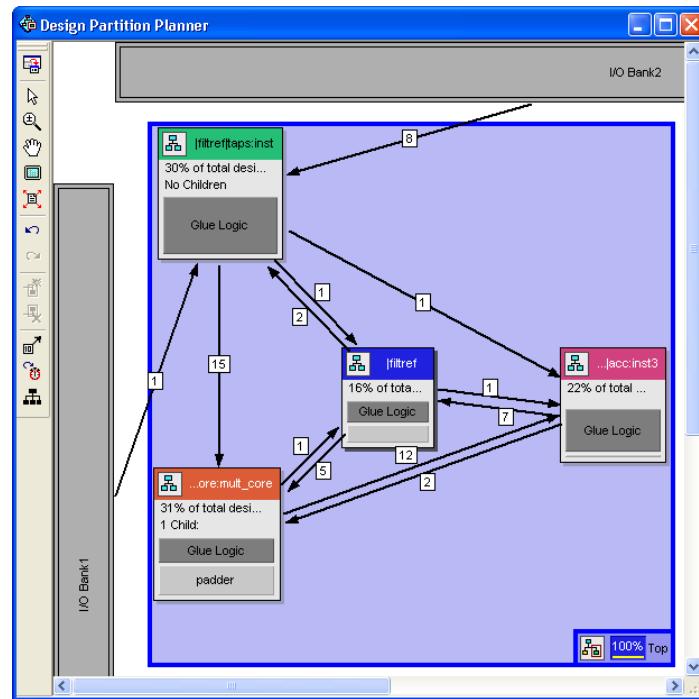
Creating Design Partitions with the Design Partitions Window

To specify an entity as a design partition, on the Assignments menu, click **Design Partitions Window**. Figure 5 shows the Design Partitions window.

Figure 5. Design Partitions Window

Right-click the partition and click **Rename** if you want to use a name other than the full hierarchy path name. To generate an incremental compilation Tcl script, right-click the partition and click **Generate Incremental Compilation Tcl Script**.

The Design Partitions window allows you to specify one of the following options for **Netlist Type**:

- **Source File**—directs the Compiler to compile from source design files
- **Post-Synthesis**—preserves synthesis results for the partition (default option for new partitions)
- **Post-Fit**—preserves placement results for the partition
- **Empty**—skips compilation for the partition

You can specify the netlist type from the list in the **Netlist Type** column or by right-clicking the partition and clicking **Properties**.

If you want to make a LogicLock assignment for a partition, you can drag the partition from the Design Partitions window directly to the LogicLock Regions window or to a LogicLock region in the Chip Planner.

| For Information About	Refer To
Assigning design partitions and using incremental compilation	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	Best Practices for Incremental Compilation Partitions and Floorplan Assignments chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" in Quartus II Help

Importing Assignments

To import assignments into a project in the Quartus II software, click **Import Assignments** on the Assignments menu.

The **Import Assignments** dialog box allows you to specify the file that contains the assignments to import and the specific types of assignments (assignment categories) to import into the Quartus II Settings File for the

current project revision. Click **Advanced** in the **Import Assignments** dialog box to specify the type of the assignments to import, specify global or instance-level assignments to import, and specify how the assignments affect the current design. Use this dialog box to import settings files, Comma-Separated Value Files, and FPGA Xchange Files (**.fx**). You can also create a backup of the current Quartus II Settings File for the design before importing assignments (Figure 6).

Figure 6. Import Assignments Dialog Box
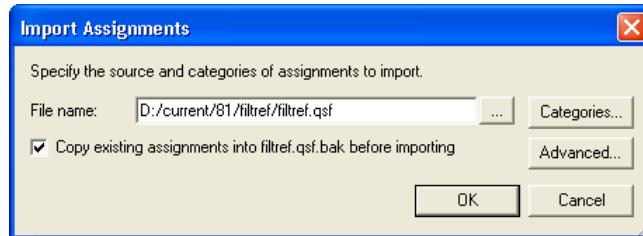
You can use this command to import settings and assignments from other projects created in the Quartus II software into your current project. For example, you can use this command to import pin assignments from a previous Quartus II project into the current Quartus II project.

For more information on using the **Import Assignments** command to import LogicLock region assignments, refer to "Exporting & Importing Partitions for Bottom-Up Design Flows" on page 106 in Chapter 7, "Block-Based Design."

| For Information About	Refer To
Importing Assignments	"Importing and Exporting Assignments" in Quartus II Help
	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial

Chapter
Five

Synthesis
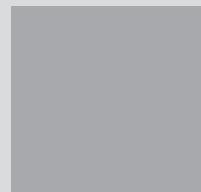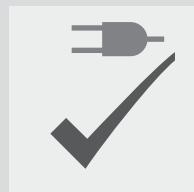
5

Introduction

You can use the Analysis & Synthesis module of the Compiler to analyze your design files and create the project database. Analysis & Synthesis uses Quartus II Integrated Synthesis to synthesize your Verilog Design Files (**.v**) or VHDL Design Files (**.vhd**). If you prefer, you can use other EDA synthesis tools to synthesize your Verilog HDL or VHDL design files, and then generate an EDIF netlist file (**.edf**) or a Verilog Quartus Mapping File (**.vqm**) that can be used with the Quartus II software. Figure 1 shows the synthesis design flow.

Figure 1. Synthesis Design Flow
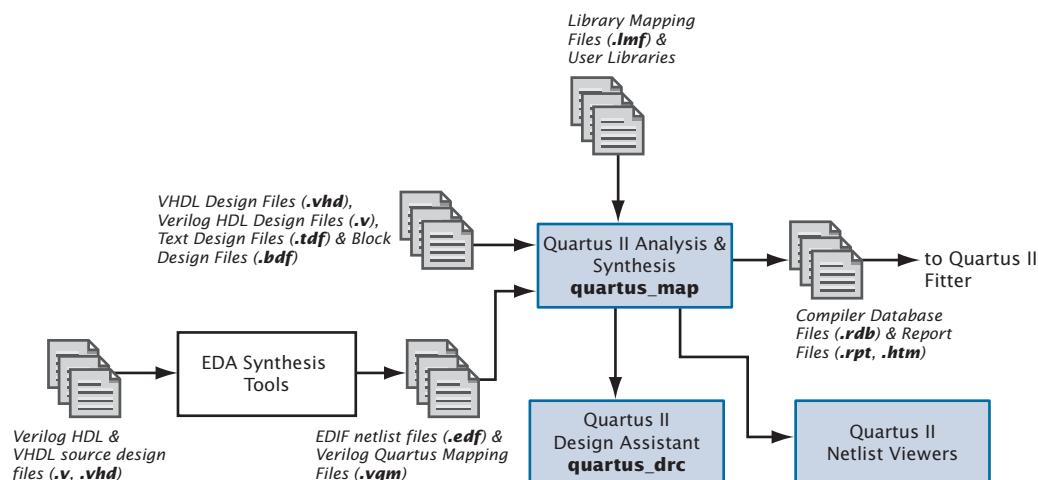
You can start a full compilation in the Quartus II software, which includes the Analysis & Synthesis module, or you can start Analysis & Synthesis separately. You can perform an Analysis & Elaboration to check a design for syntax and semantic errors without performing a complete Analysis & Synthesis or use the **Analyze Current File** command on the Processing menu to check a single design file for syntax errors.

For more information about starting a full compilation or starting Compiler modules individually, refer to "Graphical User Interface Design Flow" on page 3 and "Introduction" on page 38 in Chapter 3, "Command-Line And Tcl Design Flows."

☞ **Using the quartus_map executable**

You can also run Analysis & Synthesis separately at the command prompt or in a script that contains the **quartus_map** executable. The **quartus_map** executable creates a new project if one does not already exist.

The **quartus_map** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_map** executable, type one of the following commands at the command prompt:

```
quartus_map -h ↵
quartus_map --help ↵
quartus_map --help=<topic name> ↵
```

Using Quartus II Verilog HDL & VHDL Integrated Synthesis

You can use Analysis & Synthesis to analyze and synthesize Verilog HDL and VHDL designs. Analysis & Synthesis includes Quartus II Integrated Synthesis, which fully supports the Verilog HDL and VHDL languages and provides options to control the synthesis process.

Analysis & Synthesis supports the Verilog-1995 (IEEE Std. 1364-1995) and Verilog-2001 (IEEE Std. 1364-2001) standards, a subset of features of the SystemVerilog-2005 (IEEE Std. 1800-2005) standard, and also supports the VHDL 1987 (IEEE Std. 1076-1987) and 1993 (IEEE Std. 1076-1993) standards. You can select which standard to use; Analysis & Synthesis uses Verilog-2001 and VHDL 1993 by default. If you are using another EDA synthesis tool, you can also specify a Library Mapping File (**.lmf**) that the Quartus II software should use to map non–Quartus II functions to Quartus II functions. You can specify these and other options in the **Verilog HDL Input** and **VHDL Input** pages, which are under **Analysis & Synthesis Settings** in the **Settings** dialog box. These pages are shown in Figure 2.

For Information About	Refer To
Quartus II Verilog HDL and VHDL Synthesis support	"Quartus II Verilog HDL Support," "Quartus II VHDL Support," and "Quartus II Support for SystemVerilog 2005" in Quartus II Help

Figure 2. Verilog HDL & VHDL Input Pages of Settings Dialog Box
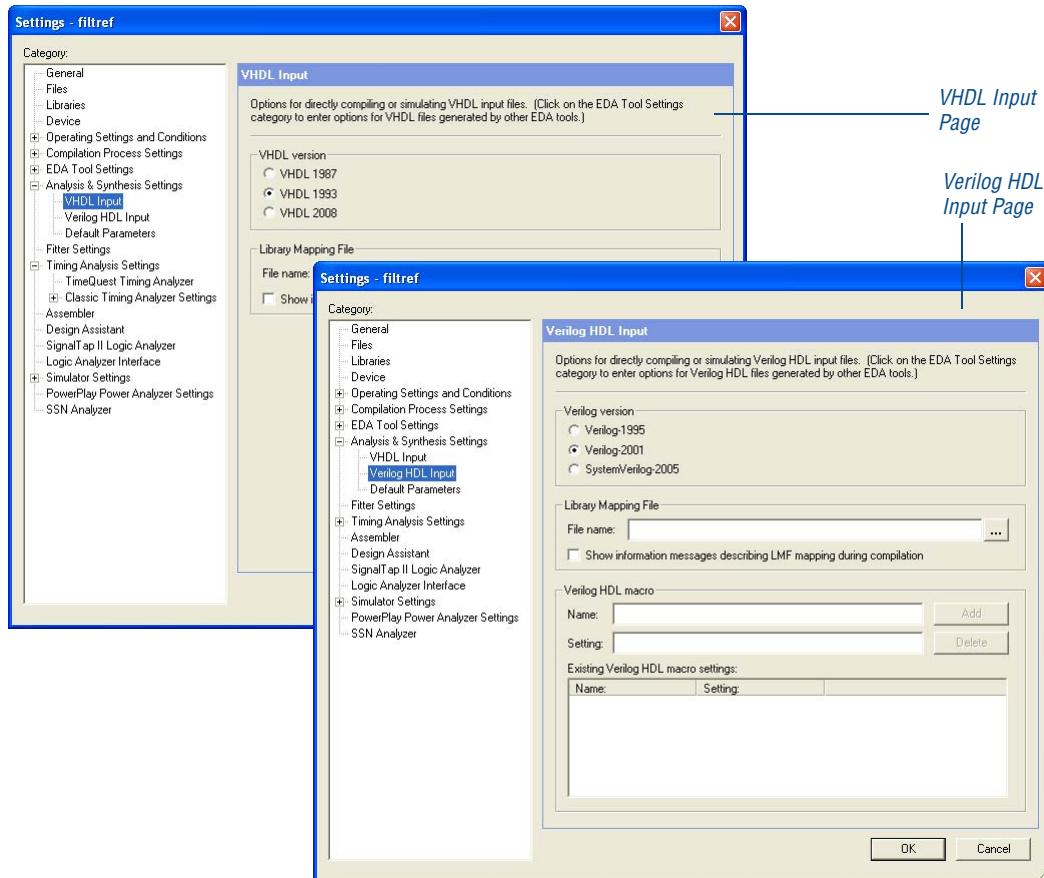
You can compile most Verilog HDL and VHDL designs successfully with Quartus II Integrated Synthesis and other EDA synthesis tools. If your design instantiates Altera megafunctions, library of parameterized modules (LPM) functions, or intellectual property (IP) megafunctions in a third-party EDA tool, you need to use a hollow-body or black box file. When you are

instantiating megafunctions for Quartus II Analysis & Synthesis, however, you can instantiate the megafunction directly without using a black box file. For more information about instantiating megafunctions, refer to "Instantiating Megafunctions in the Quartus II Software" on page 32 and "Instantiating Megafunctions in EDA Tools" on page 34 in Chapter 2, "Design Entry."

Add the design files when creating a project with the New Project Wizard, or in the **Files** page of the **Settings** dialog box. When you add files to the project, ensure you add them in the order you want Quartus II Analysis & Synthesis to process them. In addition, if your design is coded in VHDL, specify the VHDL library for the design in the **Properties** dialog box that is available from the **Files** page. If you do not specify a VHDL library, Analysis & Synthesis compiles VHDL entities into the **work** library. For more information about adding files to a project, refer to "Creating a Design" on page 24 in Chapter 2, "Design Entry."

Analysis & Synthesis builds a single project database that integrates all the design files in a design entity or project hierarchy. The Quartus II software uses this database for the remainder of project processing. Other Compiler modules update the database until it contains the fully optimized project. In the beginning, the database contains only the original netlists; at the end, it contains a fully optimized, fitted project, which is used to create one or more files for timing simulation, timing analysis, and device programming.

As it creates the database, the analysis stage of Analysis & Synthesis examines the logical completeness and consistency of the project, and checks for boundary connectivity and syntax errors. Analysis & Synthesis also synthesizes and performs technology mapping on the logic in the design entity or project's files. It infers flipflops, latches, and state machines from Verilog HDL and VHDL. It creates state assignments for state machines and makes choices that minimize resources usage. In addition, it replaces operators such as + or - with modules from the Altera library of parameterized modules (LPM) functions, which are optimized for Altera devices.

Analysis & Synthesis uses several algorithms to minimize gate count, remove redundant logic, and utilize the device architecture as efficiently as possible. You can customize synthesis by using logic option assignments. Analysis & Synthesis also applies logic synthesis techniques to help implement timing requirements for a project and optimize the design to meet these requirements.

The Messages window and the Messages section of the Report window display any messages Analysis & Synthesis generates. The Status window and Tasks window display the time spent performing Analysis & Synthesis during project compilation.

For Information About	Refer To
Verilog HDL constructs supported in the Quartus II software	"Quartus II Verilog HDL Support" in Quartus II Help
VHDL constructs supported in the Quartus II software	"Quartus II VHDL Support" in Quartus II Help
Using Quartus II Integrated Synthesis	*Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*

Using Other EDA Synthesis Tools

You can use other EDA synthesis tools to synthesize your Verilog HDL or VHDL designs, and then generate EDIF netlist files or Verilog Quartus Mapping files that can be used with the Quartus II software.

Altera provides libraries for use with many EDA synthesis tools. Altera also provides NativeLink support for many tools. NativeLink technology facilitates the seamless transfer of information between the Quartus II software and other EDA tools and allows you to run EDA tools automatically from within the Quartus II graphical user interface.

If you have created assignments or constraints using other EDA tools, you can use Tcl commands or scripts to import those constraints into the Quartus II software with your design files. Many EDA tools generate an assignment Tcl script automatically.

Table 1. Quartus II–Supported EDA Synthesis Tools (Part 1 of 2)

Synthesis Tool Name	EDIF Netlist File (.edf)	Verilog Quartus Mapping File (.vqm)	NativeLink Support
Mentor Graphics LeonardoSpectrum	✓		✓
Mentor Graphics Precision Synthesis	✓		✓

Table 1. Quartus II–Supported EDA Synthesis Tools (Part 2 of 2)

Synthesis Tool Name	EDIF Netlist File (.edf)	Verilog Quartus Mapping File (.vqm)	NativeLink Support
Synopsys Design Compiler	✓		
Synopsys Synplify	✓	✓	✓
Synopsys Synplify Pro	✓	✓	✓
Agility DK Design Suite	✓		

In the **Design Entry/Synthesis** page under **EDA Tool Settings** in the **Settings** dialog box, you can specify EDA synthesis tools, and whether an EDA tool that has NativeLink support should be run automatically as part of full compilation. The **Design Entry/Synthesis** page also allows you to specify other options for EDA synthesis tools (Figure 3).

Figure 3. Design Entry/Synthesis Page of Settings Dialog Box
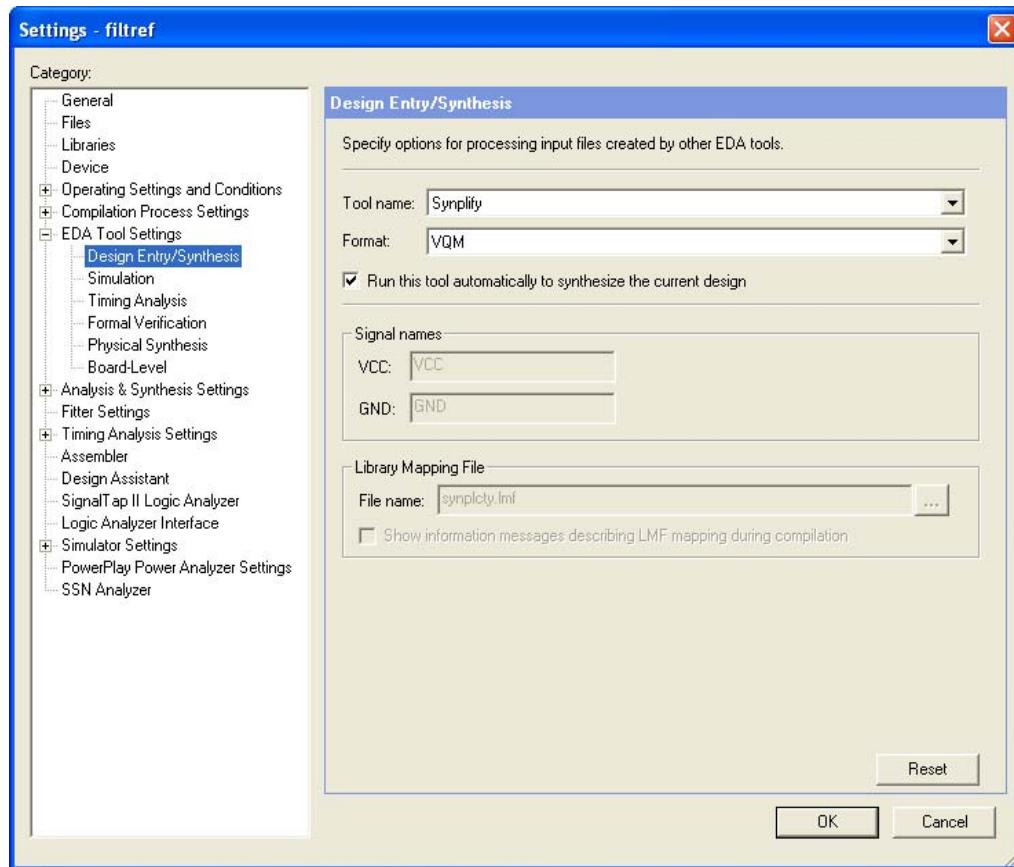
To run an EDA synthesis tool listed in the **Design Entry/Synthesis** page from within the Quartus II software, on the Processing menu, click **Start**, and then click **Start EDA Synthesis**. Many EDA tools also allow you to run the Quartus II software from within the EDA tool's GUI. Refer to your EDA tool documentation for more information.

For Information About	Refer To
Using Synopsys Synplify software	*Synopsys Synplify Support* chapter in volume 1 of the *Quartus II Handbook*
Using Mentor Graphics LeonardoSpectrum software	*Mentor Graphics LeonardoSpectrum Support* chapter in volume 1 of the *Quartus II Handbook*
Using Mentor Graphics Precision RTL Synthesis software	*Mentor Graphics Precision Synthesis Support* chapter in volume 1 of the *Quartus II Handbook*

Controlling Analysis & Synthesis

You can use the following options and features to control Quartus II Analysis & Synthesis:

■　Compiler directives and attributes
■　Quartus II logic options
■　Quartus II synthesis netlist optimization options

Using Compiler Directives and Attributes

The Quartus II software supports compiler directives, also called pragmas. You can include compiler directives, such as `translate_on` and `translate_off`, in Verilog HDL or VHDL code as comments. Synthesis tools parse these HDL comments to drive the synthesis process in a particular manner. Other tools, such as simulators, ignore these directives and treat them as comments.

You can also use attributes, which are sometimes known as pragmas or directives, that drive the synthesis process for a a specific design element. Some attributes are also available as Quartus II logic options.

For Information About	Refer To
Using Compiler directives and attributes with Quartus II Integrated Synthesis	*Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*

Using Quartus II Logic Options

Quartus II logic options allow you to set attributes without editing the source code. You can assign individual Quartus II logic options in the Assignment Editor, and you can specify global Analysis & Synthesis logic options for the project in the **Analysis & Synthesis Settings** page of the **Settings** dialog box (Figure 4).

Figure 4. Analysis & Synthesis Settings Page of Settings Dialog Box
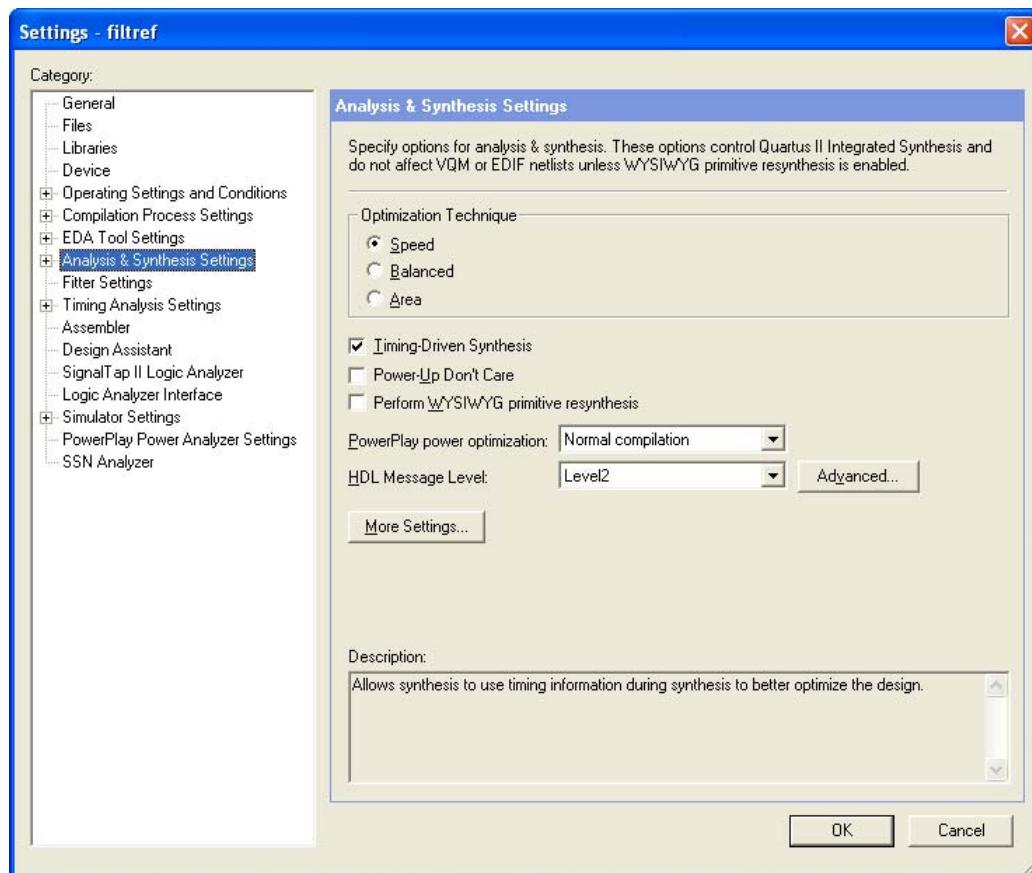
The Quartus II logic options that are available on the **Analysis & Synthesis Settings** page allow you to specify that the Compiler should optimize for speed or area, or perform a "balanced" optimization, which attempts to

achieve the best combination of speed and area. It also provides other options, such as options that control timing-driven synthesis, the logic level for power-up, and the removal of duplicate or redundant logic.

For Information About	Refer To
Using Quartus II logic options to control synthesis	"Working With Assignments in the Assignment Editor" and "Specifying Default Logic Options and Parameters" in Quartus II Help
Creating a logic option assignment	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial
Using Quartus II synthesis options and logic options that affect synthesis	*Quartus II Integrated Synthesis* chapter in volume 1 of the *Quartus II Handbook*

Using Quartus II Synthesis Netlist Optimization Options

Quartus II synthesis optimization options allow you to optimize the netlist during synthesis for many of the Altera device families. These optimization options are additional to the optimization that occurs during a standard compilation, and occur during the Analysis & Synthesis stage of a full compilation. These optimizations make changes to your synthesis netlist that are generally beneficial for area and speed. The **Physical Synthesis Optimizations** page in the **Settings** dialog box allows you to specify netlist optimization options.

For more information about synthesis netlist optimization, refer to "Using Netlist Optimizations to Achieve Timing Closure" on page 142 in Chapter 10, "Timing Closure."

For Information About	Refer To
Using Quartus II synthesis and netlist optimization options	*Netlist Optimizations and Physical Synthesis* in volume 2 of the *Quartus II Handbook*

Using the Design Assistant to Check Design Reliability

The Quartus II Design Assistant allows you to check the reliability of your design, based on a set of design rules. The Design Assistant is especially useful for checking the reliability of a design before migrating to HardCopy devices. The **Design Assistant** page of the **Settings** dialog box allows you to specify which design reliability guidelines you want to use when checking your design (Figure 5).

Figure 5. Design Assistant Page of Settings Dialog Box
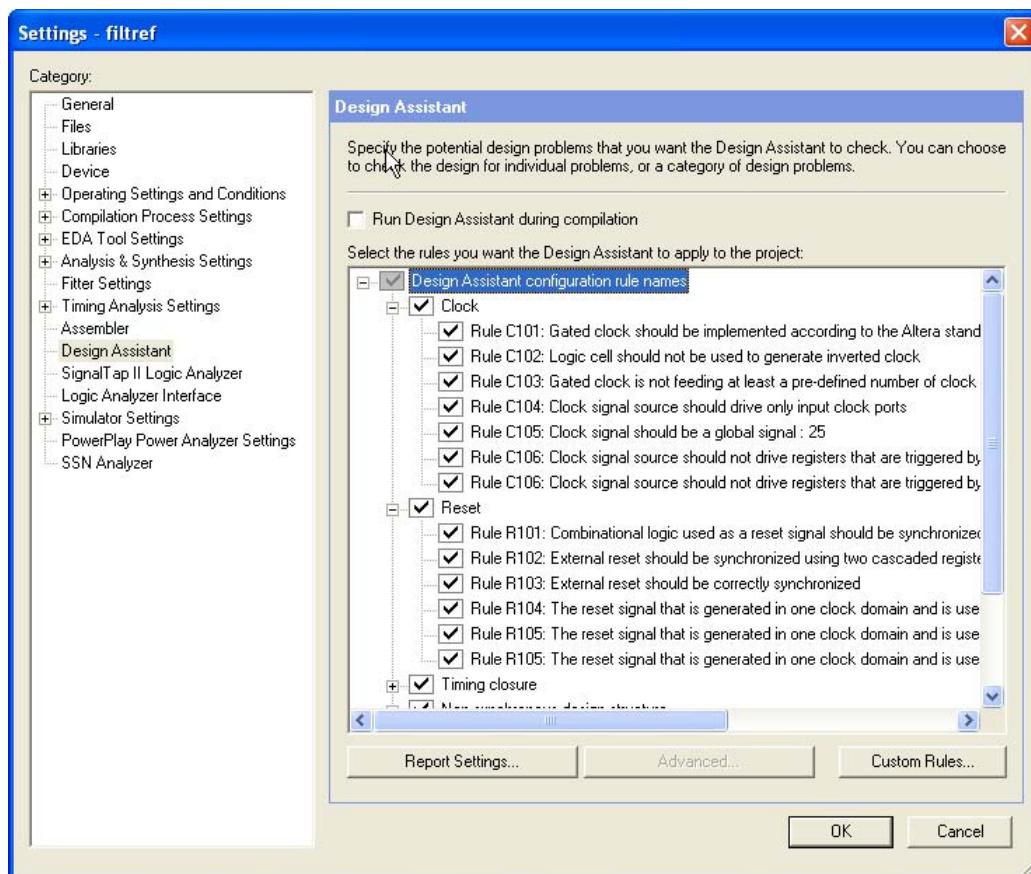
☞ **Using the quartus_drc executable**

You can also run the Design Assistant separately at the command prompt or in a script by using the **quartus_drc** executable. You must run the Quartus II Fitter executable **quartus_fit** before running the Design Assistant.

The **quartus_drc** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_drc** executable, type one of the following commands at the command prompt:

```
quartus_drc -h ↵
quartus_drc -help ↵
quartus_drc --help=<topic name> ↵
```

You can also improve design optimization by following good synchronous design practices and by following Quartus II coding style guidelines.

For Information About	Refer To
Using the Quartus II Design Assistant	"Analyzing Designs with the Design Assistant" and "About the Design Assistant" in Quartus II Help
Using Quartus II synthesis options, following synchronous design practices, and following coding style guidelines	*Design Recommendations for Altera Devices and the Quartus II Design Assistant*, *Recommended HDL Coding Styles,* and *Quartus II Integrated Synthesis* chapters in volume 1 of the *Quartus II Handbook*
	"AHDL, VHDL, and Verilog HDL Style Guide" in Quartus II Help

# Analyzing Synthesis Results With the Netlist Viewers

The Quartus II RTL Viewer and State Machine Viewer provide graphical representations of your design. To run either of these viewers for a Quartus II project, you must first perform Analysis & Synthesis or perform a full compilation.

The RTL Viewer

To display the RTL Viewer, on the Tools menu, point to **Netlist Viewers**, and then click **RTL Viewer**. In addition to the schematic view, the RTL Viewer has a hierarchy list that lists the instances, primitives, pins, and nets for the entire design netlist (Figure 6).

Figure 6. RTL Viewer
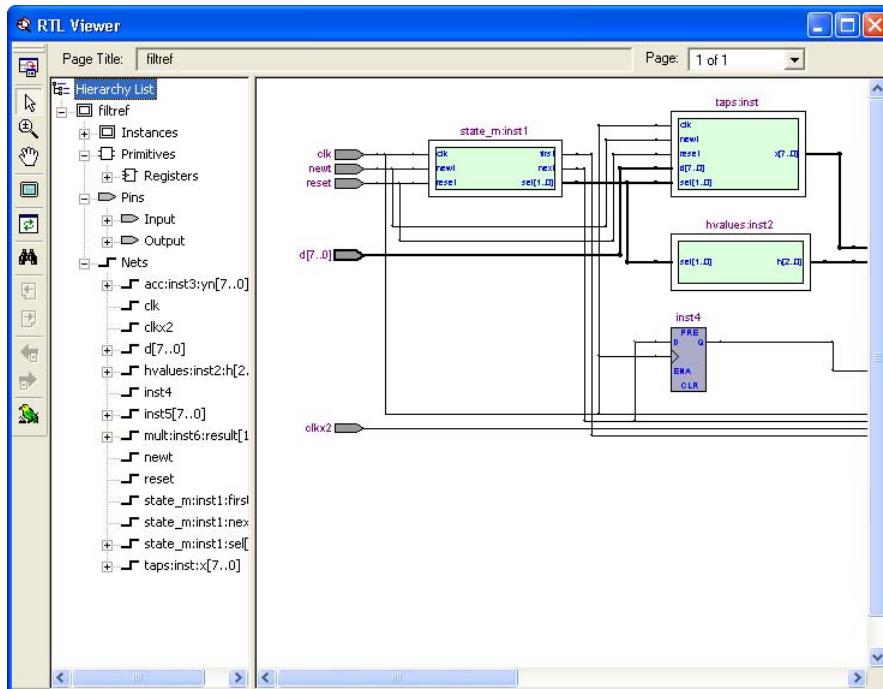
The RTL Viewer displays the Analysis & Elaboration results for Verilog HDL or VHDL designs, and AHDL Text Design Files (**.tdf**), Block Design Files (**.bdf**), and Graphic Design Files (**.gdf**). For Verilog Quartus Mapping Files or EDIF netlist files that were generated from other EDA synthesis tools, the RTL Viewer displays the hierarchy for the atom representations of WYSIWYG primitives.

You can select one or more items in the hierarchy list to highlight in the schematic view. The RTL Viewer allows you to adjust the view or focus by zooming in and out to see different levels of detail, searching through the RTL Viewer for a specific name, moving up or down in the hierarchy, or going to the source that feeds the selected net. If you want to adjust the fan-in or fan-out display, you can expand or collapse it. You can use the tooltips to see node and source information for individual items. You can also select a node in the RTL Viewer and locate it in the design file, Assignment Editor, Chip Planner, Resource Property Editor, or Technology Map Viewer, depending on which locations are available for that node.

If a design is large, the RTL Viewer partitions it into multiple pages for display. The **Netlist Viewers** page of the **Options** dialog box allows you to specify, in number of nodes or ports, how much of the design the RTL Viewer displays on each page. You can navigate through pages in the RTL Viewer by clicking **Next Page** and **Previous Page** or by clicking **Go To** on the Edit menu.

The **Filter** command allows you to filter the view to show the sources, and destinations of the selected node(s) or net(s). You can also filter the view to show the paths and nodes between two selected nodes. Each filter you choose creates a new filtered page in the RTL Viewer. Navigate through the filtered pages and the original page of the design with the **Forward** and **Back** buttons.

The State Machine Viewer

The State Machine Viewer allows you to view state machine diagrams for the relevant logic in your design. If your project has a state machine, on the Tools menu, point to **Netlist Viewers**, and then click **State Machine Viewer**. You can also display the State Machine Viewer by double-clicking an instance symbol in the RTL Viewer Window (Figure 7).

Figure 7. State Machine Instance in RTL Viewer

Double-clicking a state machine instance symbol in the RTL Viewer opens the State Machine Viewer window
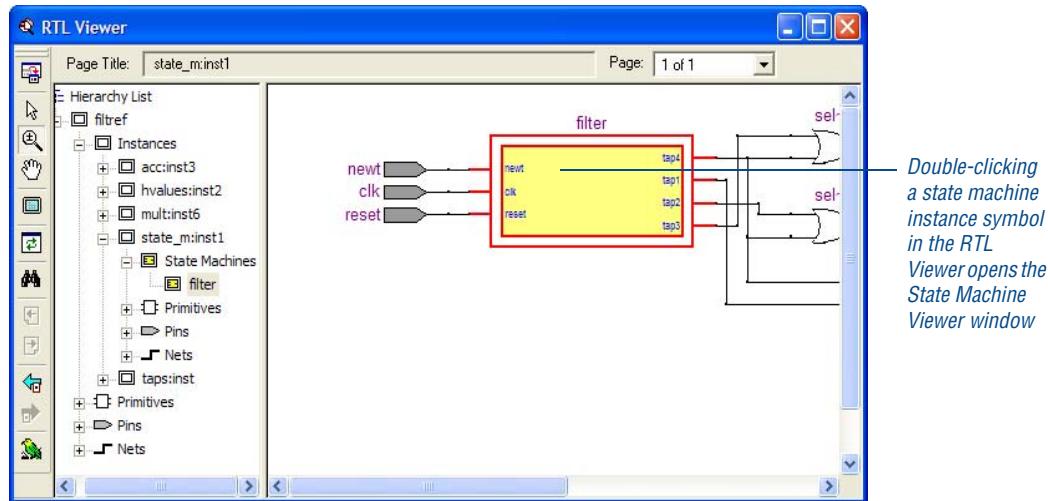
The State Machine Viewer includes a schematic view and a transition table (Figure 8).

Figure 8. State Machine Viewer

Schematic view

Double circles indicate nodes that have connections to outside logic

Transition table shows source and destination states and transition conditions
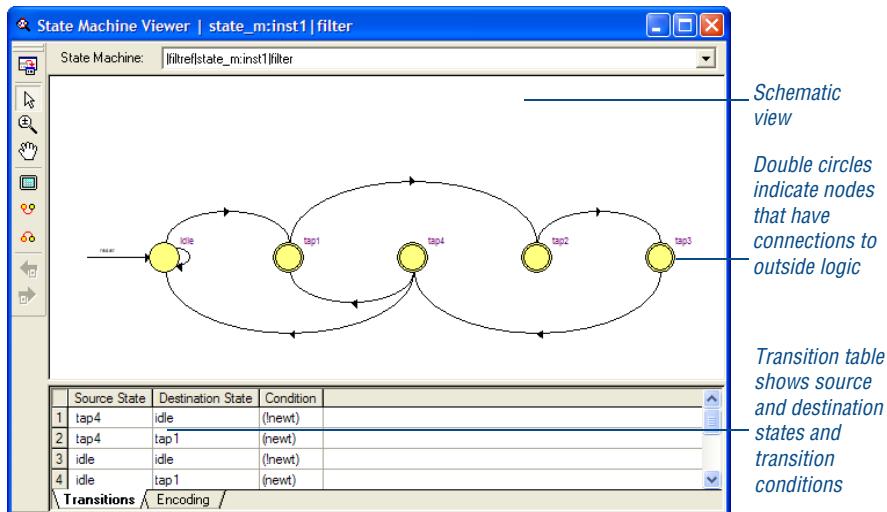
When you select a cell in a transition table, the corresponding state or transition is highlighted in the schematic view. Likewise, when you select a state or transition in the schematic view, the corresponding cell is highlighted in the transition table. The schematic view allows you to zoom in and out, scroll up and down, and highlight fan-in and fan-out. In the transition table, you can copy selected cells or the entire table to any text editor. You can also align and sort data that appears in the table columns.

If you decide to make changes to your design after viewing it with the RTL Viewer, you should perform Analysis & Elaboration again so you can analyze the updated design in the RTL Viewer.

The Technology Map Viewer

The Quartus II Technology Map Viewer provides a low-level, or atom-level, technology-specific schematic representation of a design. To run the Technology Map Viewer for a Quartus II project, you must first perform Analysis & Synthesis or a full compilation. After you have successfully performed Analysis & Synthesis, you can display the Technology Map Viewer by pointing to **Netlist Viewers** on the Tools menu, and then clicking **Technology Map Viewer**. The Technology Map Viewer includes a schematic view, and also includes a hierarchy list, which lists the instances, primitives, pins, and nets for the entire design netlist (Figure 9).

Figure 9. Technology Map Viewer
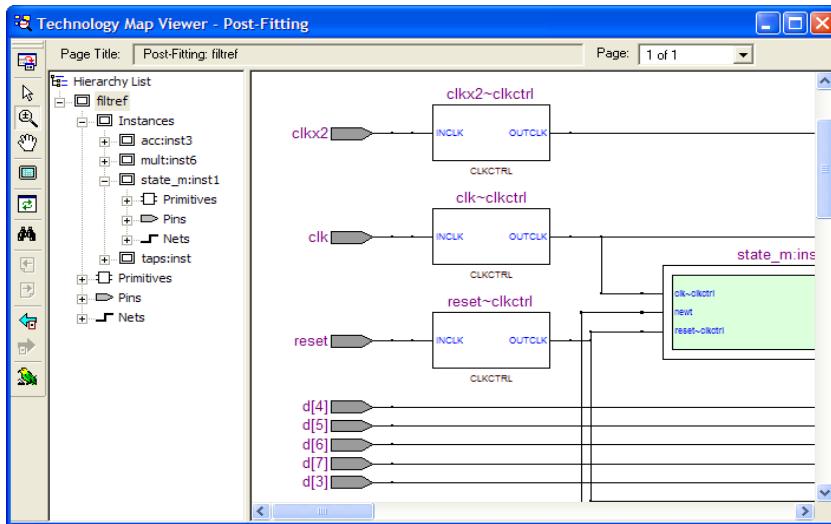
You can also use the Technology Map Viewer to display post-Analysis & Synthesis mapping and compare those results to the results from a full compilation. Display the results from Analysis & Synthesis by pointing to **Netlist Viewers** on the Tools menu, and then clicking **Technology Map Viewer (Post Mapping)**.

> ☞ **Technology Map Viewer Displays**
>
> If you have run only Analysis & Synthesis and have not performed a full compilation of your design, both of the Technology Map Viewer commands display the same post-mapping information.

In the Technology Map Viewer, you can select one or more items in the hierarchy list to highlight in the schematic view. The Technology Map Viewer allows you to navigate the view in much the same way as the RTL Viewer; see "Analyzing Synthesis Results With the Netlist Viewers" on page 76. The tooltips in the Technology Map Viewer display equation information as well as node and source information.

After performing timing analysis or performing a full compilation that includes timing analysis, you can also use the Technology Map Viewer to view the nodes that make up the timing path, including information about total delay and individual node delay. See "Viewing Timing Delays with the Technology Map Viewer" on page 132 in Chapter 9, "Timing Analysis."

| For Information About	Refer To
Using the Quartus II Technology Map Viewer	*Analyzing Designs with Quartus II Netlist Viewers* chapter in volume 1 of the *Quartus II Handbook*
	"About the Netlist Viewers" in Quartus II Help

Chapter Six

Place and Route
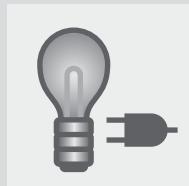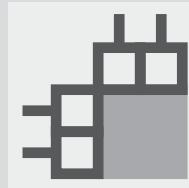
6

Introduction
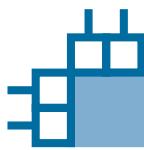
The Quartus II Fitter places and routes your design, which is also referred to as "fitting" in the Quartus II software. Using the database that has been created by Analysis & Synthesis, the Fitter matches the logic and timing requirements of the project with the available resources of the target device. It assigns each logic function to the best logic cell location for routing and timing, and selects appropriate interconnection paths and pin assignments. Figure 1 shows the place and route design flow.

Figure 1. Place and Route Design Flow
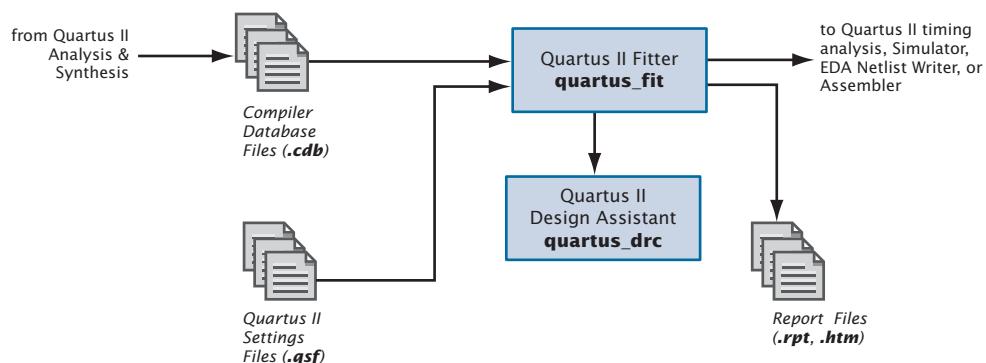
If you have made resource assignments in your design, the Fitter attempts to match those resource assignments with the resources on the device, tries to meet any other constraints you have set, and then attempts to optimize the remaining logic in the design. If you have not set any constraints on the design, the Fitter automatically optimizes it. If it cannot find a fit, the Fitter terminates compilation and issues an error message.

In the **Compilation Process Settings** page of the **Settings** dialog box, you can specify whether you want to use a normal compilation or smart compilation. With a "smart" compilation, the Compiler creates a detailed database that can help future compilations run faster, but may consume extra disk space. During a smart recompilation, the Compiler evaluates the changes made to the current design since the last compilation and then runs only the Compiler modules that are required to process those changes. If you make any changes to the logic of a design, the Compiler uses all modules during processing.

You can start a full compilation in the Quartus II software, which includes the Fitter module, or you can start the Fitter separately. You must run Analysis & Synthesis successfully before starting the Fitter separately. For information about performing a full compilation, refer to "Graphical User Interface Design Flow" on page 3 in Chapter 1, "Design Flow."

☞ Using the quartus_fit executable

You can also run the Fitter separately at the command prompt or in a script by using the **quartus_fit** executable. You must run the Analysis & Synthesis executable **quartus_map** before running the Fitter.

The **quartus_fit** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_fit** executable, type one of the following commands at the command prompt:

```
quartus_fit -h ↵
quartus_fit -help ↵
quartus_fit --help=<topic name> ↵
```


The Status window and the Tasks window each display the time spent processing in the Fitter during project compilation, as well as the processing time for any other modules you may have been running (Figure 2).

Figure 2. Status Window
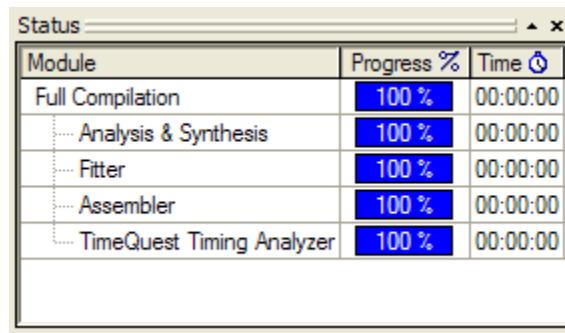
Figure 3 shows the Tasks window during compilation, while the TimeQuest timing analyzer is running.

Figure 3. Tasks Window During Compilation
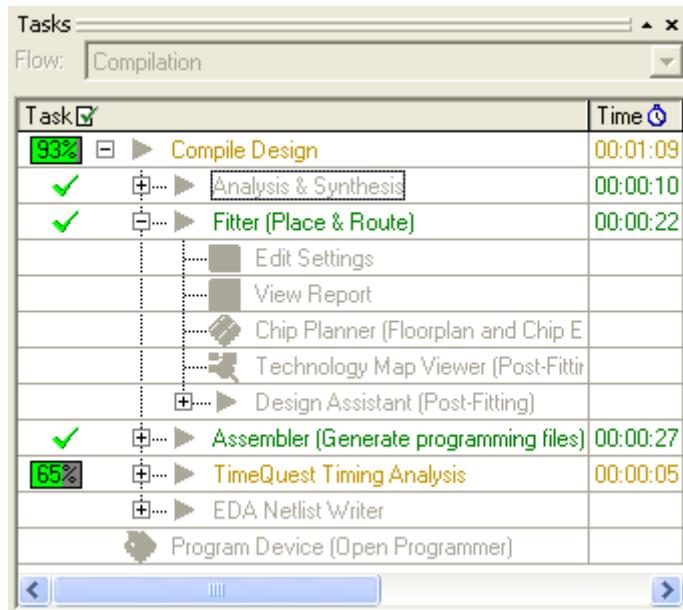
Using Incremental Compilation

The Quartus II software performs incremental compilation to reuse previous compilation results for unchanged entities in the design. Incremental compilation can be used as part of top-down and bottom-up design flows. For more information, refer to "Design Methodologies and Planning" on page 11 in Chapter 1, "Design Flow."

The following steps describe the basic flow for performing an incremental compilation:

1. Perform Analysis & Elaboration.

2. Specify one or more entities of the project as partitions. Refer to "Creating Design Partitions" on page 57 in Chapter 4, "Constraint Entry."

3. Set **Full Incremental compilation** to **Incremental compilation** mode.

4. Set the appropriate **Netlist Type** for the partitions. To preserve compilation and placement results, set the **Netlist Type** for the partitions to **Post-Fit**.

5. Assign each partition to a physical location on the device by using the Chip Planner and LogicLock assignments. Refer to "Using LogicLock Regions in Top-Down Incremental Compilation Flows" on page 105 and "Exporting & Importing Partitions for Bottom-Up Design Flows" on page 106 in Chapter 7, "Block-Based Design."

6. Compile the design.

7. Make changes to the design or design settings, as needed.

8. Compile the design again. Only the partitions that have changed are recompiled.

| For Information About	Refer To
Using Quartus II incremental compilation	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	Best Practices for Incremental Compilation Partitions and Floorplan Assignments chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" in Quartus II Help
	"Module 7: Incremental Compilation" in the Quartus II Interactive Tutorial

Analyzing Fitting Results

The Quartus II software offers several tools to help you analyze the results of compilation and fitting. The Messages window and Report window provide fitting results information. The Chip Planner allows you to view fitting results and make adjustments, if necessary. In addition, the Design Assistant helps you check the reliability of a design based on a set of design rules.

Using the Messages Window to View Fitting Results

The **Processing** tab of the Messages window and the **Messages** section of the Report window or Report File display the messages generated from the most recent compilation or simulation. Figure 4 shows the Messages window.

Figure 4. Messages Window

Arrow buttons allow you to select next and previous messages

Location list allows you to select from multiple locations

Clicking the Locate button displays the selected location
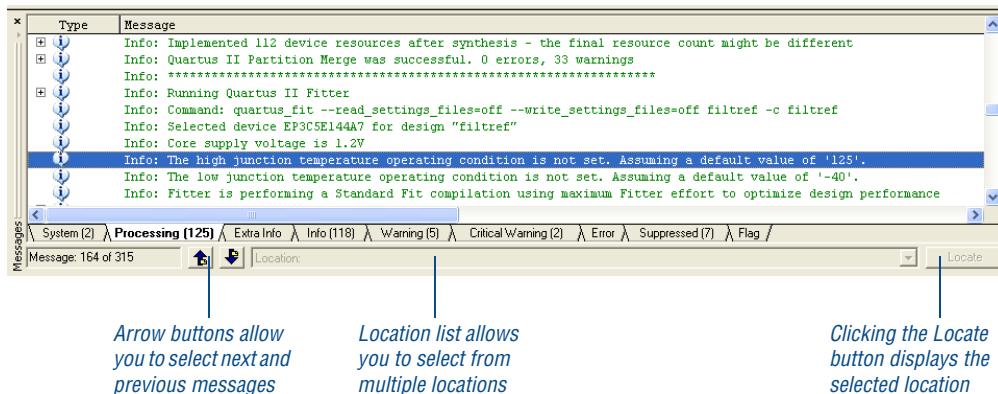
In the Messages window, you can right-click a message and click **Help** to get Help on a particular message.

By default, all message types are displayed in the **Processing** tab of the Messages window. If you want to filter the messages that appear in the Messages window, you can set options in the **Filtering** tab under **Messages** in the **Options** dialog box that control the display of warning messages, critical warning messages, information messages, and extra information messages. The **Colors** tab allows you to customize the colors for each type of message. The **Messages** tab of the **Options** dialog box allows you to specify options for displaying separate optional tabs that display the **Processing** tab's messages by type: **Extra Info**, **Info**, **Warning**, **Critical Warning**, and **Error**. Right-clicking messages in the Messages window also provides commands that allow you to filter messages and display optional message tabs.

For Information About	Refer To
Viewing messages	"About the Messages Window" and "Managing Messages in the Messages Window" in Quartus II Help
Locating the source of a message	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial
	"Locating the Source and Getting Help on Messages" in Quartus II Help

Using the Report Window or Report File to View Fitting Results

The Report window contains many sections that can help you analyze the placement and routing of your design. It includes several sections that show resource usage, and also lists error messages that were generated by the Fitter, as well as messages for any other module you were running.

By default, the Report window opens automatically when you run the Fitter or any other compilation or simulation module; however, you can specify that it should not open automatically by turning off **Automatically open the Report window before starting a processing task if the appropriate Tool window is not already open** in the **Processing** page of the **Options** dialog box. Also, if the Compiler Tool window is open, the Report window does not open automatically, but clicking the Report File icon for each module displays the report for that module. When the Fitter is processing the design, the Report window continuously updates with new information. If you stop the Fitter, the Report window contains only the information created prior to the point at which you stopped the Fitter.

The Quartus II software automatically generates text and HTML versions of reports, depending on which options you specify in the **Processing** page of the **Options** dialog box.

For Information About	Refer To
Report Window sections	"List of Compilation and Simulation Reports" in Quartus II Help
Using the Report Window	"Navigating the Report Window" in Quartus II Help
Viewing the compilation report	"Module 3: Compile a Design" in the Quartus II Interactive Tutorial

Using the Chip Planner to Analyze Results

After you run the Fitter, the Chip Planner displays the results of placement and routing. In addition, you can back-annotate the fitting results to preserve the resource assignments made during the last compilation. The Chip Planner allows you to view logic placement made by the Fitter and/or user assignments, make LogicLock region assignments, and view routing congestion (Figure 5).

Figure 5. Chip Planner
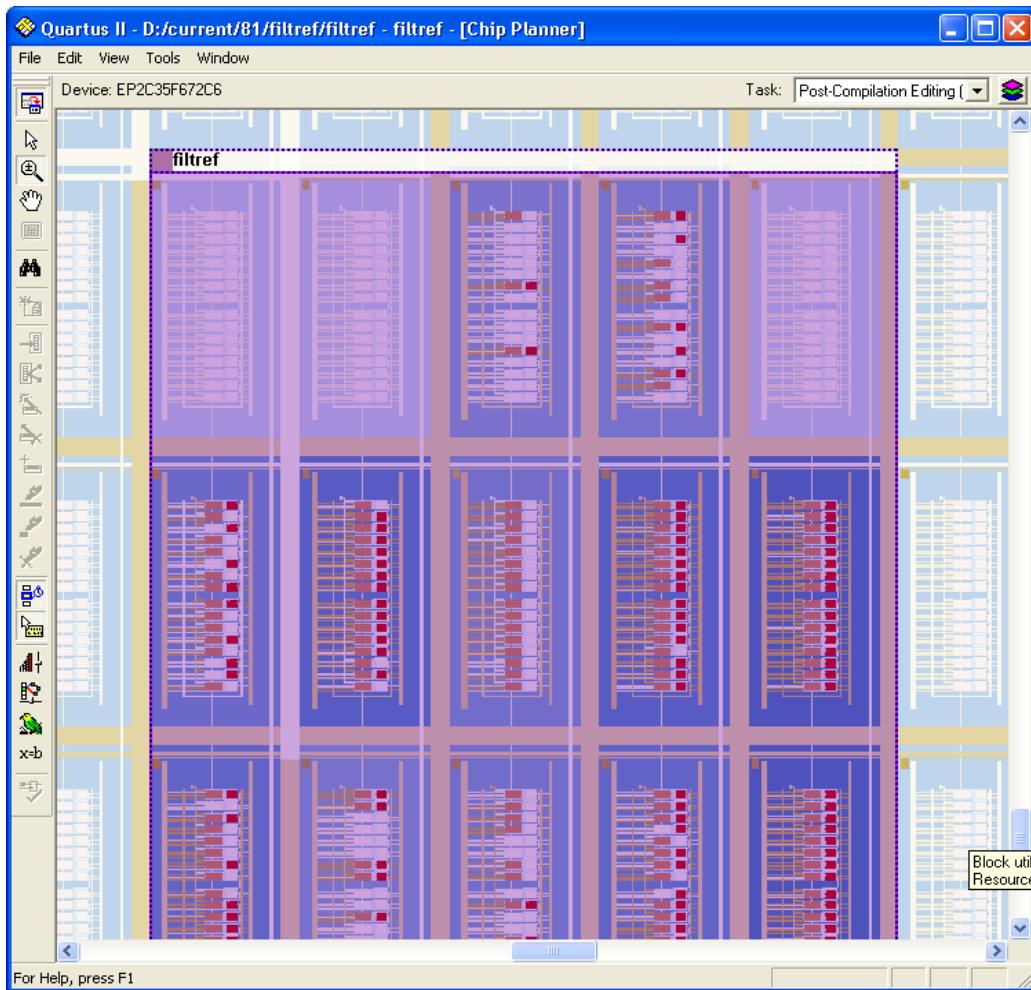
Resource usage in the Chip Planner is color coded. Different colors represent different resources, such as unassigned and assigned pins and logic cells, unrouted items, and row FastTrack® fan-outs. The Chip Planner also allows you to customize the floorplan view using filters to show pins and the interior structure of the device.

To edit assignments in the Chip Planner, you can click a resource assignment and drag it to a new location. You can use rubberbanding to display a visual representation of the number of routing resources affected by the move.

You can view the routing congestion in a design, view routing delay information for paths, and view connection counts to specific nodes. The Chip Planner also allows you to view the node fan-out and fan-in for specific structures, or view the paths between specific nodes. If necessary, you can change or delete resource assignments. For more information on using the Chip Planner, refer to "Using the Chip Planner" on page 139 in Chapter 10, "Timing Closure."

For Information About	Refer To
Viewing the fit in the Chip Planner	*Engineering Change Management with the Chip Planner* chapter in volume 3 of the *Quartus II Handbook*

Using the Design Assistant to Check Design Reliability

The Quartus II Design Assistant allows you to check the reliability of your design, based on a set of design rules, to determine whether there are any issues that may affect fitting or design optimization. The **Design Assistant** page of the **Settings** dialog box allows you to specify which design reliability guidelines to use when checking your design. For more information, refer to "Using the Design Assistant to Check Design Reliability" on page 74 in Chapter 5, "Synthesis."

Optimizing the Fit

Once you have run the Fitter and have analyzed the results, you can try several options to optimize the fit:

■ Using location assignments
■ Setting options that control place and route
■ Using the Resource Optimization Advisor
■ Using the Design Space Explorer

Using Location Assignments

You can assign logic to physical resources on the device, such as a pin, logic cell, or Logic Array Block (LAB), by using the Chip Planner or the Assignment Editor in order to control place and route. You may want to use the Chip Planner to edit assignments because it gives you a graphical view of the device and its features. If you want to create several new location assignments at once, on the Assignments menu, click **Assignment Editor**. In addition to using the Chip Planner or Assignment Editor to create assignments, you can also use Tcl commands. If you want to specify global assignments for the project, you can use the **Settings** dialog box. For more information about specifying initial design constraints, refer to "Chapter 4: Constraint Entry" on page 51.

Setting Options that Control Place & Route

You can set several options that control the Fitter and may affect place and route:

- Fitter options
- Fitting optimization and physical synthesis options
- Individual and global logic options that affect fitting

Setting Fitter Options

The **Fitter Settings** page of the **Settings** dialog box allows you to specify options that control timing-driven compilation and compilation speed. You can specify whether the Fitter should try to use registers in I/O cells (rather than registers in regular logic cells) to meet timing requirements and assignments that relate to I/O pins. You can direct the Fitter to consider only slow-corner timing delays when optimizing the design, or to consider fast-corner timing delays as well as slow-corner timing delays when optimizing the design to meet timing requirements at both corners. You can specify whether you want the Fitter to use standard fitting, which works hardest to meet your f_{MAX} timing requirements, to use the fast fit feature, which improves the compilation speed but may reduce the f_{MAX}, or to use the auto fit feature, which reduces Fitter effort after meeting timing requirements and may decrease compilation time. The **Fitter Settings** page also allows to you specify that you want to limit Fitter effort to only one attempt, which may also reduce the f_{MAX}.

Setting Physical Synthesis Optimization Options

The Quartus II software allows you to set options for performing physical synthesis to optimize the netlist during fitting. You specify physical synthesis optimization options in the **Physical Synthesis Optimizations** page under **Compilation Process Settings** page in the **Settings** dialog box. For more information about physical synthesis options, refer to "Using Netlist Optimizations to Achieve Timing Closure" on page 142 in Chapter 10, "Timing Closure."

For Information About	Refer To
Using Quartus II physical synthesis optimizations	*Netlist Optimizations & Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*
Using Quartus II Fitter optimization options	"About Synthesis" in Quartus II Help

Setting Individual Logic Options that Affect Fitting

Quartus II logic options allow you to set attributes without editing the source code. You can specify Quartus II logic options for individual nodes and entities in the Assignment Editor and can specify global default logic options in the **More Fitter Settings** dialog box, which is available by clicking **More Settings** in the **Fitter Settings** page of the **Settings** dialog box. For example, you can use logic options to specify that the signal should be available throughout the device on a global routing path, specify that the Fitter should create parallel expander chains automatically, specify that the Fitter should automatically combine a register with a combinational function in the same logic cell, also known as "register packing," or limit the length of carry chains, cascade chains, and parallel expander chains.

For Information About	Refer To
Using Quartus II logic options to control place and route	"Logic Options" and "Working with Assignments in the Assignment Editor" in Quartus II Help

# Using the Resource Optimization Advisor
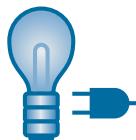
The Resource Optimization Advisor offers recommendations for optimizing your design for resource usage in the following areas:

- Logic elements
- Memory blocks
- DSP blocks
- I/O elements
- Routing resources

If you have an open project, you can view the Resource Optimization Advisor by clicking **Resource Optimization Advisor** on the Tools menu. If the project has not been compiled yet, the Resource Optimization Advisor provides only general recommendations for optimizing resource usage. If the project has been compiled, however, the Resource Optimization Advisor can provide specific recommendations for the project, based on the project information and current settings. Figure 6 shows the Resource Optimization Advisor.

Figure 6. Resource Optimization Advisor Summary Page
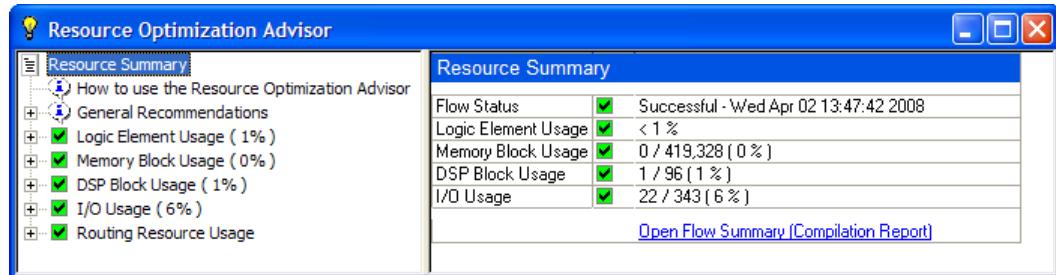
The first page of the Resource Optimization Advisor summarizes the resource usage after compilation, and indicates possible problem areas. The left pane of the Resource Optimization Advisor shows a hierarchical list of problems and recommendations, with icons that indicate whether the recommendation might be appropriate for the current design and target device family, or whether the current design already has the recommended setting. When you click a recommendation in the hierarchical list, the right pane provides a detailed description of the recommendation, a summary, the current global settings, and one or more recommended actions, as shown in Figure 7.

Figure 7. Resource Optimization Advisor Recommendation Page
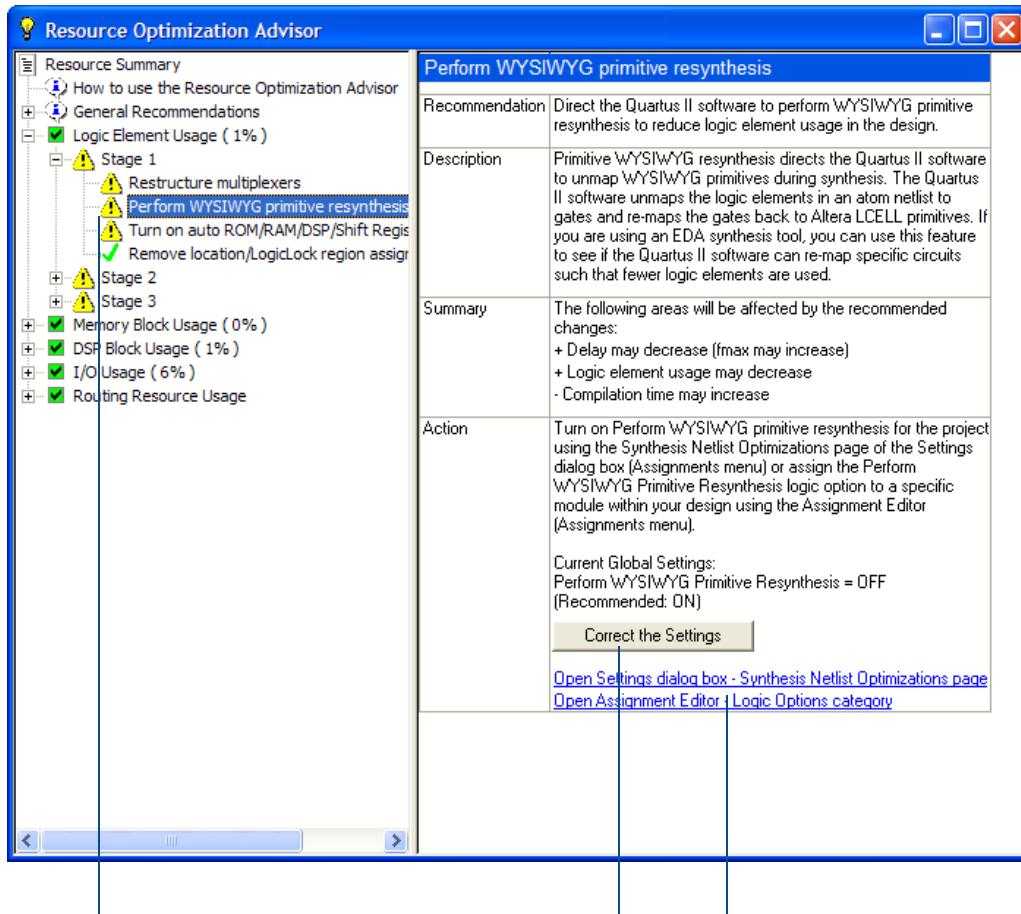
Hierarchical list of recommendations—icons indicate potential problem areas

Some recommendations include buttons that cause the reccomended changes to be made in your design.

Clicking a link in the recommendations page opens the appropriate dialog box, page, or feature.

If the recommended action involves changing a Quartus II setting, the right pane of the Resource Optimization Advisor may include a link to the appropriate dialog box, page, or feature in the Quartus II software or may include a button that provides more information about the design. It may also include links to Quartus II Help or other documentation on the Altera website.

If you want to view recommendations for improving timing results, you can use the Timing Optimization Advisor. See "Using the Timing Optimization Advisor" on page 141 in Chapter 10, "Timing Closure."

Using the Design Space Explorer

Another way to control Quartus II fitting to optimize for power, area, and performance, is to use the Design Space Explorer (DSE). The DSE interface allows you to explore a range of Quartus II options and settings automatically to determine which settings you should use to obtain the best possible result for the project. To start DSE, on the Tools menu, click **Launch Design Space Explorer**.

You can specify the effort level that DSE puts into determining the optimal settings the current project. The DSE interface also allows you to specify optimization goals and allowable compilation time.

DSE provides several exploration modes, which are listed under **Exploration Settings** in the DSE window. Selecting the **Advanced Search** option opens the **Advanced** tab, which allows you to specify additional options for exploration space, optimization goal, and search method.

After you have specified your exploration settings, you can use the **Explore Space** command on the Processing menu to start the exploration. You can see the results of the exploration on the **Explore** tab.

☞ Running the Design Space Explorer

You can run DSE in graphical user interface mode by typing the following command at a command prompt:

```
quartus_sh --dse ↵
```

You can run DSE in command-line mode by typing the following command at a command prompt, along with any additional DSE options:

```
quartus_sh --dse -nogui <project name> [-c <revision name>] ↵
```

For help on DSE options, type `quartus_sh --help=dse` ↵ at command prompt, or, on the help menu, click **Show Documentation**.

Many of the exploration space modes allow you to specify the degree of effort you want DSE to spend in fitting the design; however, increasing the effort level usually increases the compilation time. Custom exploration mode allows you to specify various parameters, options, and modes and then explore their effects on your design.

The Signature modes allow you to explore the effect of a single parameter on your design and its trade-offs for f_{MAX}, slack, compile time, and area. In the Signature modes, DSE tests the effects of a single parameter over multiple seeds, and then reports the average of the values so you can evaluate how that parameter interacts in the space of your design.

DSE also provides a list of **Optimization Goal** options, which allow you to specify whether DSE should optimize for area, speed, or for negative slack and failing paths.

In addition, you can specify **Search Method** options, which provide additional control over how much time and effort DSE should spend on the search.

After you have completed a design exploration with DSE, you can create a new revision from a DSE point. You can then close DSE and open the project with the new revision from within the Quartus II software.

For Information About	Refer To
Using the Design Space Explorer	*Design Space Explorer* chapter in volume 2 of the *Quartus II Handbook*
Parameters and settings for optimizing performance	*Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*

Chapter Seven

Block-Based Design
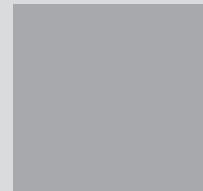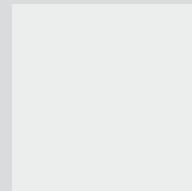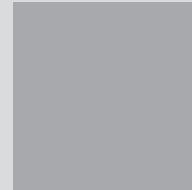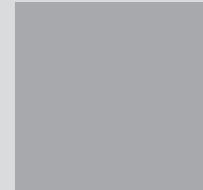
7

Introduction

The Quartus II Incremental Compilation feature and LogicLock regions feature enable a block-based design flow by allowing you to create modular designs, designing and optimizing each module separately before incorporating it into the top-level design. Incorporating each module into the top-level design does not affect the performance of the lower level modules, as long as each module has registered inputs and outputs.

LogicLock regions are flexible, reusable constraints that increase your ability to guide logic placement on the target device. You can define any region of physical resources on the target device as a LogicLock region. Assigning nodes or entities to a LogicLock region directs the Fitter to place those nodes or entities inside the region during fitting.

LogicLock regions support team-oriented, block-based design by enabling you to optimize logic blocks individually, and then import them and their placement constraints into a larger design. The LogicLock methodology also promotes module reuse. You can develop modules separately, then constrain the modules to LogicLock regions for use in other designs with no loss in performance.

The LogicLock feature also allows you to assign design partitions to physical locations in the device as part of a top-down or bottom-up incremental compilation flow.

Block-Based Design Flow

In traditional top-down design flows, there is only one netlist for the design. In a top-down design flow, individual modules of the design can have different performance from the overall design when implemented on their own. In bottom-up block-based design flows, there are separate netlists for each module. You can optimize each module independently and then incorporate it into the top-level design as a block. You can use block-based design in the following flows:

■ **Modular design flow**: In the modular design flow, you divide a design into a top-level design that instantiates separate submodules. You can develop each module separately and then incorporate each module into the top-level design. Placement is determined manually by you or automatically by the Quartus II software.

■ **Incremental compilation flow**: In the incremental compilation flow, you create and optimize a system, and then add future modules with little or no effect on the performance of the original system.

■ **Team-based design flow**: In the team-based design flow, you partition a design into separate modules, and instantiate and connect the modules in a top-level design. Other team members then separately develop the lower-level modules, creating separate projects for each module, while using the assignments developed for the top-level design. Once the lower-level modules are complete, they are imported into the top-level design, which then undergoes final compilation and verification.

In all three design flows, you can preserve performance at all levels of development by partitioning designs into functional blocks, organized according to the physical structure of the circuit or by critical paths.

Using LogicLock Regions

A LogicLock region is defined by its size and location on the device. You can specify the size and location of a region, or direct the Quartus II software to create them automatically. Table 1 lists the major properties of LogicLock regions that you can specify in the Quartus II software.

Table 1. LogicLock Region Properties (Part 1 of 2)

Property	Values	Behavior
State	Floating or Locked	Floating regions allow the Quartus II software to determine the region's location on the device. Locked regions have a user-defined location. Locked regions are shown in the floorplan with a solid boundary and floating regions are shown with a dashed boundary in the floorplan. A locked region must have a fixed size.
Size	Auto or Fixed	Auto-sized regions allow the Quartus II software to determine the appropriate size of a region given its contents. Fixed regions have a user-defined shape and size.

*Table 1. LogicLock Region Properties (Part 2 of 2)*

Property	Values	Behavior
Reserved	On, Limited, or Off	The reserved property allows you to define whether the Quartus II software can use the resources within a region for entities that are not assigned to the region. If the reserved property is on, only items assigned to the region can be placed within its boundaries. If this property is set to **Limited**, nodes from other entities may be placed in the region if the nodes are not assigned to a parent region.
Soft	On or Off	Soft regions give more deference to timing requirements, and allow some entities to leave a region if it improves the performance of the overall design. Hard regions do not allow the Quartus II software to place contents outside the boundaries of the region.
Origin	Any Floorplan Location	The origin defines the placement of the LogicLock region in the floorplan.

With the LogicLock design flow, you can define a hierarchy for a group of regions by declaring parent and child regions. The Quartus II software places child regions completely within the boundaries of a parent region. You can lock a child module relative to its parent region without constraining the parent region to a locked location on the device.

You can create and modify LogicLock regions by using the Chip Planner, the **LogicLock Regions Window** command on the **Assignments** menu, the **Hierarchy** tab of the Project Navigator, or by using Tcl scripts. All LogicLock attributes and constraint information (clock settings, pin assignments, and relative placement information) are stored in the Quartus II Settings File for the project.

After you create a LogicLock region, you can use the LogicLock Regions window to view all of the LogicLock regions in your design, including size, state, width, height, and origin. You can also edit and add new LogicLock regions. (Figure 1).

Figure 1. LogicLock Regions Window
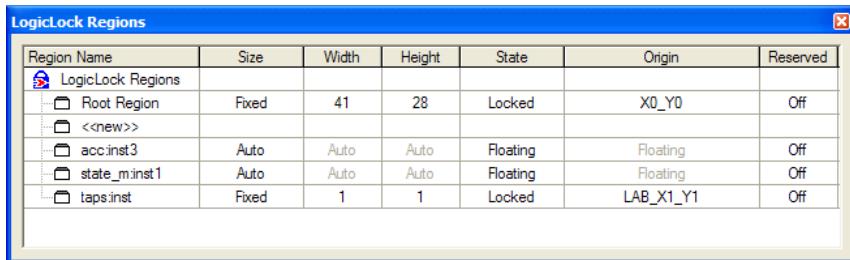
Region Name	Size	Width	Height	State	Origin	Reserved
⊛ LogicLock Regions						
🗀 Root Region	Fixed	41	28	Locked	X0_Y0	Off
🗀 <<new>>						
🗀 acc:inst3	Auto	Auto	Auto	Floating	Floating	Off
🗀 state_m:inst1	Auto	Auto	Auto	Floating	Floating	Off
🗀 taps:inst	Fixed	1	1	Locked	LAB_X1_Y1	Off

You can also use the **LogicLock Regions Properties** dialog box to edit existing LogicLock regions, view information on the LogicLock regions in the design, and determine which regions contain illegal assignments.

In addition, you can add path-based assignments (based on source and destination nodes), wildcard assignments, and Fitter priority for path-based and wildcard assignments to LogicLock regions. Setting the priority allows you to specify the order in which the Quartus II software resolves conflicting path-based and wildcard assignments. You can open the **Priority** dialog box from the **LogicLock Region Properties** dialog box. (Figure 2).

Figure 2. LogicLock Region Properties Dialog Box
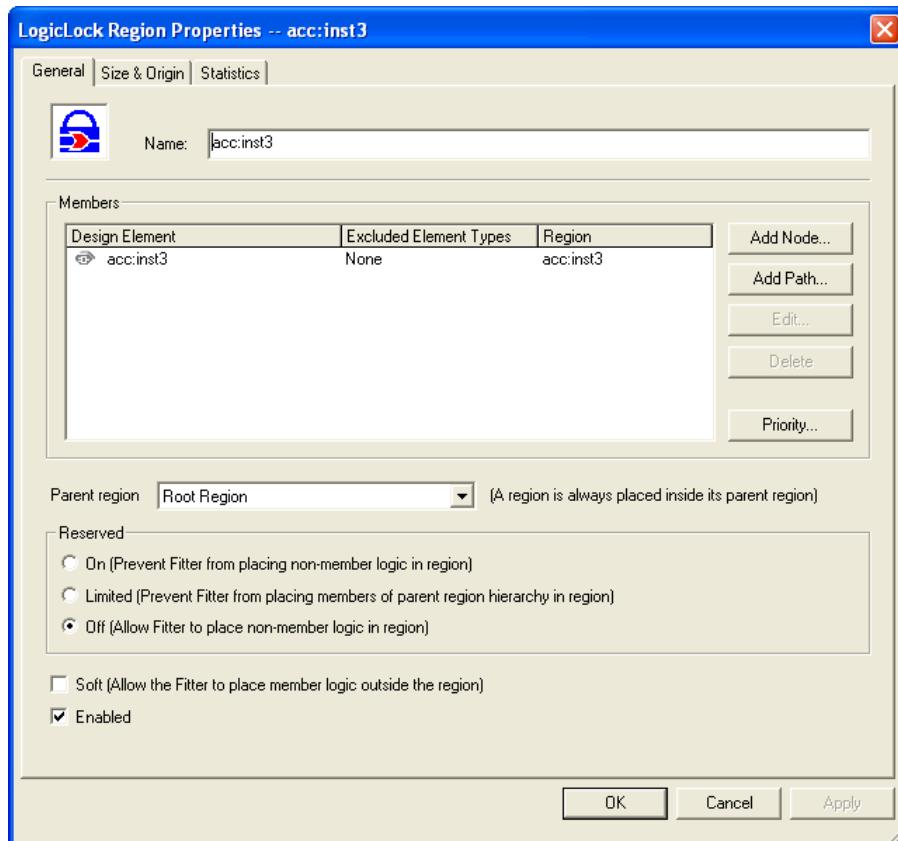
After you perform analysis and elaboration or a full compilation, the Quartus II software displays the hierarchy of the design in the **Hierarchy** tab of the Project Navigator. You can click any of the design entities in this view and create new LogicLock regions from them, or drag them into an existing LogicLock region in the Timing Closure Floorplan.

Altera also provides LogicLock Tcl commands to assign LogicLock region content at the command line or in the Quartus II Tcl Console window. You can use the provided Tcl commands to create floating and auto-size LogicLock regions, add a node or a hierarchy to a region, preserve the hierarchy boundary, back-annotate placement results, import and export regions, and save intermediate synthesis results.

For Information About	Refer To
Using LogicLock with the Quartus II software	*Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*
	"About LogicLock Regions" in Quartus II Help

Using LogicLock Regions in Top-Down Incremental Compilation Flows

If you are planning to perform a full incremental compilation, it is important to assign design partitions to physical locations on the device. You can assign design partitions to LogicLock regions by dragging a design partition from the **Hierarchy** tab of the Project Navigator window, the Design Partitions window, or the Node Finder and dropping it directly in the LogicLock Regions window or to a LogicLock region in the Chip Planner.

Altera recommends that you create one LogicLock region for each partition in your design. You can achieve the best performance when these regions are all fixed-size, fixed-location regions. Ideally, you should assign the LogicLock regions manually to specific physical locations in the device by using the Chip Planner; however, you can also allow the Quartus II software to assign LogicLock regions to physical locations somewhat automatically by setting the LogicLock region **Size** option to **Auto** and the **State** properties to **Floating**. After the initial compilation, you should back-annotate the LogicLock region properties (not the nodes) to ensure that all the LogicLock regions have a fixed size and a fixed location. This process creates initial floorplan assignments that can be modified more easily, as needed.

After the initial or setup compilation, Altera recommends that you set the **Size** to **Fixed** in order to yield better f_{MAX} results. If device utilization is low, increasing the size of the LogicLock region may allow the Fitter additional flexibility in placement and may produce better final results.

When you perform an incremental compilation, the fitting and synthesis results and settings for design partitions are saved in the project database.

For more information about assigning design partitions, refer to "Creating Design Partitions" on page 57 in Chapter 4, "Constraint Entry."For more information about incremental compilation, refer to "Top-Down Incremental Compilation Flow" on page 12 in Chapter 1, "Design Flow" and "Using Incremental Compilation" on page 86 in Chapter 6, "Place and Route".

For Information About	Refer To
Using Quartus II incremental compilation with LogicLock regions	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	"Module 7: Incremental Compilation" in the Quartus II Interactive Tutorial
	"About Incremental Compilation" in Quartus II Help

Exporting & Importing Partitions for Bottom-Up Design Flows

The bottom-up flow refers to the design methodology in which a project is first divided into smaller sub-designs that are implemented as separate projects, potentially by different designers. The compilation results of these lower-level projects are then exported and given to the designer (or the project lead) who is responsible for importing them into the top-level project to obtain a fully functional design. The bottom-up flow facilitates team-based development and permits the reuse of compilation results from another project, with the ultimate goals of performance preservation and compilation time reduction.

Preparing the Top-Level Design

To set up your design for bottom-up incremental compilation, use the following general steps:

1. Create a top-level project. The top-level design file must include the top-level entity that instantiates all the lower-level subdesigns that you plan to compile in separate Quartus II projects and import as separate design partitions.

2. In your top-level project, include a wrapper design file for each subdesign partition that defines at least the port interface of the subdesign.

3. Create all global assignments, including the device assignment, pin location assignments, and timing assignments, so that the final design meets its requirements.

4. Set up the top-level design with design partitions, turn on incremental compilation, and create a design floorplan using LogicLock assignments.

Importing a Lower-Level Partition Into the Top-Level Project

When your subdesign partition is ready to be incorporated into the top-level design, in the subdesign project, on the Project menu, click **Export Design Partition**. In the **Quartus II Export Partition File** box of the **Export Project as Design Partition** dialog box, type the name of the Quartus II Exported Partition File (**.qxp**). By default, the directory path and file name are the same as the current project.

Under **Netlist to export**, select either **Post-fit netlist** or **Post-synthesis netlist**, and then click **Export**. The Quartus II software creates the Quartus II Exported Partition File in the specified directory.

You must import the design netlist from the Quartus II Exported Partition File and add it to the database for the top-level project. Importing filters the assignments from the subdesign and creates the appropriate assignments in the top-level project.

For Information About	Refer To
Importing and exporting designs as Quartus II Exported Partition Files and back-annotating assignments	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" and "Using the Team-Based Bottom-Up Design Flow" in Quartus II Help

Chapter
Eight

Simulation

8

Introduction

You can perform functional and timing simulation of your design by using EDA simulation tools or the Quartus II Simulator. The Quartus II software provides the following features for performing simulation of designs in EDA simulation tools:

- NativeLink integration with EDA simulation tools
- Generation of output netlist files
- Functional and timing simulation libraries
- Generation of test bench template and Memory Initialization Files (**.mif**)
- Generation of Signal Activity Files (**.saf**) for power analysis

Figure 1 shows the simulation flow with EDA simulation tools and the Quartus II Simulator.

Figure 1. Simulation Flow
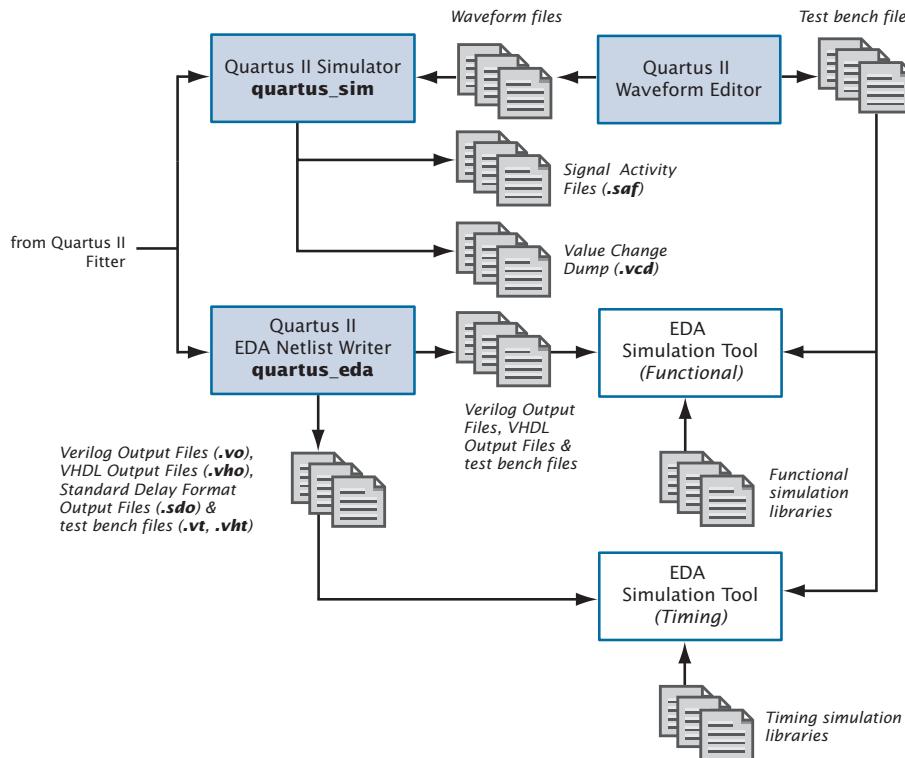
Simulating with EDA Tools

The EDA Netlist Writer module of the Quartus II software generates VHDL Output Files (**.vho**) and Verilog Output Files (**.vo**) for performing functional or timing simulation, and Standard Delay Format Output Files (**.sdo**) that are required for performing timing simulation with EDA simulation tools. The Quartus II software generates SDF Output Files in Standard Delay Format version 2.1. The EDA Netlist Writer places simulation output files in a tool-specific directory under the current project directory.

In addition, the Quartus II software offers seamless integration for timing simulation with EDA simulation tools through the NativeLink feature. The NativeLink feature allows the Quartus II software to pass information to EDA simulation tools, and includes the ability to launch EDA simulation tools from within the Quartus II software.

Table 1 lists the EDA simulation tools that are supported by the Quartus II software and indicates which tools support the NativeLink feature.

Table 1. Quartus II–Supported EDA Simulation Tools

Simulation Tool Name	NativeLink Support
Cadence NC-Verilog	✓
Cadence NC-VHDL	✓
Mentor Graphics ModelSim	✓
Mentor Graphics ModelSim-Altera	✓
Active-HDL	✓
Synopsys VCS MX	✓
Synopsys VCS	✓

☞ The ModelSim-Altera Software

The Mentor Graphics ModelSim-Altera software is included in Altera design software subscriptions for functional simulation and HDL test bench support.

Specifying EDA Simulation Tool Settings

You can select an EDA simulation tool in the **New Project Wizard** on the File menu when you create a new project, or in the **Simulation** page under **EDA Tool Settings** in the **Settings** dialog box on the Assignments menu. The **Simulation** page allows you to select a simulation tool and specify options for the generation of Verilog and VHDL output files and the corresponding SDF Output File, and, for power analysis, a Signal Activity File. Figure 2 shows the **Simulation** page.

Figure 2. Simulation Page in Settings Dialog Box
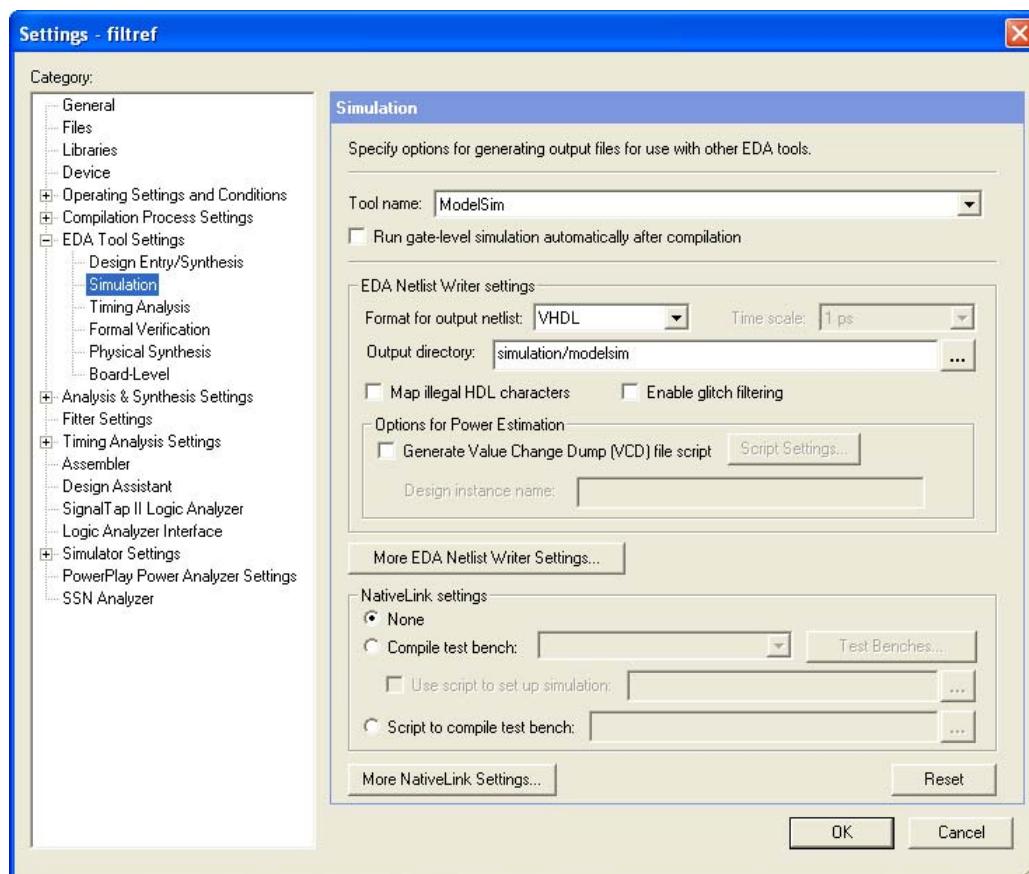
Generating Simulation Output Files

You can run the EDA Netlist Writer module to generate Verilog Output Files and VHDL Output Files by specifying EDA tool settings and compiling the design. If you have already compiled a design in the Quartus II software, you can specify different simulation output settings in the Quartus II software (for example, a different simulation tool) and then regenerate the Verilog Output Files or VHDL Output Files by clicking **Start EDA Netlist Writer** on the Processing menu. If you are using the NativeLink feature, you can also run a simulation after an initial compilation with the **Run EDA Simulation Tool** command on the Tools menu.

☞ **Using the quartus_eda executable**

You can also run the EDA Netlist Writer separately at the command prompt or in a script by using the **quartus_eda** executable.

The **quartus_eda** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_eda** executable, type one of the following commands at the command prompt:

```
quartus_eda -h ↵
quartus_eda --help ↵
quartus_eda --help=<topic name> ↵
```

The Quartus II software also allows you to generate the following types of output files for use in performing functional and timing simulation in EDA simulation tools:

■ **Test Bench Files:** You can create Verilog Test Bench Files (**.vt**) and VHDL Test Bench Files (**.vht**) for use with EDA simulation tools from a Vector Waveform File (**.vwf**) in the Quartus II Waveform Editor, using the **Export** command on the File menu. Verilog HDL and VHDL Test Bench Files are test bench template files that contain an instantiation of the top-level design file and test vectors from the Vector Waveform File. You can also generate self-checking test bench files if you specify the expected values in the Vector Waveform File.

■ **Memory Initialization Files:** You can use the Quartus II Memory Editor to enter the initial contents of a memory block, for example, content-addressable memory (CAM), RAM, or ROM, in a Memory Initialization File (**.mif**) or a Hexadecimal (Intel-Format) File (**.hex**).

■ **Signal Activity Files:** You can create Signal Activity Files for use with the PowerPlay Power Analyzer. A Signal Activity File contains toggle rate and static probability data for a design. You can specify a limit for the signal activity period, and can also specify that glitch filtering can be performed.

EDA Simulation Flow

Using the NativeLink feature, you can direct the Quartus II software to compile a design, generate the appropriate output files, and then automatically perform the simulation using EDA simulation tools. Alternatively, you can run EDA simulation tools manually before compilation (functional simulation) or after compilation (timing simulation) in the Quartus II software.

EDA Tool Functional Simulation Flow

You can perform a functional simulation at any point in your design flow. The following steps describe the basic flow needed to perform a functional simulation of a design using an EDA simulation tool. Refer to Quartus II Help for more information on specific EDA simulation tools. To perform a functional simulation using EDA simulation tools:

1. If you have not already done so, set up the project in the EDA simulation tool.

2. Create a working library.

3. Compile the appropriate functional simulation libraries with the EDA simulation tool.

4. Compile the design files and test bench files with the EDA simulation tool.

5. Perform the simulation with the EDA simulation tool.

NativeLink Simulation Flow

You can use the NativeLink feature to perform the steps to set up and run an EDA simulation tool automatically from within the Quartus II software. The following steps describe the basic flow for using EDA simulation tools with the NativeLink feature:

1. Specify EDA tool settings in the Quartus II software, either in the **Settings** dialog box on the Assignments menu, or during project setup, with the **New Project Wizard** on the File menu.

2. Turn on **Run this tool automatically after compilation** when specifying EDA tool settings.

3. On the **EDA Tool Options** page of the **Options** dialog box, double-click the Location of Executable column for the appropriate tool and specify the correct path.

4. On the **Simulation** page under **EDA Tool Settings** on the **Settings** dialog box, under NativeLink settings, specify your testbench file.

5. Compile the design in the Quartus II software. The Quartus II software performs the compilation, generates the Verilog HDL or VHDL output files and corresponding SDF Output Files (if you are performing a timing simulation), and launches the simulation tool. The Quartus II software directs the simulation tool to create a working library; compile or map to the appropriate libraries, design files, and test bench files; set up the simulation environment; and run the simulation.

Manual Timing Simulation Flow

If you want more control over the simulation, you can generate the Verilog HDL or VHDL output files and corresponding SDF Output File in the Quartus II software, and then manually launch the simulation tool to perform the simulation. The following steps describe the basic flow needed to perform a timing simulation of a Quartus II design using an EDA simulation tool. Refer to Quartus II Help for more information on specific EDA simulation tools.

1. Specify EDA tool settings in the Quartus II software, either in the **Settings** dialog box on the Assignments menu, or during project setup, with the **New Project Wizard** on the File menu.

2. Compile the design in the Quartus II software to generate the output netlist files. The Quartus II software places the files in a tool-specific directory.

3. Launch the EDA simulation tool.

4. Set up the project and a working directory with the EDA simulation tool.

5. Compile or map to the timing simulation libraries, and compile the design and test bench files with the EDA simulation tool.

6. Perform the simulation with the EDA simulation tool.

Simulation Libraries

Altera provides functional simulation libraries for designs that contain Altera-specific components, and atom-based timing simulation libraries for designs compiled in the Quartus II software. You can use these libraries to perform functional or timing simulation of any design with Altera-specific components in EDA simulation tools that are supported by the Quartus II software. Additionally, Altera provides pre-compiled functional and timing simulation libraries for simulation in the ModelSim-Altera software.

Altera provides functional simulation libraries for designs that use Altera megafunctions and standard library of parameterized modules (LPM) functions. Altera also provides precompiled versions of the **altera_mf** and **220model** libraries for simulation in the ModelSim software.

In the Quartus II software, the information for specific device architecture entities and megafunctions is located in post-routing atom-based timing simulation libraries. The timing simulation library files differ based on device family and whether you are using Verilog Output Files or VHDL Output Files. For VHDL designs, Altera provides VHDL Component Declaration files for designs with Altera-specific megafunctions.

For Information About	Refer To
Functional Simulation libraries included with the Quartus II software	"Altera Functional Simulation Libraries" in Quartus II Help
Performing simulation using the ModelSim or ModelSim-Altera software	*Mentor Graphics ModelSim Support* chapter in volume 3 of the *Quartus II Handbook*
Performing simulation with the VCS software	*Synopsys VCS and VCS-MX Support* chapter in volume 3 of the *Quartus II Handbook*
Performing simulation with the NC-Sim software	*Cadence NC-Sim Support* chapter in volume 3 of the *Quartus II Handbook*
Performing simulation with the Aldec Active-HDL software	*Aldec Active HDL Support* chapter in volume 3 of the *Quartus II Handbook*
Performing simulation of Altera IP with EDA tools	*Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*

Using the Quartus II Simulator

You can use the Quartus II Simulator to simulate any design. Depending on the type of information you need, you can perform a functional simulation to test the logical operation of your design, a timing simulation to test both the logical operation and the worst-case timing for the design in the target device, or a timing simulation using the Fast Timing model to simulate the fastest possible conditions with the fastest device speed grade.

The Quartus II software allows you to simulate an entire design, or to simulate any part of a design. You can designate any design entity in a project as the top-level design entity and simulate the top-level entity and all of its subordinate design entities.

You can specify the type of simulation that should be performed, the time period covered by the simulation, the source of vector stimuli, and other simulation options in the **Simulator Settings** page of the **Settings** dialog box on the Assignments menu or in the Simulator Tool window on the Tools menu. Figure 3 shows the **Simulator Settings** page.

Figure 3. Simulator Settings Page in Settings Dialog Box
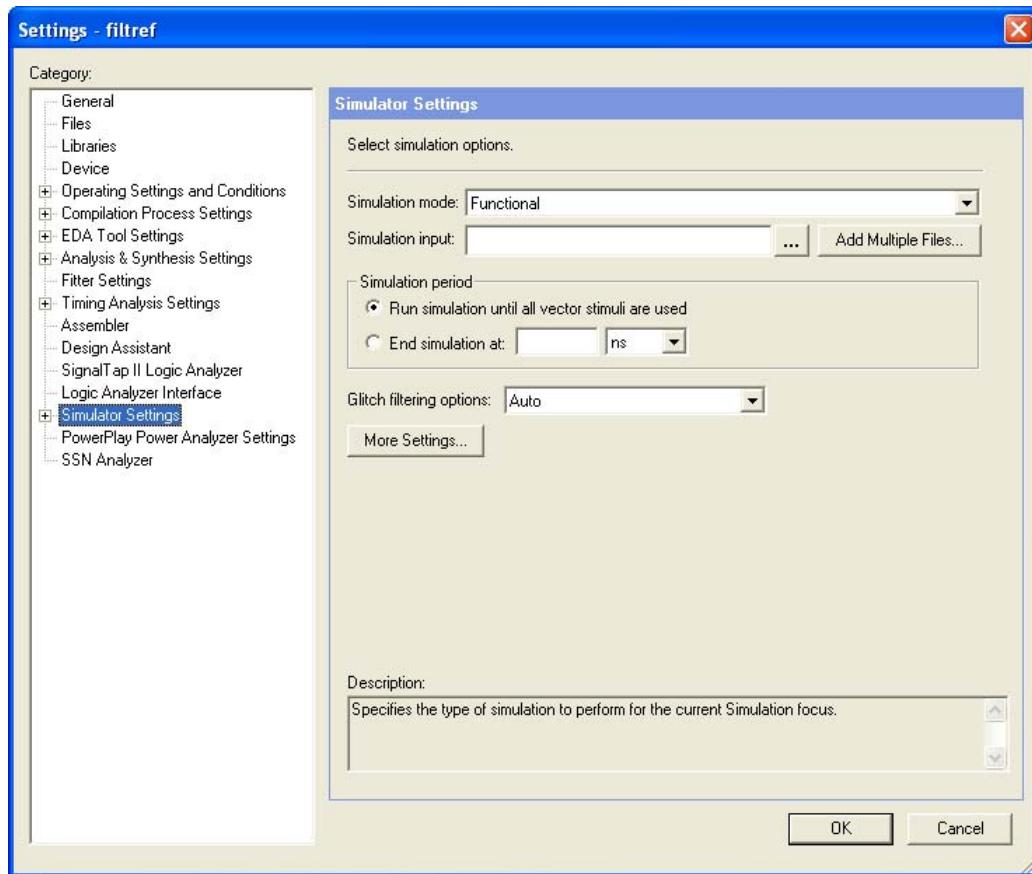
Before starting a simulation, you must generate the appropriate simulation netlist by either compiling the design for timing simulation or clicking **Generate Functional Simulation Netlist** on the Processing menu for functional simulation. In addition, you must create and specify a vector source file as the source of simulation input vectors. The Simulator uses the input vectors contained in the vector source file to simulate the output signals that a programmed device would produce under the same conditions.

The following steps describe the basic flow for performing either a functional or timing simulation in the Quartus II software:

1.   Specify Simulator settings.

2. If you are performing a functional simulation, click **Generate Functional Simulation Netlist** on the Processing menu. If you are performing a timing simulation, compile the design.

3. Create and specify a vector source file.

4. To run the simulation, point to **Start** on the Processing menu and click **Start Simulation**.

The Status window shows the progress of a simulation and the processing time. The Summary report in the Report window shows the simulation results.

☞ **Using the quartus_sim executable**

You can also run the Simulator separately at the command prompt or in a script by using the **quartus_sim** executable.

The **quartus_sim** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_sim** executable, type one of the following commands at the command prompt:

```
quartus_sim -h ↵
quartus_sim --help ↵
quartus_sim --help=<topic name> ↵
```

Creating Waveform Files

The Quartus II Waveform Editor allows you to create and edit input vectors for simulation in waveform or text format. Using the Waveform Editor, you can add input vectors to the waveform file that describe the behavior of the logic in your design (Figure 4).

Figure 4. The Quartus II Waveform Editor
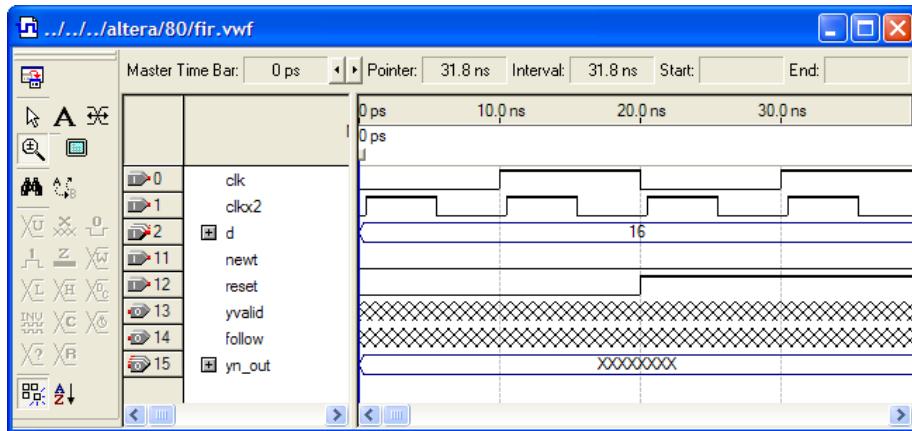
The Quartus II software supports waveform files in the Vector Waveform File (**.vwf**), Vector Table Output File (**.tbl**), Vector File (**.vec**), and Simulator Channel File (**.scf**) formats. You cannot edit a Simulator Channel File or Vector File in the Waveform Editor, but you can save it as a Vector Waveform File.

Using the Simulator Tool

You can also use the **Simulator Tool** command on the Tools menu to set Simulator settings, start or stop the Simulator, and open the simulation waveform for the current project, and generate a Value Change Dump (**.vcd**) file. To perform a simulation, you must first generate a simulation netlist with the **Generate Functional Simulation Netlist** button in the Simulator Tool for functional simulation or by compiling the design if you are performing a timing simulation. Figure 5 shows the Simulator Tool window.

Figure 5. Simulator Tool Window
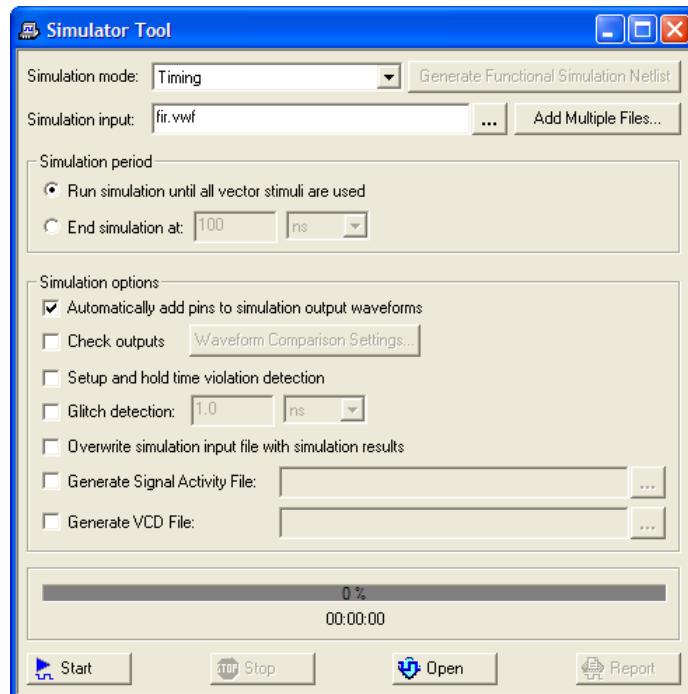
	For Information About	Refer To
	The Quartus II Simulator	*Quartus II Simulator* chapter in volume 3 of the *Quartus II Handbook*

Chapter
Nine

Timing Analysis
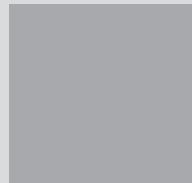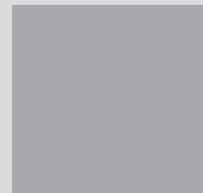
9

Introduction

The Quartus II TimeQuest Timing Analyzer allows you to analyze the performance of all logic in your design and help to guide the Fitter to meet timing requirements. The TimeQuest analyzer uses industry-standard Synopsys Design Constraint (SDC) methodology for constraining designs and reporting results. You can use the information generated by the timing analyzer to analyze, debug, and validate the timing performance of your design.

Running the TimeQuest Timing Analyzer

You can specify the TimeQuest analyzer as the default timing analyzer in the **Timing Analysis Settings** page of the **Settings** dialog box.

The TimeQuest analyzer provides an intuitive and easy-to-use GUI that allows you to constrain and analyze designs efficiently. The GUI is divided into the following four panes:

■ View pane
■ Tasks pane
■ Console
■ Report pane

Each pane provides features that enhance the productivity of performing static timing analysis in the TimeQuest analyzer (Figure 1).

Figure 1. TimeQuest Timing Analyzer Window
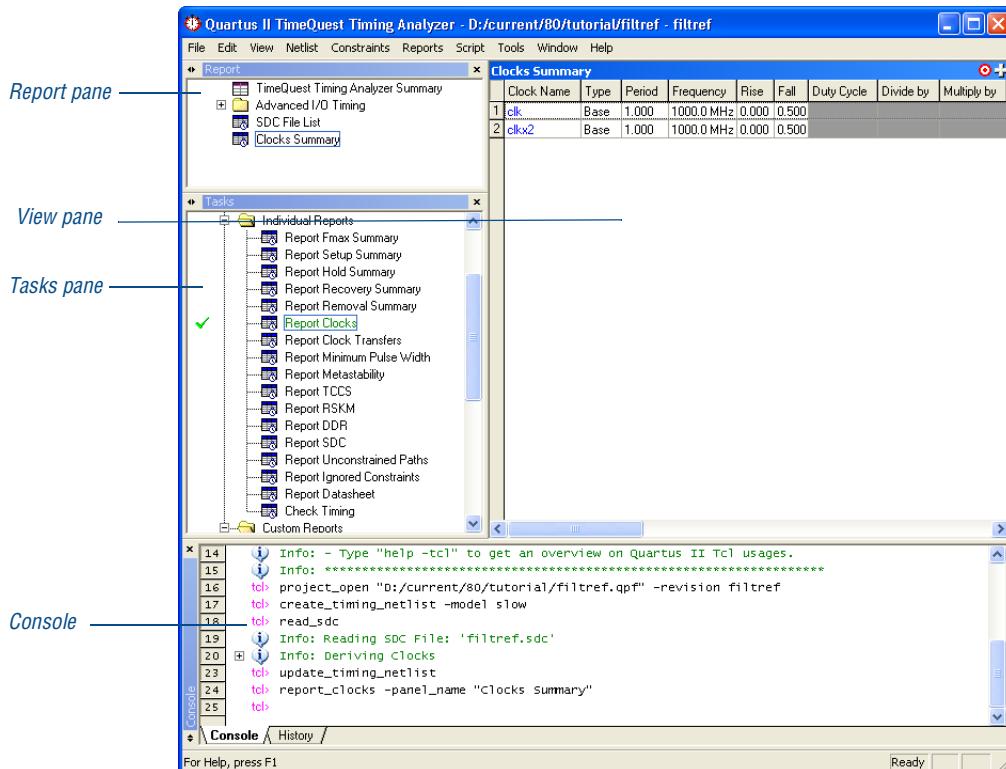
Tasks Pane

The **Tasks** pane provides easy access to commonly performed tasks such as netlist and report generation. Each command found in the **Tasks** pane has an equivalent Tcl command, which you can specify and view results from in the Console.

Console

The Console displays messages and a command prompt for the TimeQuest analyzer. The Console has two tabs: the **Console** tab and the **History** tab. All information and warning messages appear in the **Console** tab. All executed Synopsys Design Constraints files and Tcl commands are recorded in the

History tab. You can rerun a command in the **History** tab after the timing netlist has been updated by right-clicking the command you want to repeat, and then clicking **Rerun**.

Report Pane

The **Report** pane lists the reports you generate from the **Tasks** pane and those that are generated by any custom report commands. Once you select a report from the **Report** pane, the report appears in the **View** pane. If a report is out-of-date with respect to the current constraints, a "?" icon is shown next to the report.

You can use the **Write SDC File** command to save the constraints you have created to a Synopsys Design Constraints File (**.sdc**).

View Pane

The **View** pane displays timing analysis results, including any summary reports, custom reports, or histograms. Figure 2 shows the **View** pane when you use the Report Clocks command in the **Tasks** pane for a design that includes two defined clocks, clk and clkx2.

Figure 2. Output from Report Clocks Shown in the View Pane

Clocks Summary					
Clock Name	Type	Period	Frequency	Rise	Fall
1 clk	Base	1.000	1000.0 MHz	0.000	0.500
2 clkx2	Base	1.000	1000.0 MHz	0.000	0.500

The TimeQuest analyzer reports results only when requested. You can customize each report on demand to display specific timing information.

Specifying Timing Constraints

You can make individual timing constraints for individual entities, nodes, and pins with the **Constraints** menu of the TimeQuest analyzer. Individual timing assignments override project-wide requirements. You can also asssign timing exceptions to nodes and paths to avoid reporting of incorrect

or irrelevant timing violations. The TimeQuest analyzer supports point-to-point timing constraints, wildcards to identify specific nodes when making constraints, and assignment groups to make individual constraints to groups of nodes.

You can make the following types of individual timing assignments in the TimeQuest analyzer:

■ **Clock settings**—Allow you to perform an accurate multiclock timing analysis by defining the timing requirements and relationship of all clock signals in the design. The TimeQuest analyzer supports both single-clock and multiclock frequency analysis.

■ **Clock uncertainty assignments—**Allow you to specify the expected clock setup or hold uncertainty (jitter) that should be used when performing setup and hold checks. The TimeQuest analyzer subtracts the specified setup uncertainty from the data required time when calculating setup checks and adds the specified hold uncertainty to the data required time when calculating hold checks.

■ **Input and Output Delays—**Allow you to specify external device or board timing parameters by specifying the required data arrival times at specified input and output ports relative to the clock.

You can make the following types of individual timing exceptions as assignments in the TimeQuest analyzer:

■ **Multicycle paths**—Paths between registers that require more than one clock cycle to become stable. You can set multicycle paths to instruct the analyzer to relax its measurements and avoid incorrect setup or hold time violations.

■ **False paths**—You can designate as false paths any paths in the design which the timing analyzer disregards during analysis and reporting. By default, the Quartus II software cuts (directs the timing analyzer to ignore) paths between unrelated clock domains when there are no timing requirements set or only the default required f_{MAX} clock setting is used. The Quartus II software also cuts paths between unrelated clock domains if individual clock assignments are set but there is no defined relationship between the clock assignments.

- **Maximum delay requirements**—Requirements for input or output maximum delay, or maximum timing requirements for t_{SU}, t_H, t_{PD}, and t_{CO} on specific nodes in the design. You can make these assignments to specific nodes or groups to override project-wide maximum timing requirements.

- **Minimum delay requirements**—Requirements for input or output minimum delay, or minimum timing requirements for t_H, t_{PD}, and t_{CO} for specific nodes or groups. You can make these assignments to specific nodes or groups to override project-wide minimum timing requirements.

☞ **Using the quartus_sta executable**

You can also run the TimeQuest analyzer separately at the command prompt or in a script by using the **quartus_sta** executable. You must run the Quartus II Fitter executable **quartus_fit** before running the TimeQuest analyzer.

The **quartus_sta** executable creates a separate text-based report file that can be viewed with any text editor.

You can also launch the **quartus_sta** Tcl scripting shell, to run timing-related Tcl commands, by typing the following command at a command prompt:

```
quartus_sta -s ⏎
```

If you want to get help on the **quartus_sta** executable, type one of the following commands at the command prompt:

```
quartus_sta --h ⏎
quartus_sta --help ⏎
quartus_sta --help=<topic name> ⏎
```

Additionally, the **quartus_staw** executable provides the GUI for the TimeQuest analyzer as a stand-alone application.

For Information About	Refer To
Specific timing settings and performing a timing analysis in the Quartus II Software	"About the TimeQuest Timing Analyzer" in Quartus II Help
	Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the *Quartus II Handbook*
	"Module 4: Run Timing Analysis" in the Quartus II Interactive Tutorial

Early Timing Estimation

When you use the **Start Early Timing Estimate** command, the Compiler performs a complete Analysis & Synthesis but stops before fitting is complete. You can then review the early timing estimates in the TimeQuest analyzer reports, just as you would view regular timing analysis results; however, all early timing estimates are preliminary.

To generate data for an early timing estimate before completely fitting the design, on the Processing menu, point to **Start**, and then click **Start Early Timing Estimate**. You can specify options for early timing estimation in the **Early Timing Estimate** page under **Compilation Process Settings** in the **Settings** dialog box on the Assignments menu. You can select from the following options:

- **Realistic**
- **Optimistic**
- **Pessimistic**

Figure 3 shows the **Early Timing Estimate** page.

Figure 3. Early Timing Estimate Page in the Settings Dialog Box
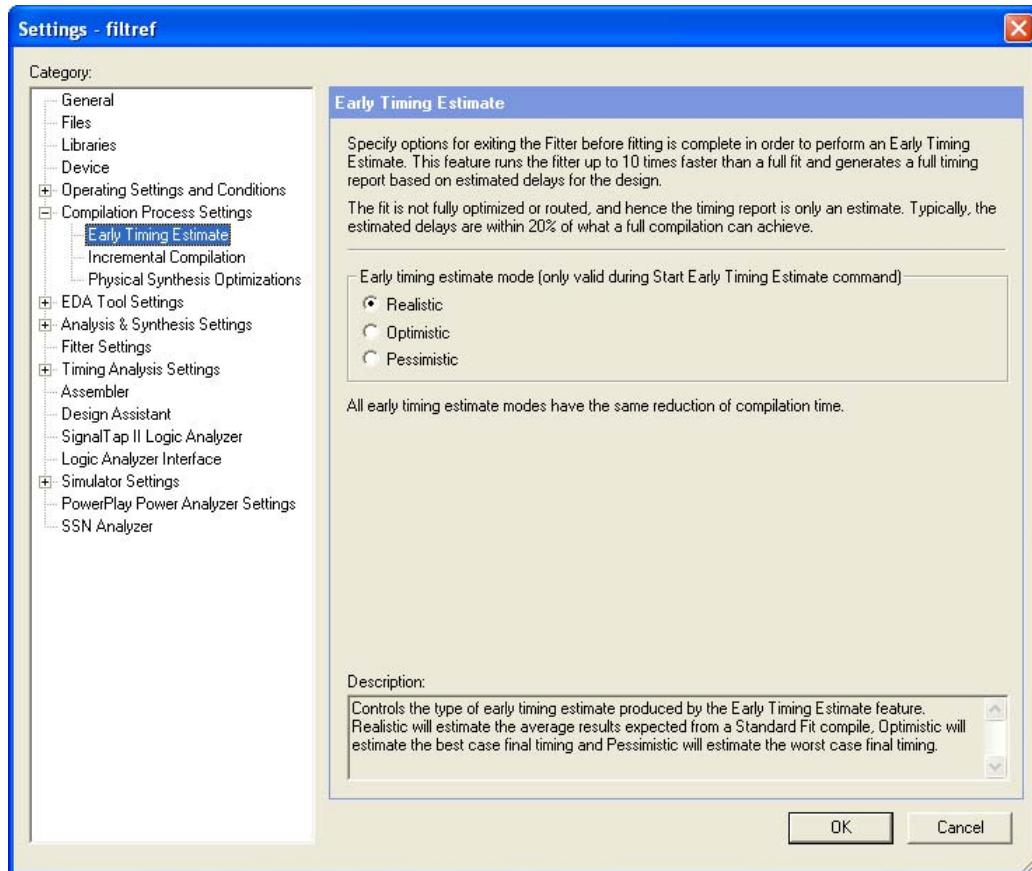
Viewing Timing Information for a Path

You can use the **Report Timing** dialog box to generate comprehensive timing information for any path or paths in your design. You can specify the number of paths to report, the type of path (including minimum timing paths), and how to report the information.

The **Report Timing** dialog box allows you to filter reported paths. For information on every constrained path in the design (except false paths), leave the fields in the **Report Timing** dialog box unchanged and click **Report Timing**. (Figure 4).

Figure 4. Report Timing Dialog Box
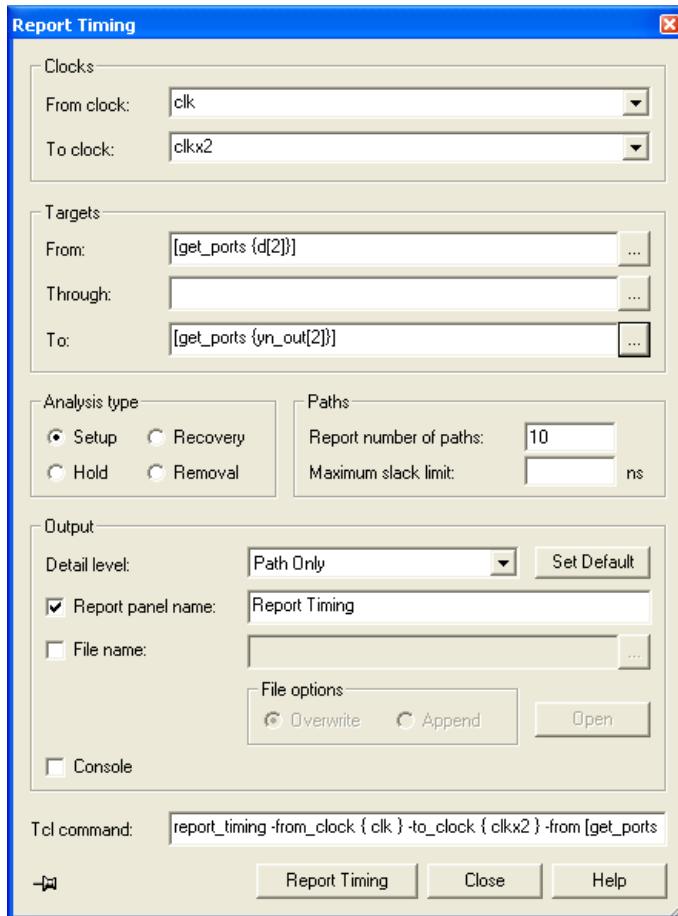
When information about a path is reported by the TimeQuest analyzer, you can use the **Locate Path** command directly from the timing analyzer reports to view path information in the Chip Planner, Technology Map Viewer, and RTL Viewer.

You can also use the **Locate Path** command to take advantage of the Chip Planner features for making assignments to a specific path. For more information on using the Chip Planner, refer to "Using the Chip Planner" on page 139 in Chapter 10, "Timing Closure."

Viewing Timing Delays with the Technology Map Viewer

The Quartus II Technology Map Viewer provides a low-level, or atom-level, technology-specific schematic representation a design. The Technology Map Viewer includes a schematic view, and also includes a hierarchy list, which lists the instances, primitives, pins, and nets for the entire design netlist.

After performing timing analysis or performing a full compilation that includes timing analysis, you can use the Technology Map Viewer to view the nodes that make up a timing path, including information about total delay and individual node delay (Figure 5).

To view timing information in the Technology Map Viewer, right-click path information in a timing analyzer report, and then click **Locate Path**. In the **Locate** dialog box, under **Locate in**, select **Technology Map Viewer**.

Figure 5. Technology Map View Window—Delay Information
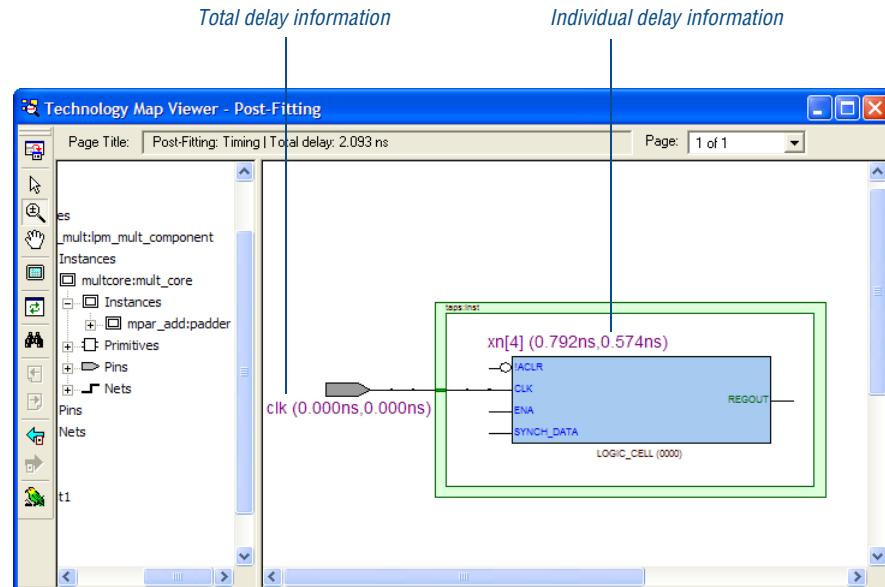
For Information About	Refer To
Using the Quartus II Technology Map Viewer	*Analyzing Designs with Quartus II Netlist Viewers* chapter in volume 1 of the *Quartus II Handbook*

Performing Timing Analysis with EDA Tools

The Quartus II software supports timing analysis and minimum timing analysis with the Synopsys PrimeTime software on Linux and board-level timing analysis with the Mentor Graphics Tau board-level verification tools.

To generate the necessary output files for performing timing analysis in EDA timing analysis tools, specify the appropriate timing analysis tool in the **Timing Analysis** and **Board-Level** pages under **EDA Tool Settings** in the **Settings** dialog box, and then perform a full compilation. Figure 6 shows the **Timing Analysis** page under **EDA Tool Settings**.

Figure 6. Timing Analysis Page of Settings Dialog Box
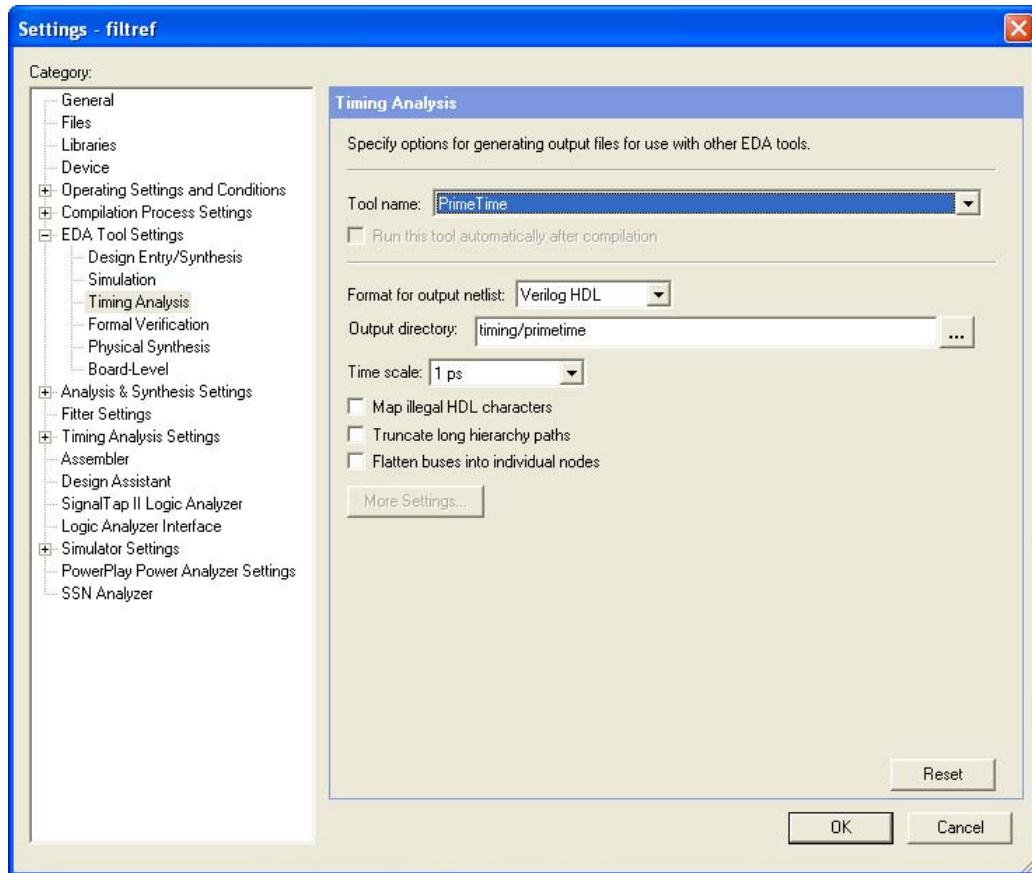
You can also generate the files by pointing to **Start** on the Processing menu, and then clicking **Start EDA Netlist Writer** after an initial compilation. If you are using the NativeLink feature, you can also run a timing analysis after an initial compilation by clicking **Run EDA Timing Analysis Tool** on the Tools menu.

> ☞ **Using the quartus_eda executable**
>
> You can also run the EDA Netlist Writer to generate the necessary output files separately at the command prompt or in a script by using the **quartus_eda** executable. You must run the Quartus II Fitter executable **quartus_fit** before running the EDA Netlist Writer.
>
> The **quartus_eda** executable creates a separate text-based report file that can be viewed with any text editor.
>
> If you want to get help on the **quartus_eda** executable, type one of the following commands at the command prompt:
>
> ```
> quartus_eda -h ↵
> quartus_eda -help ↵
> quartus_eda --help=<topic name> ↵
> ```

Using the PrimeTime Software

The Quartus II software generates a Verilog Output File or VHDL Output File, a Standard Delay Format Output File (**.sdo**) that contains timing delay information, and a Tcl Script File (**.tcl**) that sets up the PrimeTime environment.

With the NativeLink feature, you can specify that the Quartus II software launches the PrimeTime software in either command-line or GUI mode. You can also specify a Synopsys Design Constraints File that contains timing assignments for use in the PrimeTime software.

The following steps describe the basic flow to manually use the PrimeTime software to perform timing analysis on a design after compilation in the Quartus II software:

1. Specify EDA tool settings, either in the **Settings** dialog box on the Assignments menu, or during project setup, with the **New Project Wizard,** on the File menu.

2. Compile your design in the Quartus II software to generate the output netlist files. The Quartus II software places the files in a tool-specific directory.

3. Source the Quartus II-generated Tcl Script File to set up the PrimeTime environment.

4. Perform timing analysis in the PrimeTime software.

Using the Tau Software

The Quartus II software generates STAMP model files that can be imported into the Tau software to perform board-level timing verification.

The following steps describe the basic flow for generating STAMP model files:

1. Specify EDA tool settings, either in the **Settings** dialog box on the Assignments menu, or during project setup, using the **New Project Wizard** on the File menu.

2. Compile the design in the Quartus II software to generate the STAMP model files. The Quartus II software places the files in a tool-specific directory.

3. Use the STAMP model files in the Tau software to perform board-level timing verification.

For Information About	Refer To
Using the Synopsys PrimeTime software with the Quartus II software	*Synopsys PrimeTime Support* chapter in volume 3 of the *Quartus II Handbook*
Using the Mentor Graphics Tau software with the Quartus II software	"About Using the Tau Software with the Quartus II Software" in Quartus II Help

Chapter
Ten

Timing Closure

10

Introduction

The Quartus II software offers a fully integrated timing closure flow that allows you to meet your timing goals by controlling the synthesis and place and route of a design. Using the timing closure flow results in faster timing closure for complex designs, reduced optimization iterations, and automatic balancing of multiple design constraints.

The timing closure flow allows you to perform an initial compilation, view design results, and perform further design optimization efficiently. You can use Chip Planner to analyze the placement and routing of the design and make assignments, use the Timing Optimization Advisor to view recommendations for optimizing your design for timing, use netlist optimizations on the design after synthesis and during place and route, use LogicLock region assignments, and use the Design Space Explorer (DSE) to further optimize the design. Figure 1 shows the timing closure flow.

Figure 1. Timing Closure Flow
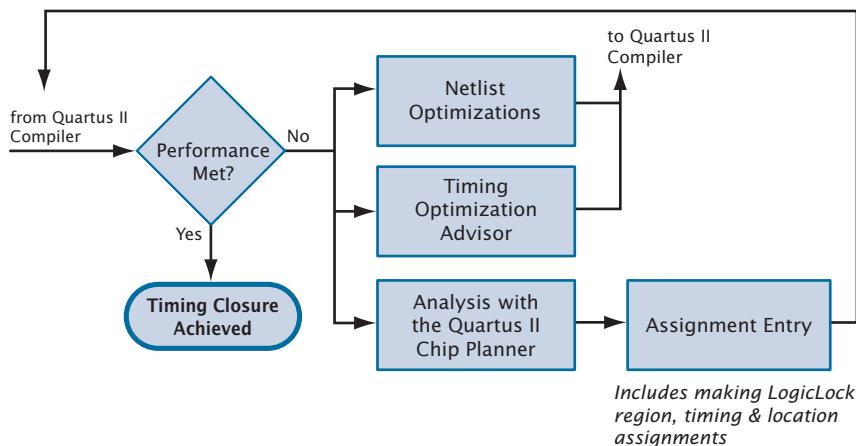
Using the Chip Planner

You can use the Chip Planner to view logic placement made by the Fitter, view user assignments and LogicLock region assignments, and routing information for a design. You can use this information to identify critical paths in the design and make timing assignments, location assignments, and LogicLock region assignments to achieve timing closure.

> ☞ **Using Chip Planner to Achieve Timing Closure**
>
> The Quartus II Chip Planner provides a single interface for viewing and making changes to the design floorplan as well as making ECO-style post-fit netlist changes. For the list of devices supported by Chip Planner, see Quartus II Help.

Chip Planner Tasks And Layers

The Chip Planner can simultaneously show user assignments and Fitter location assignments. You can customize the way the Chip Planner displays information with the **Task** list and the commands the View menu.

The following are the pre-defined tasks in the Chip Planner:

- Floorplan editing
- Post compilation editing
- Partition display
- Clock region assignment creation

You can use the **Layers Settings** command on the View menu to select more than one combination of these elements for a customized view of your design in the device. You can view global and local routing, ports, used and unused assignments, pin and location assignments, user and fitter-placed LogicLock regions, clock regions, and other elements in any combination.

Making Assignments

To facilitate achieving timing closure, the Chip Editor assignment tasks allow you to make or change location assignments directly in the floorplan. You can create and assign nodes or entities to custom regions and to LogicLock regions, and you can also edit existing assignments to logic cells, rows, columns, regions, MegaLAB structures, and LABs. You can also locate any node (or set of nodes) and make assignments in the Assignment Editor.

For Information About	Refer To
Working with the Chip Planner	*Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*
	"Displaying Resources and Information" in Quartus II Help
	"Working with Assignments in the Chip Planner" in Quartus II Help

Using Incremental Compilation to Achieve Timing Closure
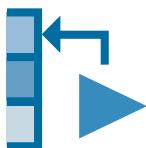
When making or changing assignments, you can use incremental compilation to achieve timing closure by assigning design partitions, compiling the design, and then changing assignments in one or more partitions while preserving the compilation results for the other design partitions.

For more information about incremental compilation, refer to "Top-Down Incremental Compilation Flow" on page 12 in Chapter 1, "Design Flow" and "Using Incremental Compilation" on page 86 in Chapter 6, "Place and Route".

Using the Timing Optimization Advisor

The Timing Optimization Advisor offers recommendations for optimizing your design for timing in the following areas:

- Maximum frequency (f_{MAX})
- Setup timing (t_{SU})
- Clock-to-output (t_{CO})
- Propagation delay (t_{PD})

If you have an open project, view the Timing Optimization Advisor by pointing to **Advisors** on the Tools menu, and then clicking **Timing Optimization Advisor**. If the project has not been compiled yet, the Timing Optimization Advisor provides only general recommendations for optimizing for timing. If the project has been compiled, however, the Timing Optimization Advisor can provide specific timing recommendations for the project, based on the project information and current settings. Figure 2 shows the Timing Optimization Advisor.

Figure 2. Timing Optimization Advisor

Using Netlist Optimizations to Achieve Timing Closure

The Quartus II software includes netlist optimization options to further optimize your design during synthesis and during place and route. Netlist optimizations are push-button features that offer improvements to f_{MAX} results by making modifications to the netlist to improve performance. These options can be applied regardless of the synthesis tool used. Depending on your design, some options may have more of an effect than others.

You can specify synthesis and physical synthesis netlist optimizations in the **Analysis & Synthesis Settings** page and **Physical Synthesis Optimizations** page of the **Settings** dialog box. See Figure 3 on page 143.

Netlist optimizations for synthesis include the following options:

- **Timing Driven Synthesis**—Directs the Quartus II software to synthesize your design as directed by timing analysis results from a previous compilation, where possible.

- **Perform WYSIWYG primitive resynthesis**—Directs the Quartus II software to unmap WYSIWYG primitives during synthesis. When this option is turned on, the Quartus II software unmaps the logic elements in an atom netlist to gates, and remaps the gates to Altera LCELL primitives. This option allows the Quartus II software to use techniques specific to a device architecture during the remapping process and uses the optimization technique (**Speed**, **Balanced**, or **Area**) that you specified in the **Analysis & Synthesis Settings** page of the **Settings** dialog box.

- **Perform register retiming**—Allows registers to be moved across combinational logic to balance timing, but does not change the functionality of the current design. This option moves registers across combinational gates only, and not across user-instantiated logic cells, memory blocks, DSP blocks, or carry or cascade chains, and has the ability to move registers from the inputs of a combinational logic block to the block's output, potentially combining the registers. It can also create multiple registers at the input of a combinational logic block from a register at the output of a combinational logic block.

Figure 3. Netlist Optimizations
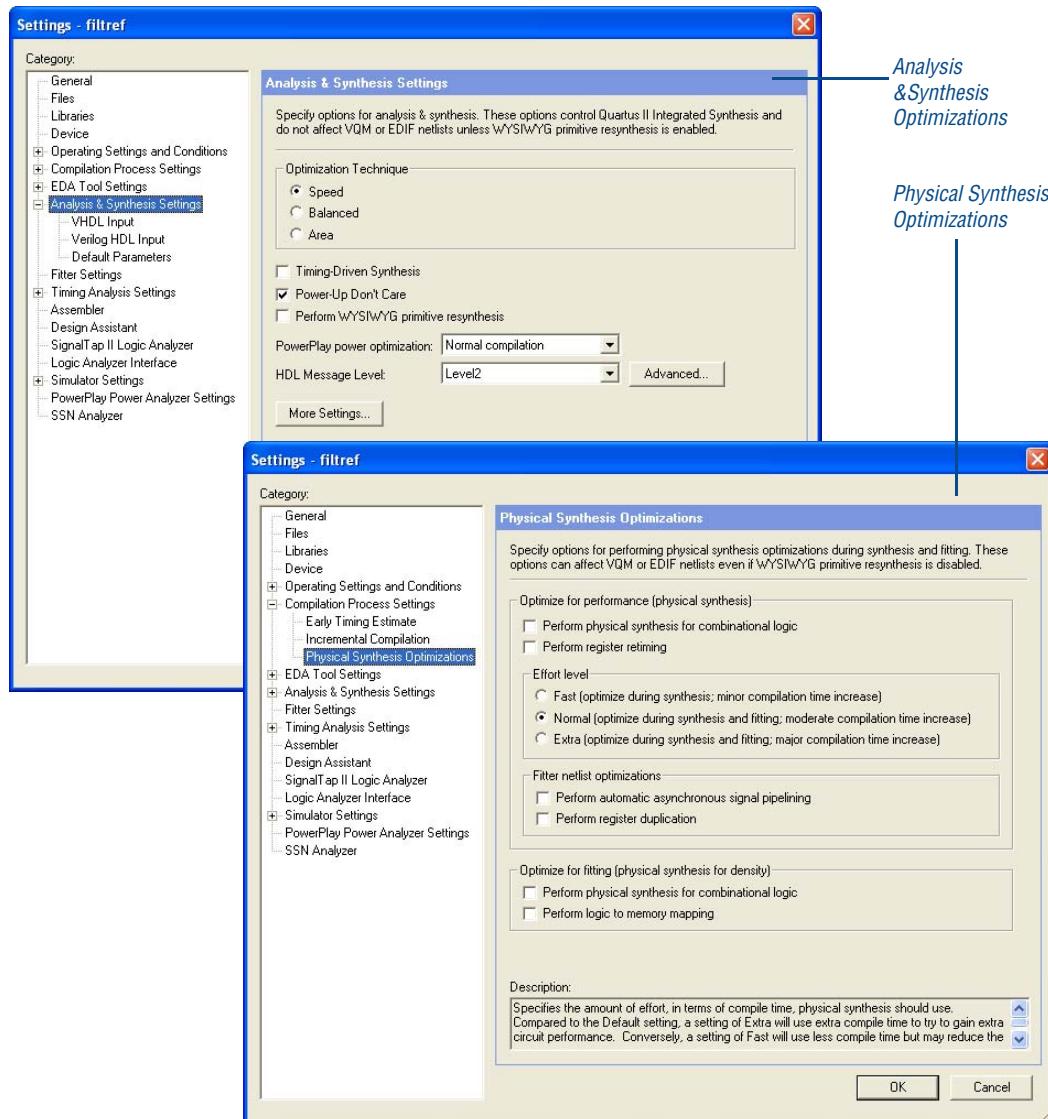
Netlist optimizations for physical synthesis and fitting include the following groups of options:

■ **Optimize for performance (physical synthesis)**: Options to perform physical synthesis optimizations on combinational logic, and to perform register retiming, during fitting.

■ **Effort level**: Specifies the level of effort used by the Quartus II software when performing physical synthesis (**Normal**, **Extra**, and **Fast**).

■ **Optimize for fitting (physical synthesis for density)**: Options to reduce combinational logic elements and registers in a design by eliminating duplicate nodes and by mapping logic to unused memory blocks.

The Quartus II software cannot perform these netlist optimizations for fitting and physical synthesis on a back-annotated design. In addition, if you use one or more of these netlist optimizations on a design, and then back-annotate the design, you must generate a Verilog Quartus Mapping File (**.vqm**) if you wish to save the results. The Verilog Quartus Mapping File must be used in place of the original design source code in future compilations.

For Information About	Refer To
Achieving timing closure using netlist optimizations	*Netlist Optimizations and Physical Synthesis* chapter in volume 2 of the *Quartus II Handbook*

Using LogicLock Regions to Preserve Timing

You can use LogicLock regions to achieve timing closure by analyzing your design in the Chip Planner, and then constraining critical logic in LogicLock regions. Defining hierarchical LogicLock regions can give you more control over the placement and performance of modules or groups of modules. You can use the LogicLock feature on individual nodes, for instance, by assigning the nodes along the critical path to a LogicLock region.

Successfully improving performance by using LogicLock regions requires a detailed understanding of the critical paths in your design. Once you have implemented LogicLock regions and attained the desired performance, back-annotate the contents of the region to lock the logic placement.

Soft LogicLock Regions

LogicLock regions have predefined boundaries and nodes assigned to a particular region that always reside within the boundary or LogicLock region size. Soft LogicLock regions can enhance design performance by removing the fixed boundaries of LogicLock regions. With the soft region property enabled, the Fitter attempts to place as many assigned nodes in the region as close together as possible, and has the added flexibility of moving nodes outside the soft region to meet the performance requirements for your design.

For Information About	Refer To
Using LogicLock regions in floor planning	*Analyzing and Optimizing the Design Floorplan* chapter in volume 2 of the *Quartus II Handbook*

Path-Based Assignments

The Quartus II software enables you to assign specific source and destination paths to LogicLock regions, allowing for easy grouping of critical design nodes into a LogicLock region. You can create path-based assignments with the **Add Path** dialog box, by dragging and dropping critical paths from the Timing Analyzer reports and the Chip Planner into LogicLock regions.

The **Add Path** dialog box allows you to specify a path by identifying a source and destination node and using wildcards when identifying nodes. You can click **List Nodes** to determine how many nodes are assigned to the LogicLock region. You open this dialog box by clicking **Add Path** in the **General** tab of the **LogicLock Region Properties** dialog box, or by double-clicking a path in the Chip Planner. Figure 4 shows the **Add Path** dialog box.

Figure 4. Add Path Dialog Box
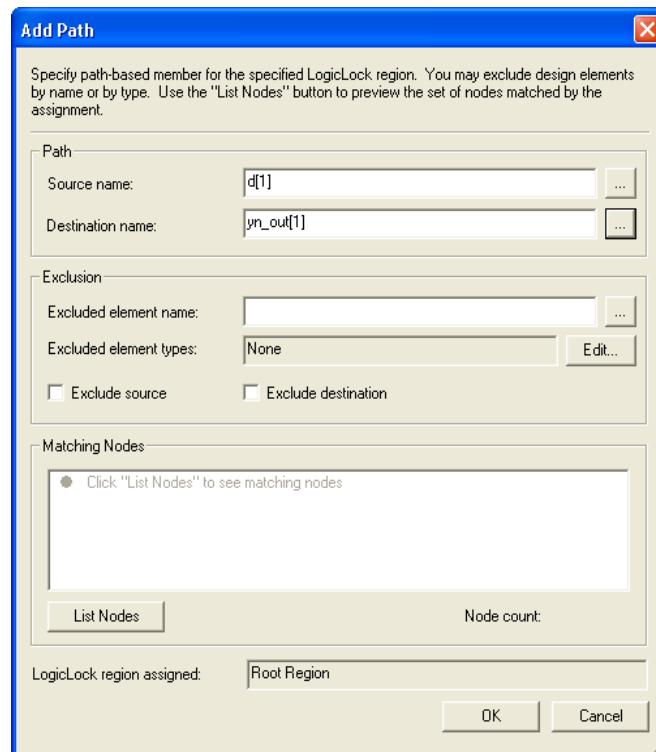
Using the Design Space Explorer to Achieve Timing Closure

You can use the Design Space Explorer (DSE) to optimize your design for timing. The DSE interface allows you to explore a range of Quartus II options and settings automatically to determine which settings should be used to obtain the best possible result for the project. You can specify the level of change DSE can evaluate, your optimization goals, the target device, and the allowable compilation time.

To run the Design Space Explorer click **Launch Design Space Explorer** on the Tools menu. For more information on using the Design Space Explorer, refer to "Using the Design Space Explorer" on page 97 in Chapter 6, "Place and Route".

Chapter Eleven

Power Analysis
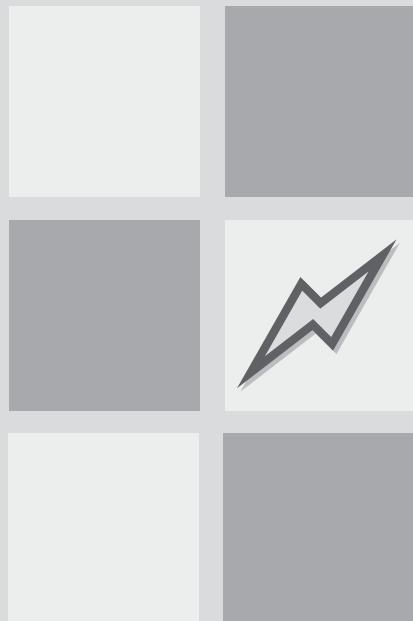
11

Introduction

The Quartus II PowerPlay Power Analysis Tools provide an interface that allows you to estimate static and dynamic power consumption throughout the design cycle. The PowerPlay Power Analyzer performs postfitting power analysis and produces a power report that highlights, by block type and entity, the power consumed. The Altera PowerPlay Early Power Estimator estimates power consumption at other stages of the design process and produces a Microsoft Excel-based spreadsheet with estimate information.

Figure 1. PowerPlay Power Analysis Flow
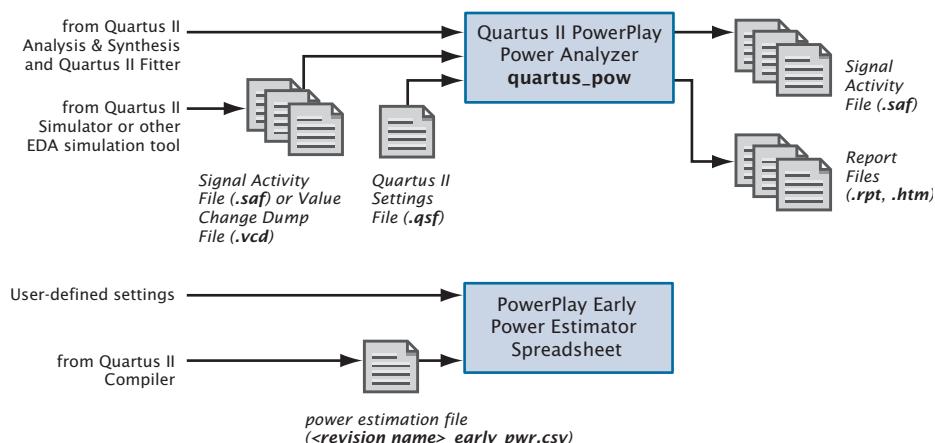
Power Analysis with the PowerPlay Power Analyzer

You can use the **PowerPlay Power Analyzer Tool** command on the Processing menu after running Analysis & Synthesis and the Fitter successfully. You can specify whether you want to use an input file, such as a Signal Activity File (**.saf**) or Value Change Dump File (**.vcd**) generated by the Quartus II Simulator or a Value Change Dump File generated by another EDA simulation tool, to initialize toggle rates and static probabilities during power analysis, and also whether you want the signal activities used during power analysis written to an output file. In addition, you can specify

entity-based toggle rates and static probabilities using user assignments in the Quartus II user interface or in the Quartus II Settings File (**.qsf**). For some device families, the Quartus II software fills in any missing signal activity information by analyzing the design topology and function.

You can then start power analysis by clicking **Start** in the Power Analyzer Tool window. When power analysis is complete, click **Report** to display the Report File (**.rpt**, **.htm**). Figure 2 shows the **Power Play Power Analyzer Tool** dialog box.

Figure 2. PowerPlay Power Analyzer Tool

The Start button starts power analysis.

The progress bar shows the elapsed time spent processing the design.

The Report button displays the Power Analysis section of the Report File.
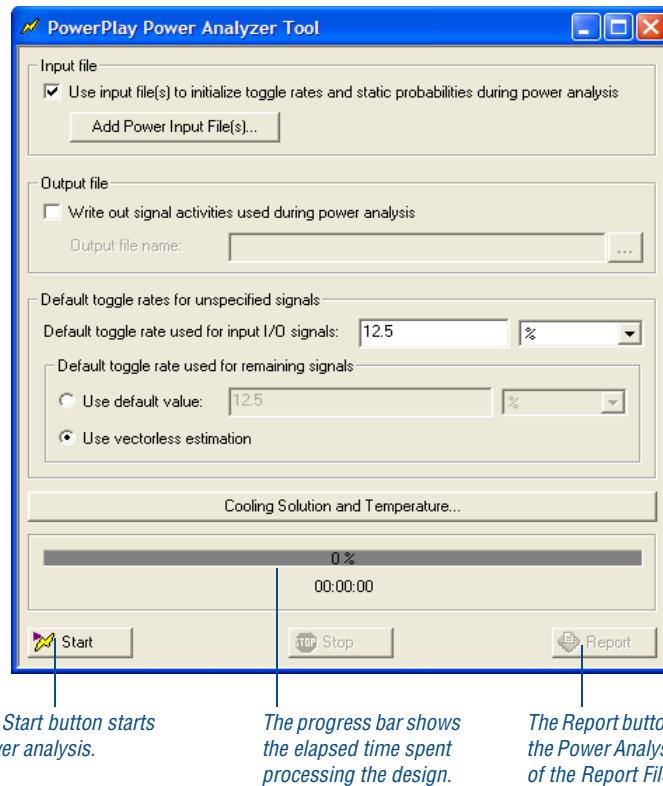
☞ **Using the quartus_pow executable**

You can also run the PowerPlay Power Analyzer separately at the command prompt or in a script by using the **quartus_pow** executable. You must run the Quartus II Fitter, **quartus_fit** (and in some cases **quartus_asm**), successfully before running the PowerPlay Power Analyzer.

The **quartus_pow** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_pow** executable, type one of the following commands at the command prompt:

```
quartus_pow -h ↵
quartus_pow -help ↵
quartus_pow --help=<topic name> ↵
```

For Information About	Refer To
Using the Quartus II PowerPlay Power Analyzer	*PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*
	"PowerPlay Power Analyzer Window" and "About Power Estimation and Analysis" in Quartus II Help

Specifying Power Analyzer Options

Specify default settings for power analysis in the **PowerPlay Power Analyzer Settings** page, which is available from the **Settings** dialog box on the Assignments menu. You can specify default settings for what type of input file is used, what type of output file is written, and whether the signal activities are written to the report file, as well as settings for default toggle rates. See Figure 3.

Figure 3. PowerPlay Power Analyzer Settings Page of Settings Dialog Box
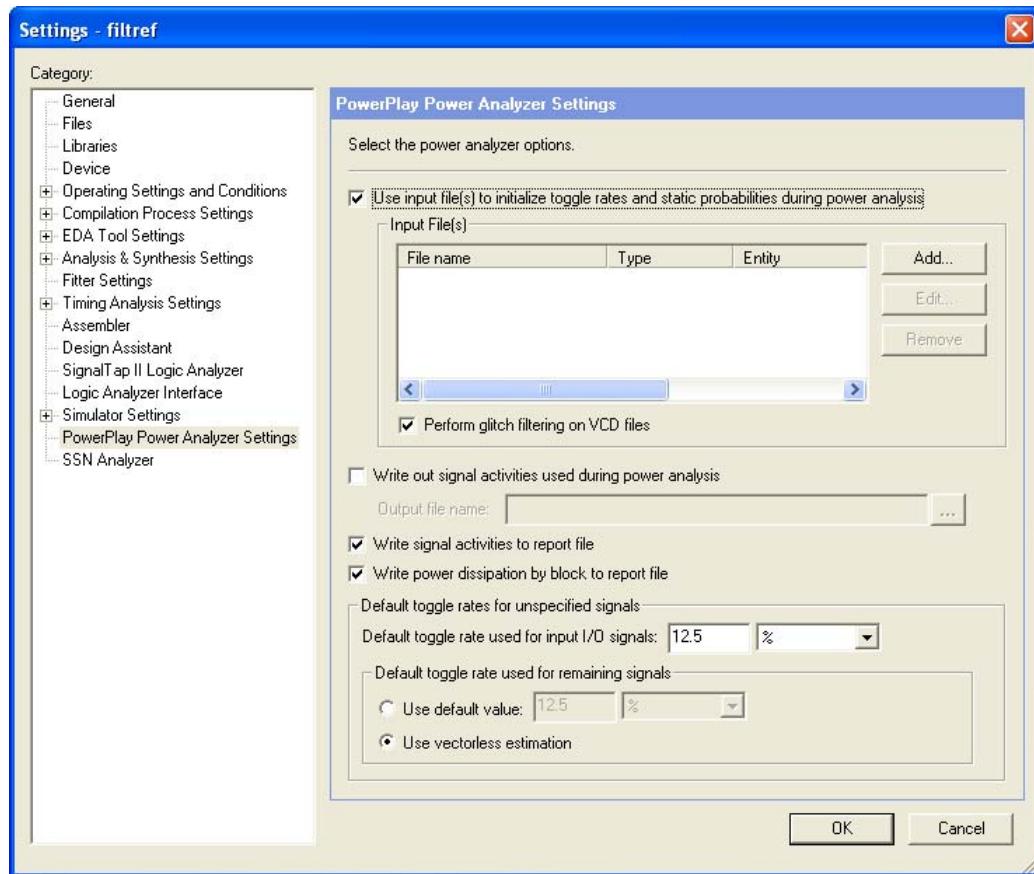
Depending on the target device family, you can also specify default operating conditions for power analysis. You can specify the junction temperature, cooling solution requirements, and device characteristics in the **Operating Settings and Conditions** pages of the **Settings** dialog box.

PowerPlay Early Power Estimator Spreadsheets

You can calculate power requirements for certain device families with the Altera PowerPlay Early Power Estimator spreadsheets, which you can download from the Power Consumption section of the Altera website. If you have not started the FPGA design, or if it is only partially complete, you can use PowerPlay Early Power Estimator spreadsheets to provide a preliminary estimate of the power requirements of the design. A macro in the Excel-based PowerPlay Early Power Estimator spreadsheet calculates the power estimation and then provides a current (I_{CC}) and power (P) estimation.

You can use the PowerPlay Early Power Estimator to estimate power at any stage of the design process; however, Altera recommends that you use the PowerPlay Power Analyzer, rather than the PowerPlay Early Power Estimator, after the design is complete in order to obtain the most accurate power analysis.

If you use the PowerPlay Early Power Estimator before you start your design, you can specify device resources, operating frequency, toggle rates, and other parameters for the PowerPlay Early Power Estimator. If you use it after you have created a design, you can compile the design in the Quartus II software and then use the **Generate Power Play Early Power Estimator File** command on the Project menu to generate a power estimation file, which is a text-based file named **<revision name>_early_pwr.csv** that contains power information for the current device and design. You can then import this power estimation file into the PowerPlay Early Power Estimator.

☞ **Using Early Power Estimations**

Power calculations that are provided by the PowerPlay Early Power Estimator should be used only as an estimation of power, not as a specification. Be sure to verify the actual I_{CC} during device operation, because this measurement is sensitive to the actual device design and the environmental operating conditions.

For Information About	Refer To
Using the PowerPlay Early Power Estimator	*PowerPlay Early Power Estimator User Guide* on the Altera website
	PowerPlay Power Analysis chapter in volume 3 of the *Quartus II Handbook*
	"About Power Estimation and Analysis" in Quartus II Help
Information about device requirements	Individual device handbooks or data sheets on the Altera website

Chapter Twelve

Programming & Configuration

12

Introduction

Once you have successfully compiled a project with the Quartus II software, you can program or configure an Altera device. The Assembler module of the Quartus II Compiler generates programming files that the Quartus II Programmer can use to program or configure a device with Altera programming hardware. You can also use a stand-alone version of the Quartus II Programmer to program and configure devices. Figure 1 shows the programming design flow.

Figure 1. Programming Design Flow
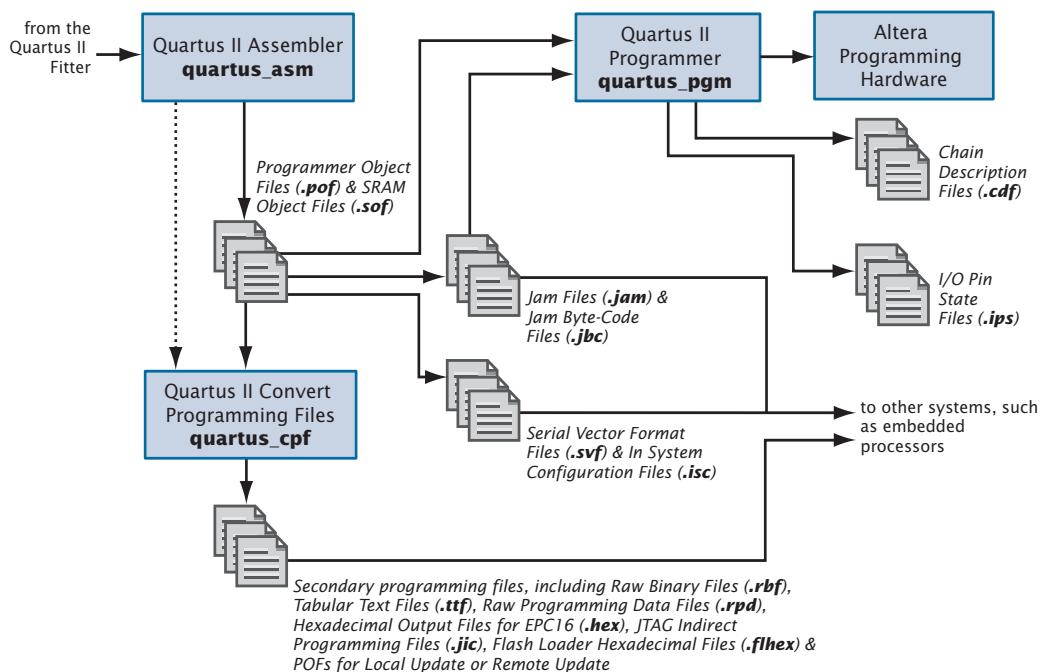
Creating and Using Programming Files

The Assembler automatically converts the Fitter's device, logic cell, and pin assignments into a programming image for the device, in the form of one or more Programmer Object Files (**.pof**) or SRAM Object Files (**.sof**) for the target device.

You can start a full compilation in the Quartus II software, which includes the Assembler module, or you can run the Assembler separately.

☞ Using the quartus_asm executable

You can also run the Assembler separately at the command prompt or in a script by using the **quartus_asm** executable. You must run the Quartus II Fitter executable, **quartus_fit**, successfully before running the Assembler.

The **quartus_asm** executable creates a separate text-based report file that can be viewed with any text editor.

If you want to get help on the **quartus_asm** executable, type one of the following commands at the command prompt:

```
quartus_asm -h ↵
quartus_asm -help ↵
quartus_asm --help=<topic name> ↵
```

You can also direct the Quartus II software to generate programming files in other formats by using one of the following methods:

■ The **Device and Pin Options** dialog box, which is available on the **Device** page of the **Settings** dialog box, allows you to specify optional programming file formats, such as Hexadecimal (Intel-Format) Output Files (**.hexout**), Tabular Text Files (**.ttf**), Raw Binary Files (**.rbf**), Jam™ Files (**.jam**), Jam Byte-Code Files (**.jbc**), Serial Vector Format Files (**.svf**), and In System Configuration Files (**.isc**).

■ The **Create JAM, SVF, or ISC File** command under **Create/Update** on the File menu generates Jam Files, Jam Byte-Code Files, Serial Vector Format Files, or In System Configuration Files.

■ The **Create/Update IPS File** command under **Create/Update** on the File menu displays the **ISP CLAMP State Editor** dialog box, which allows you to create or update I/O Pin State Files (**.ips**) that contain pin state information for specific devices used to configure pin states during programming.

■ The **Convert Programming Files** command on the File menu combines and converts SRAM Object Files and Programmer Object Files for one or more designs into other secondary programming file formats, such as Raw Programming Data Files (**.rpd**), HEXOUT Files for EPC16 or SRAM, Programmer Object Files, Programmer Object Files for Local Update or Remote Update, Raw Binary Files, Tabular Text Files, JTAG Indirect Configuration Files (**.jic**), and Flash Loader Hexadecimal Files (**.flhex**).

These secondary programming files can be used in embedded processor-type programming environments, and, for some Altera devices, by other programming hardware.

The Programmer uses the Programmer Object Files and SRAM Object Files generated by the Assembler to program or configure all Altera devices supported by the Quartus II software. You use the Programmer with Altera programming hardware, such as the MasterBlaster™, ByteBlasterMV™, ByteBlaster™ II, USB-Blaster™, or EthernetBlaster download cable; or the Altera Programming Unit (APU).

☞ **Using the Stand-Alone Programmer**

If you want to use only the Quartus II Programmer, you can install the stand-alone version of the Quartus II Programmer, **quartus_pgmw**, instead of installing the complete Quartus II software.

The Programmer allows you to create a Chain Description File (**.cdf**) that contains the name and options of devices used for a design. You can also open a JTAG Chain File (**.jcf**) or FLEX Chain File (**.fcf**) and save it in the Quartus II Programmer as a Chain Description File.

For some programming modes that allow programming or configuring multiple devices, the Chain Description File also specifies top-to-bottom order of the SRAM Object Files, Programmer Object Files, Jam Files, Jam Byte-Code Files, and devices used for a design, as well as the order of the devices in the chain. Figure 2 shows the Programmer window.

Figure 2. Programmer Window
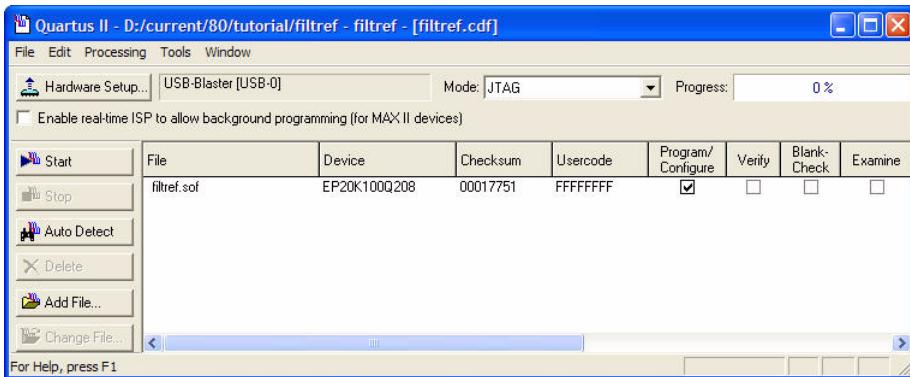
> **☞ Using the quartus_pgm executable**
>
> You can also run the Programmer separately at the command prompt or in a script by using the **quartus_pgm** executable. You may need to run the Assembler executable, **quartus_asm**, in order to produce a programming file before running the Programmer.
>
> If you want to get help on the **quartus_pgm** executable, type one of the following commands at the command prompt:
>
> ```
> quartus_pgm -h ↵
> quartus_pgm -help ↵
> quartus_pgm --help=<topic name> ↵
> ```

The Programmer has four programming modes:

■ Passive Serial
■ JTAG
■ Active Serial
■ In-Socket

The Passive Serial and JTAG programming modes allow you to program single or multiple devices using a Chain Description File and Altera programming hardware. You can program a single EPCS1 or EPCS4 serial configuration device using Active Serial Programming mode and Altera programming hardware. You can program a single CPLD or configuration device using In-Socket Programming mode with a Chain Description File and Altera programming hardware.

If you want to use programming hardware that is not available on your computer, but is available via a JTAG server, you can also use the Programmer to specify and connect to remote JTAG servers.

For Information About	Refer To
General programming information	"Programming Files" glossary definition, "Programming Devices" and "About Programming" in Quartus II Help
Using the Programmer	*Quartus II Programmer* chapter in volume 3 of the *Quartus II Handbook* "Module 6: Configure a Device" in the Quartus II Interactive Tutorial
Altera programming hardware	*MasterBlaster Serial/USB Communications Cable User Guide*, *ByteBlaster II Download Cable User Guide*, *ByteBlasterMV Download Cable User Guide*, *USB-Blaster Download Cable User Guide*, and *EthernetBlaster Communications Cable User Guide* on the Altera website
Device-specific programming information	The *Configuration Handbook* on the Altera website

Converting Programming Files

You can use the **Convert Programming Files** dialog box on the File menu to combine and convert SRAM Object Files or Programmer Object Files into other programming file formats. For example, you can add a remote update-enabled SRAM Object File to a Programmer Object File for Remote Update, which is used to program a configuration device in remote update configuration mode, or you can convert a Programmer Object File into a HEXOUT File for EPC16 for use by an external host. Or you can convert a POF into a Raw Programming Data File for use with some configuration devices. You can also convert SRAM Object Files or Programmer Object Files into JTAG Indirect Configuration Files, which you can use to program the configuration data for certain device families into an EPCS1 or EPCS4 serial configuration device.

You can use the **Convert Programming Files** dialog box to set up output programming files by arranging the chain of SRAM Object Files stored in a HEXOUT File for SRAM, Programmer Object Files, Raw Binary Files, or

Tabular Text Files, or by specifying a Programmer Object File to be stored in a HEXOUT File for EPC16. The settings you specify in the **Convert Programming Files** dialog box are saved to a Conversion Setup File (**.cof**) that contains information such as device and file names, device order, device properties, and file options. Figure 3 shows the **Convert Programming Files** dialog box.

Figure 3. Convert Programming Files Dialog Box
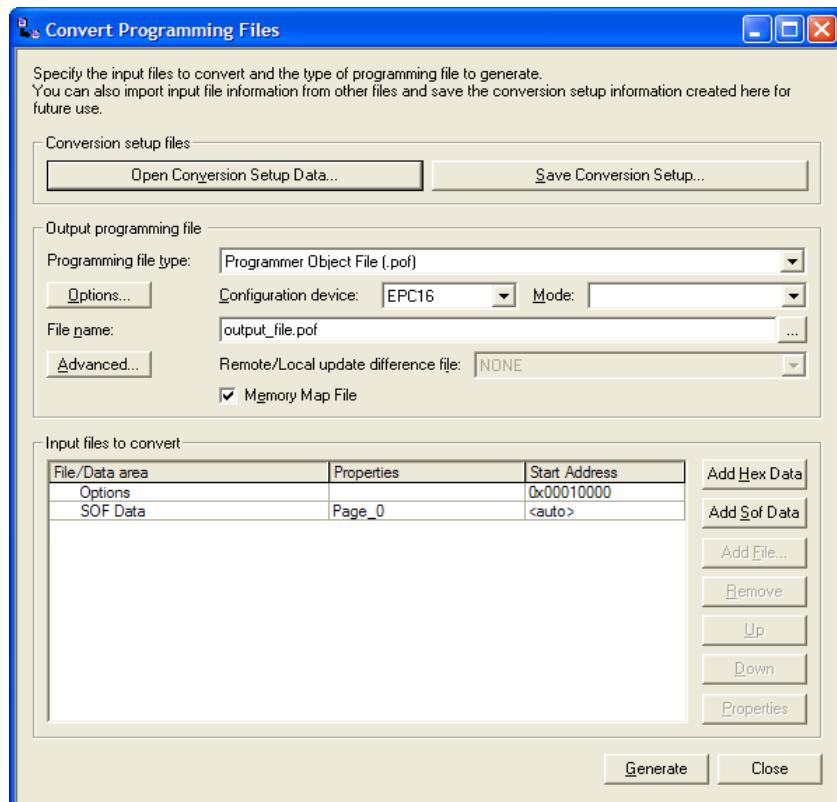
For a Programmer Object File for an EPC4, EPC8, or EPC16 configuration device, you can also specify the following information:

- Establish different configuration bitstreams, which are stored in pages in the configuration memory space.
- Create parallel chains of SRAM Object Files within each page.
- Arrange the order of SRAM Object Files and Hexadecimal (Intel-Format) Files (**.hex**) stored in flash memory.

- Specify the properties of **SOF Data** items and HEX Files.
- Add or remove **SOF Data** items from the configuration memory space.
- Create Memory Map Files (**.map**).

For Programmer Object Files for Local Update and Programmer Object Files for Remote Update, you can specify the following information:

- Add or remove remote update enabled Programmer Object Files and remote update enabled SRAM Object Files from the configuration memory space.
- Specify the properties of **SOF Data** items.
- Add or remove **SOF Data** items.
- Create Memory Map Files, and generate remote update difference files and local update difference files.

You can also use the **Convert Programming Files** dialog box to arrange and combine multiple SRAM Object Files into a single Programmer Object Files in Active Serial Configuration mode. The Programmer Object File can be used to program an EPCS1 or EPCS4 serial configuration device, which can then be used to configure multiple devices.

☞ Using the quartus_cpf executable

You can also run the Convert Programming Files feature separately at the command prompt or in a script by using the **quartus_cpf** executable. You may need to run the Assembler executable, **quartus_asm**, in order to produce a programming file before running the Programmer.

If you want to get help on the **quartus_cpf** executable, type one of the following commands at the command prompt:

```
quartus_cpf -h ↵
quartus_cpf -help ↵
quartus_cpf --help=<topic name> ↵
```

For Information About	Refer To
Converting programming files and generating different kinds of programming files	"Generating Secondary Programming files" and "Convert Programming Files Dialog Box" in Quartus II Help
In-system programmability and in-circuit reconfigurability	*Configuration Handbook* on the Altera website
In-system programming	"Module 6: Configure a Device" in the Quartus II Interactive Tutorial

Using the Quartus II Software to Program Via a Remote JTAG Server

In the **Hardware Setup** dialog box, which is available from the **Hardware** button in the Programmer window or on the Edit menu, you can add remote JTAG servers that you can connect to, for example, to use programming hardware that is not available on your computer, and configure local JTAG server settings so remote users can connect to your local JTAG server.

You can specify that remote clients should be enabled to connect to the JTAG server in the **Configure Local JTAG Server** dialog box, which is available from the **JTAG Settings** tab of the **Hardware Setup** dialog box.

You can specify the remote server you want to connect to in the **Add Server** dialog box, which is available from the **JTAG Settings** tab of the **Hardware Setup** dialog box. When you connect to a remote server, the programming hardware type that is attached to the remote server is displayed in the **Hardware Settings** tab.

For Information About	Refer To
Using a Local JTAG Server	"Configure Local JTAG Server Dialog Box" and "Adding a JTAG Server" in Quartus II Help

Chapter
Thirteen

Debugging

13

Introduction

The Quartus II SignalTap II Logic Analyzer, the External Logic Analyzer Interface, the SignalProbe feature, the In-System Memory Content Editor, and the In-System Sources and Probes Editor enable you to analyze internal device nodes and I/O pins while operating in-system and at system speeds. The SignalTap II Logic Analyzer is an embedded logic analyzer that routes the signal data through the JTAG port to the Quartus II software based on user-defined trigger conditions. You can use the External Logic Analyzer Interface to connect an off-chip logic analyzer to nodes in the design. The SignalProbe feature uses otherwise unused device routing resources to route selected signals to an external logic analyzer or oscilloscope. The In-System Memory Content and In-System Sources and Probes Editors allow you to view and modify, at run-time, data in a design.

Figure 1 and Figure 2 show the SignalTap II and SignalProbe debugging flows.

Figure 1. SignalTap II Debugging Flow
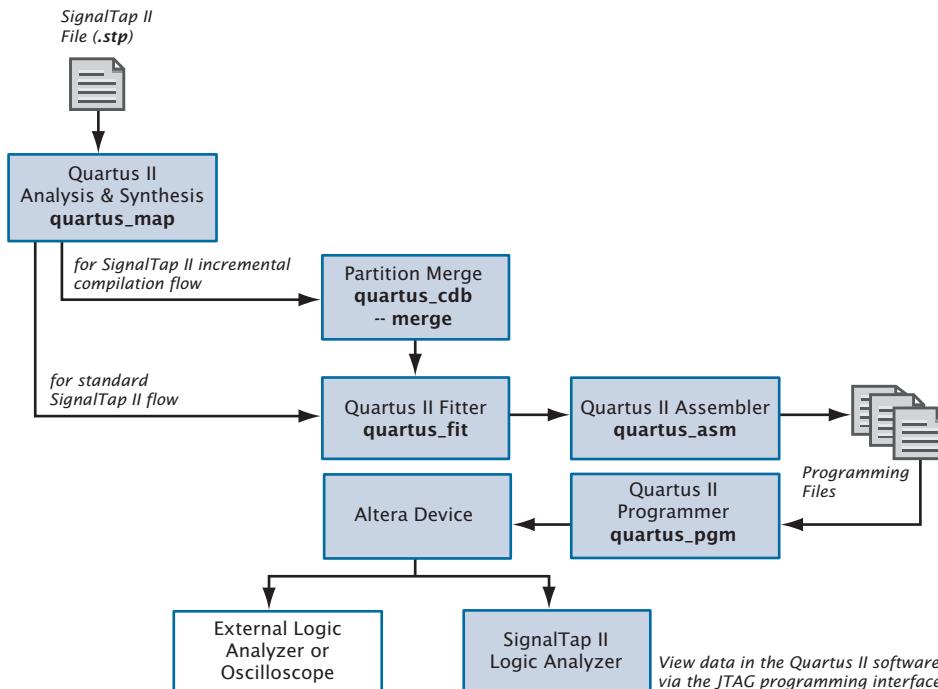
Figure 2. SignalProbe Debugging Flow
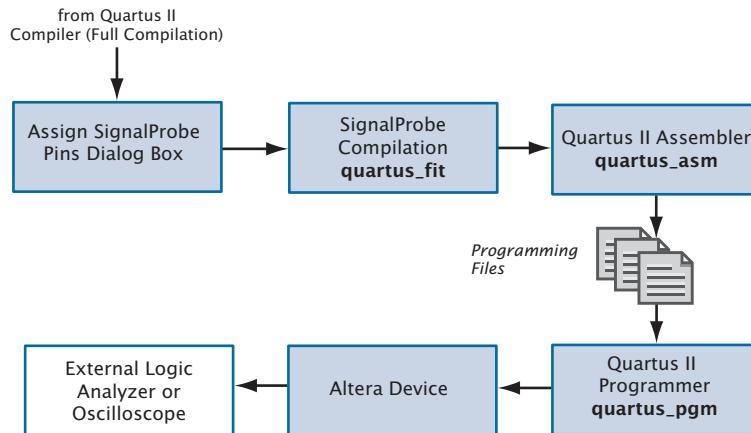
Using the SignalTap II Logic Analyzer

The SignalTap II Logic Analyzer is a system-level debugging tool that captures and displays real-time signal behavior, allowing you to observe interactions between hardware and software in system designs. The Quartus II software allows you to select which signals to capture, when signal capture starts, and how many data samples to capture. You can also select whether the data is routed from the device's memory blocks to the SignalTap II Logic Analyzer via the JTAG port, or to the I/O pins for use by an external logic analyzer or oscilloscope.

You can use a MasterBlaster, ByteBlasterMV, ByteBlaster II, USB-Blaster, or EthernetBlaster communications cable to download configuration data to the device. These cables are also used to upload captured signal data from the RAM resources of the device to the Quartus II software. The Quartus II software then displays data acquired by the SignalTap II Logic Analyzer as waveforms.

Setting Up the SignalTap II Logic Analyzer

To use the SignalTap II Logic Analyzer, you must first create a SignalTap II File (**.stp**), which includes all the configuration settings and displays the captured signals as a waveform. Once you have set up the SignalTap II File, you can compile the design, program the device, and use the logic analyzer to acquire and analyze data.

Each logic analyzer instance is embedded in the logic on the device. The SignalTap II Logic Analyzer supports up to 2,048 channels and 128K samples on a single device.

After compilation, you can run the SignalTap II Logic Analyzer with the **Run Analysis** command on the Processing menu.

The following steps describe the basic flow of setting up a SignalTap II File and acquire signal data:

1. Create a new SignalTap II File.

2. Add instances to the SignalTap II File and nodes to each instance. You can use the SignalTap II filters in the Node Finder to find all pre-synthesis and post-fitting SignalTap II nodes.

3. Assign a clock to each instance.

4. Set other options, such as sample depth and trigger level, and assign signals to the data/trigger input and debug port.

5. If necessary, specify advanced trigger conditions.

6. Compile the design.

7. Program the device.

8. Acquire and analyze signal data in the Quartus II software or with an external logic analyzer or oscilloscope.

Figure 3 shows the SignalTap II Logic Analyzer.

Figure 3. The SignalTap II Logic Analyzer

Instance Manager   Setup View   JTAG Chain Configuration
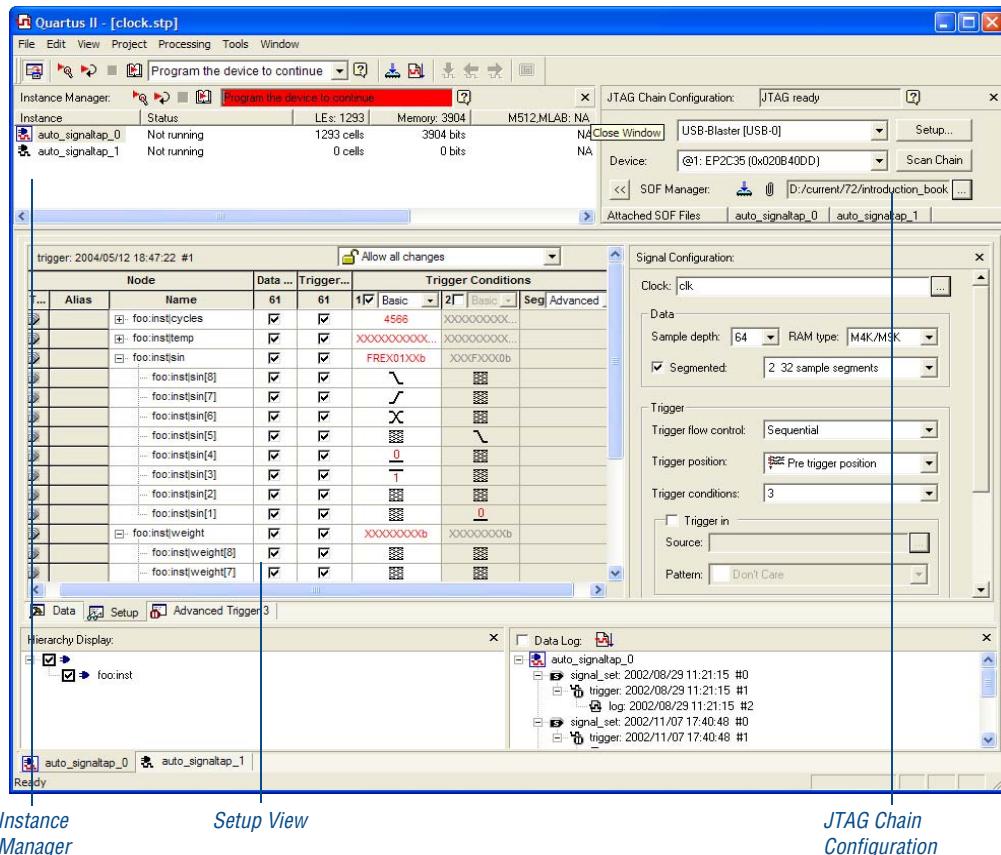
☞ **Using the Stand-Alone SignalTap II Logic Analyzer**

If you want to use only the SignalTap II Logic Analyzer, you can use the stand-alone graphical user interface version of the SignalTap II Logic Analyzer, **quartus_stpw**.

You can use the following features to set up the SignalTap II Logic Analyzer:

■ **Instance Manager**: The Instance Manager allows you create and perform SignalTap II logic analysis on multiple embedded instances of the logic analyzer in each device. You can use it to create, delete, rename, and apply settings to separate and unique logic analyzer instances in the SignalTap II File. The Instance Manager displays all instances in the current SignalTap II File, the current status of each

associated instance, and the number of logic elements and memory bits used in the associated instance. The Instance Manager helps you to check the amount of resource usage that each logic analyzer requires on the device. You can start multiple logic analyzers at the same time by selecting them and clicking **Run Analysis** on the Processing menu.

■ **Triggers**: A trigger is a pattern of logic events defined by logic levels, clock edges, and logical expressions. The SignalTap II Logic Analyzer supports multilevel triggering, multiple trigger positions, multiple segments, and external trigger events. You can set trigger options in the **Signal Configuration** panel in the SignalTap II Logic Analyzer window and specify advanced triggers by selecting **Advanced** in the **Trigger Levels** column in the **Setup** tab of the SignalTap II Logic Analyzer window.

Advanced triggers provide the ability to build flexible, user-defined logic expressions and conditions based on the data values of internal buses or nodes. On the **Advanced Trigger** tab, you can drag and drop symbols from the **Node List** and the **Object Library** to create a logical expression composed of logical, comparison, bitwise, reduction, shift operators, and event counters. Figure 4 shows the **Advanced Trigger** tab of the SignalTap II window.

Figure 4. Advanced Trigger Tab of the SignalTap II Window
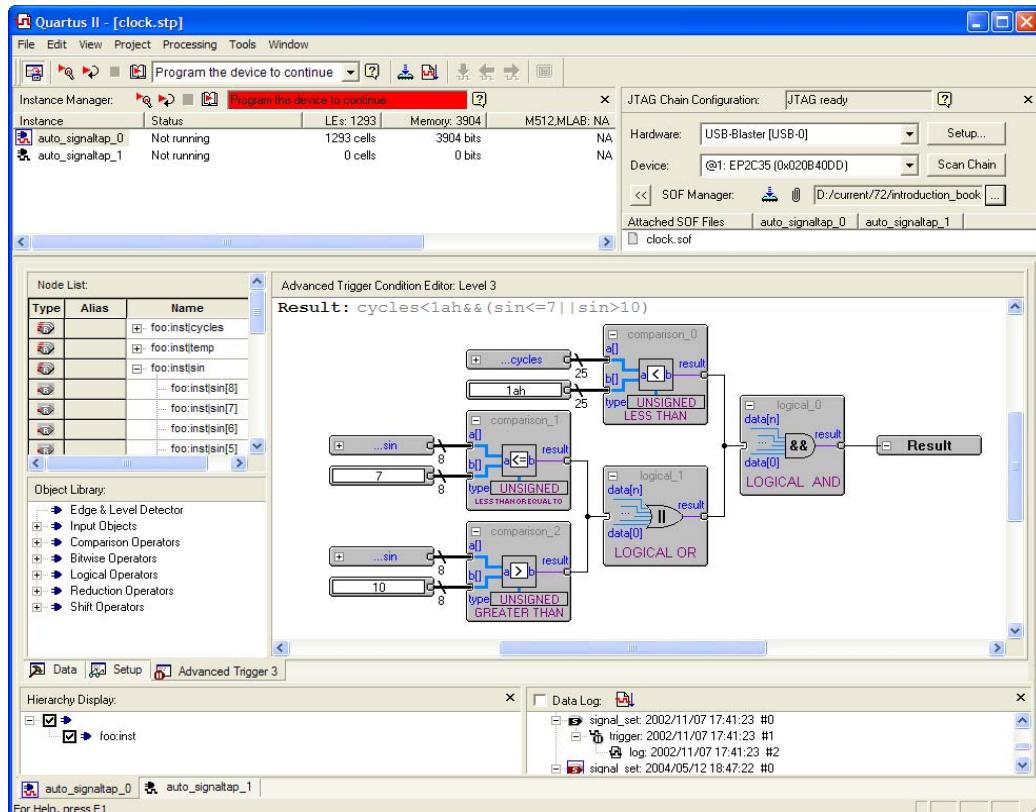
You can configure the logic analyzer with up to ten trigger levels, which help you view only the most significant data. You can specify four separate trigger positions: pre, center, post, and continuous. The trigger position allows you to specify the amount of data that should be acquired before the trigger and after.

Segmented mode allows you to capture data for recurring events without allocating a large sample depth by segmenting the memory into discrete time periods.

■   **Attaching Programming File**: Allows you to have multiple SignalTap II configurations (trigger setups) and the associated programming files in a single SignalTap II File. You can use the SOF Manager to add, rename, or remove SRAM Object Files (**.sof**), extract SRAM Object Files from the SignalTap II File, or program the device.

Logic Analysis with Incremental Compilation

The incremental compilation feature helps to shorten the debugging process time considerably by allowing you to analyze post-fitting nodes incrementally with the SignalTap II Logic Analyzer without performing a full compilation of the design each time you modify your analysis.

For more information about incremental compilation, refer to "Top-Down Incremental Compilation Flow" on page 12 in Chapter 1, "Design Flow." and "Using Incremental Compilation" on page 86 in Chapter 6, "Place and Route."

For Information About	Refer To
Using Quartus II incremental compilation	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" in Quartus II Help

Analyzing SignalTap II Data

When you use the SignalTap II Logic Analyzer to view the results of a logic analysis, the data is stored in the internal memory on the device and then streamed to the waveform view in the logic analyzer, via the JTAG port.

In the waveform view, you can insert time bars, align node names, and duplicate nodes; create, rename, and ungroup a bus; specify a data format for bus values; and print the waveform data. The data log that is used to create the waveform shows a history of data that is acquired with the SignalTap II Logic Analyzer. The data is organized in a hierarchical manner; logs of captured data with the same trigger are grouped together in **Trigger Sets**. Figure 5 shows the waveform view.

Figure 5. SignalTap II Waveform View
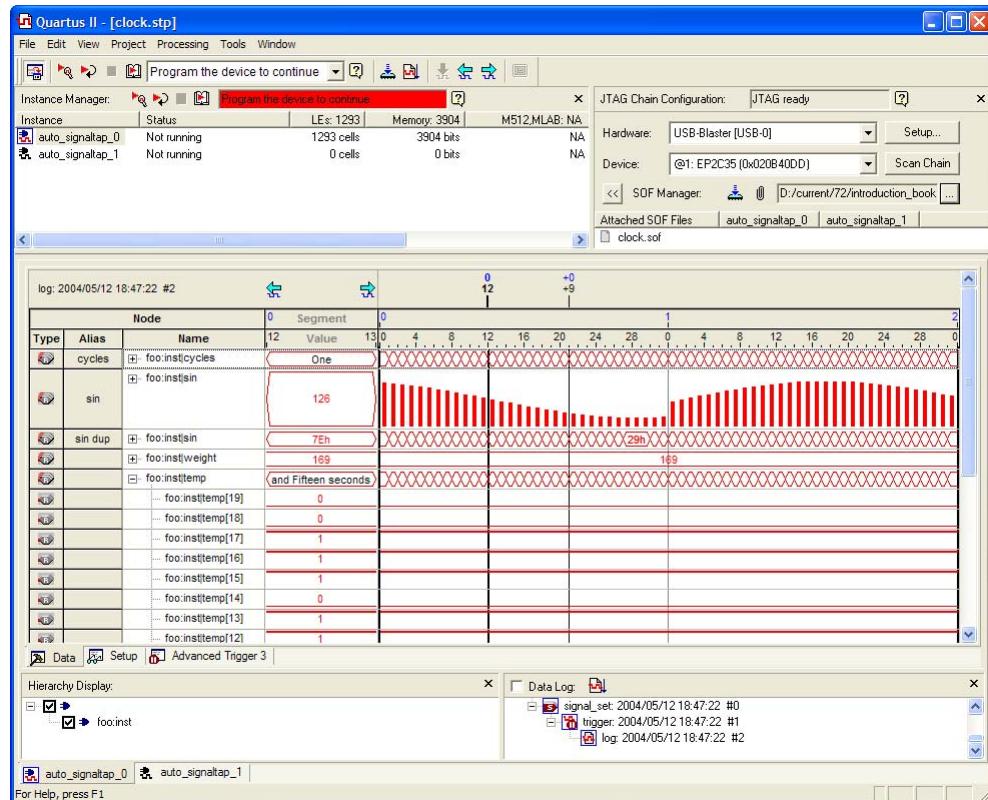
The **Waveform Export** utility allows you to export the acquired data to the following industry-standard formats that can be used by other tools:

- Comma Separated Values File (**.csv**)
- Table File (**.tbl**)
- Value Change Dump File (**.vcd**)
- Vector Waveform File (**.vwf**)
- Joint Photographic Experts Group File (**.jpeg**)
- Bitmap File (**.bmp**)

You can also configure the SignalTap II Logic Analyzer to create mnemonic tables for a group of signals. The mnemonic table feature allows you to assign a predefined name to a set of bit patterns, so that captured data is more meaningful. See Figure 6.

Figure 6. Mnemonic Table Setup Dialog Box
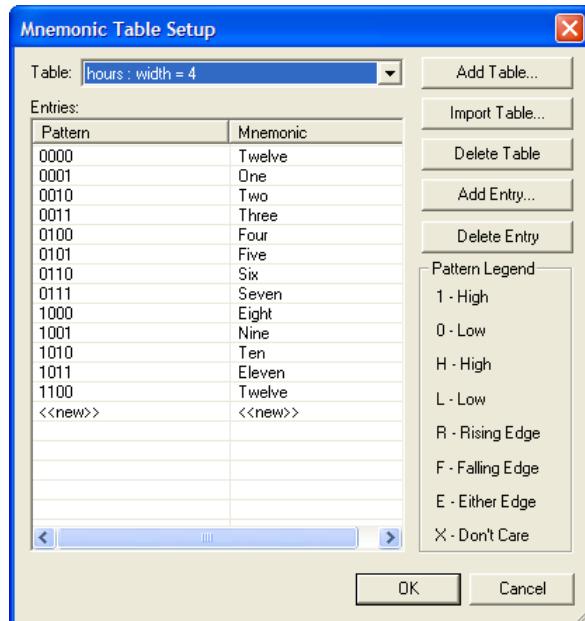
For Information About	Refer To
Using the SignalTap II Logic Analyzer	*Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*
	"About the SignalTap II Logic Analyzer" in Quartus II Help
	"Module 8: SignalTap II Logic Analyzer" in the Quartus II Interactive Tutorial

Using an External Logic Analyzer

The Logic Analyzer Interface is logic within the device you use to connect a large set of internal device signals to a small number of output pins for debugging purposes. The Logic Analyzer Interface enables you to connect to and transmit internal signals buried within your FPGA to an external logic analyzer for analysis. The Logic Analyzer Interface allows you to debug a large set of internal signals using a small number of output pins. In the

Quartus II Logic Analyzer Interface, the internal signals are grouped together, distributed to a user-configurable multiplexer, and then output to available I/O pins on your FPGA. Instead of having a one-to-one relationship between internal signals to output pins, the Quartus II Logic Analyzer Interface enables you to map many internal signals to a smaller number of output pins. The exact number of internal signals that you can map to an output pin varies based on the multiplexer settings in the Logic Analyzer Interface.

Logic Analyzer Interface Files (**.lai**) appear in the Logic Analyzer Interface Editor window.

Edit the Logic Analyzer Interface File to specify the number of pins and banks to use, and the capture mode. In the **Output/Capture mode** list, specify **Combinational/Timing** or **Registered State**. In the **Clock** box, specify the clock signal associated with the design. In the **Power-up state** list, select **Bank 0** or **Tri-stated**.

To specify the nodes to be observed, in the **Setup View** pane, select a bank from the list. Click the table of banks to open the Node Finder to find and use any node name in a Quartus II project after you have performed compilation. In the **Nodes Found** list, select the node names you want to analyze. Click **OK**. Connect the probe points from your external logic analyzer to the debug header of your device.

To enable the Logic Analyzer Interface for a project, turn on **Enable Logic Analyzer Interface** on the **Logic Analyzer Interface** page of the **Settings** dialog box on the Assignments menu. In the **Logic Analyzer Interface file name** box, specify the name of the Logic Analyzer Interface File you want to enable. Click **OK**, and then program the device.

For Information About	Refer To
Using External Logic Analyzers	*In-System Debugging Using External Logic Analyzers* chapter in volume 3 of the *Quartus II Handbook*
	"About the Logic Analyzer Interface Editor" in Quartus II Help

Using SignalProbe

The SignalProbe feature allows you to route user-specified signals to output pins without affecting the existing fitting in a design, so that you can debug signals without having to perform another full compilation. Starting with a fully routed design, you can select and route signals for debugging through I/O pins that were either previously reserved or are currently unused.

The SignalProbe feature allows you to specify which signals in the design to debug, perform a SignalProbe compilation that connects those signals to unused or reserved output pins, and then send the signals to an external logic analyzer. You can use the Node Finder when assigning pins to find the available SignalProbe sources. A SignalProbe compilation typically takes approximately 20% to 30% of the time required for a standard compilation.

To use the SignalProbe feature to reserve pins and perform a SignalProbe compilation on a design:

1. Perform a full compilation of the design.

2. Select signals for debugging and the I/O pins to route the signals, and point to **Signal Probe Pins** on the Tools menu, and then click **SignalProbe Pins**. Figure 7 shows the **Signal Probe Pins** dialog box.

Figure 7. SignalProbe Pins Dialog Box
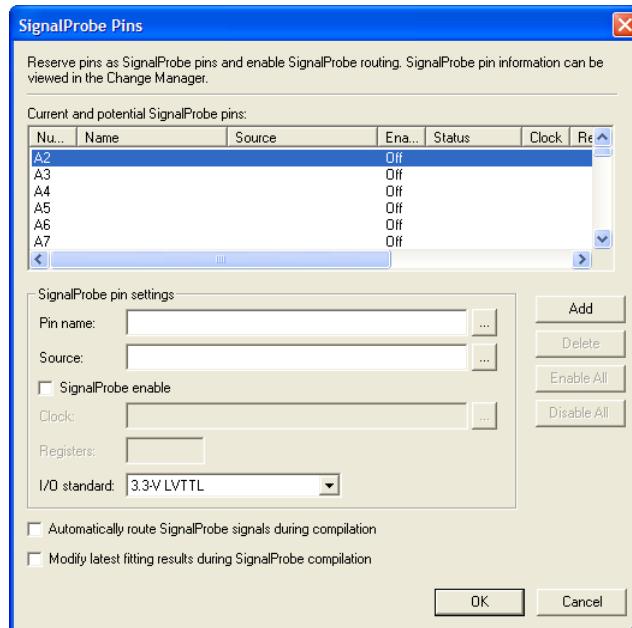
3. Perform a SignalProbe compilation, by pointing to **Start** on the Processing menu, and then clicking **Start SignalProbe Compilation**.

4. Configure the device with the new programming data to examine the signals.

You can also use the register pipelining feature to force signal states to output on a clock edge, or to delay a signal output. You can also use register pipelining to synchronize multiple SignalProbe outputs from a bus of signals.

You can use the SignalProbe feature with Tcl. With Tcl commands, you can add and remove SignalProbe assignments and sources, perform a SignalProbe compilation on a design, and compile routed SignalProbe signals in a full compilation.

For Information About	Refer To
Using the SignalProbe feature	*Quick Design Debugging Using SignalProbe* chapter in volume 3 of the *Quartus II Handbook*
	"About SignalProbe" in Quartus II Help

Using the In-System Memory Content Editor

The In-System Memory Content Editor allows you to view and modify, at run-time, RAM, ROM, or register content independently of the system clock of a design. You analyze design memory with the In-System Memory Content Editor through a JTAG interface using standard programming hardware.

You can enable RAM and ROM for the In-System Memory Content Editor with the **MegaWizard Plug-In Manager** on the Tools menu when generating lpm_rom, lpm_ram_dq, altsyncram, and lpm_constant megafunctions or when instantiating these megafunctions directly in the design, with the LPM_HINT megafunction parameter.

The In-System Memory Content Editor captures and updates data in the device. You can export or import data in Memory Initialization File (**.mif**), Hexadecimal (Intel-Format) File (**.hex**), and RAM Initialization File (**.rif**) formats. The In-System Memory Content Editor offers the following features:

■ **Instance Manager**: contains a list of memory instances, including index, instance name, status, data width, data depth, type, and mode. The Instance Manager controls which memory blocks have data that is viewed, offloaded, or updated. Commands from the Instance Manager affect the entire selected memory block.

■ **JTAG Chain Configuration**: allows you to select the programming hardware and device to acquire data from or read data to, and to select the SRAM Object File (**.sof**) for programming.

■   **HEX Editor**: used to make edits and save changes to in-system memory at run-time, to display the current data within the memory block, and to update or offload selected sections of a memory block. You can use the **Go To** command shortcut to automatically go to a specific data address within a specific memory block within a specific instance. Words are displayed with each hexadecimal value separated by a space. Memory addresses are displayed in the left column, and the ASCII values (if the word width is a multiple of eight) in the right column. Each memory instance has a separate pane in the HEX Editor. Figure 8 shows the HEX Editor in the In-System Memory Content Editor window.

Figure 8. In-System Memory Content Editor Window
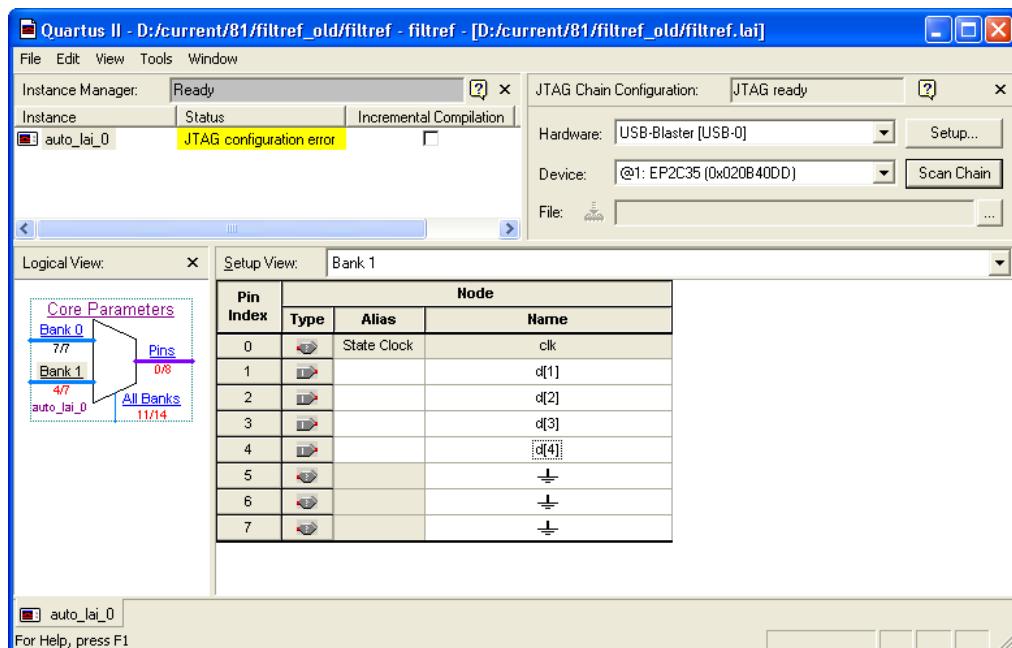
For Information About	Refer To
Using the In-System Memory Content Editor	*In-System Updating of Memory and Constants* chapter in volume 3 of the *Quartus II Handbook*
	"About the In-System Memory Content Editor" in Quartus II Help

Using the In-System Sources and Probes Editor

The In-System Sources and Probes Editor allows you to control all of the altsource_probe megafunction instances within your design. It displays all available instances in your design, provides a push-button interface to drive all of your source nodes, and a logging feature to store your probe and source data.

To add in-system sources and probes functionality to your design, you must first customize and instantiate the altsource_probe megafunction. Like any other megafunction, the altsource_probe megafunction can be easily customized using the **MegaWizard Plug-In Manager**. Each source or probe port can be up to 256 bits wide. You can have up to 128 instances of the altsource_probe megafunction in your design.

The Sources and Probes Editor window organizes and displays the data from all sources and probes in your design, organized according to the index numbers of the altsource_probe instances. The editor provides an easy way to manage your signals, allowing you to rename signals or to group them into buses. All data collected from source and probe nodes are recorded in the event log and displayed as a timing diagram. The In-System Sources and Probes Editor has the following features:

■ **JTAG Chain Configuration**—Allows you to specify programming hardware, device, and file settings that the In-System Sources and Probes Editor uses to program and acquire data from a device.

■ **Instance Manager**—Displays information about the instances generated when you compile a design, and allows you to control the data the In-System Sources and Probes Editor acquires.

■ **Sources and Probes Editor Window**—Displays the data read from the selected instance and allows you to modify source data to be written to your device.

Figure 9. In-System Sources and Probes Editor Window
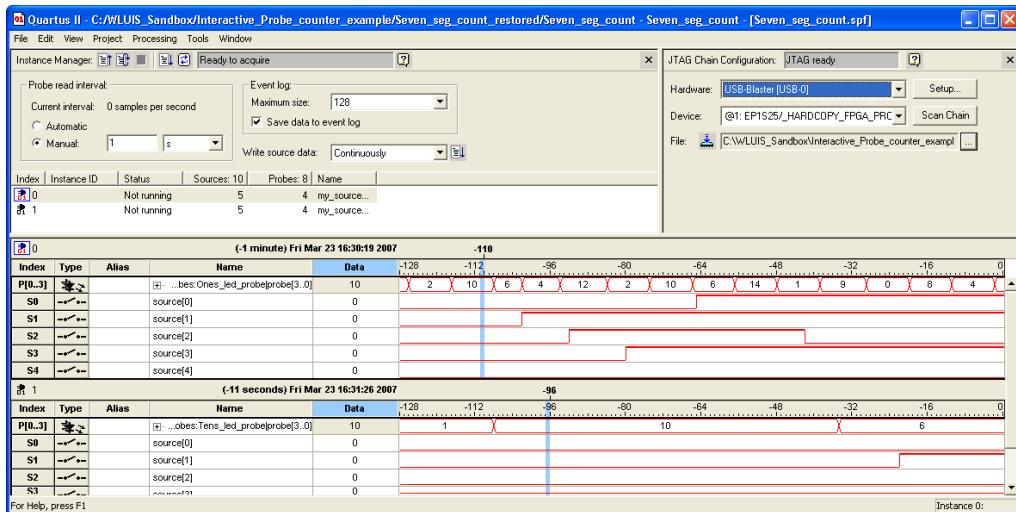
For Information About	Refer To
Using the In-System Sources and Probes Editor	*Design Debugging Using In-System Sources and Probes* chapter in volume 3 of the *Quartus II Handbook* "About the In-System Sources and Probes Editor" in Quartus II Help

Using the RTL Viewer & Technology Map Viewer For Debugging

You can use the RTL Viewer to analyze your design after analysis and elaboration is complete. The RTL Viewer provides a gate-level schematic view of your design and a hierarchy list, which lists the instances, primitives, pins, and nets for the entire design netlist. You can filter the information that appears in the schematic view and navigate through different pages of the design view to examine your design and determine what changes should be made.

The Quartus II Technology Map Viewer provides a low-level, or atom-level, technology-specific schematic representation of a design. The Technology Map Viewer includes a schematic view, and a hierarchy list, which lists the instances, primitives, pins, and nets for the entire design netlist.

For more information on using the RTL Viewer and the Technology Map Viewer, refer to "Analyzing Synthesis Results With the Netlist Viewers" and "The Technology Map Viewer" on pages 76 and 79 in Chapter 5, "Synthesis."

Using the Chip Planner for Debugging

You can use the Chip Planner in conjunction with the SignalTap II Logic Analyzer and SignalProbe debugging tools to speed up design verification and incrementally fix bugs uncovered during design verification. After you run the SignalTap II Logic Analyzer or verify signals with the SignalProbe feature, you can use the Chip Planner to view details of post-compilation placement and routing. You can also use the Resource Property Editor to make post-compilation edits to the properties and parameters of logic cell, I/O element, or PLL atoms, without requiring a full recompilation.

Chapter Fourteen

Engineering Change Management

14

Introduction

The Quartus II software allows you to make small modifications, often referred to as engineering change orders (ECO), to a design after a full compilation. These ECO changes can be made directly to the design database, rather than to the source code or the Quartus II Settings File (**.qsf**). Making the ECO change to the design database allows you to avoid running a full compilation in order to implement the change. Figure 1 shows the engineering change management design flow.

Figure 1. Engineering Change Management Design Flow
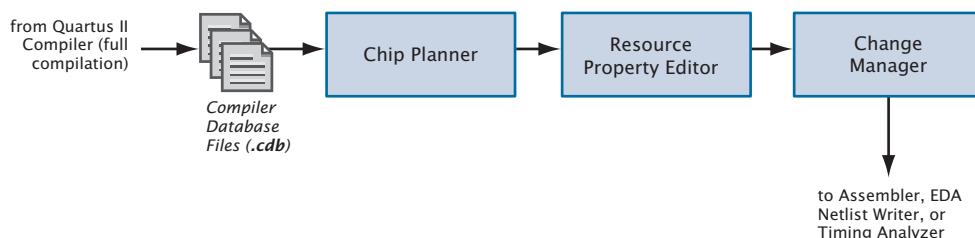
The following steps describe the design flow for engineering change management in the Quartus II software.

1. After a full compilation, use the Chip Planner to view design placement and routing details and identify which resources you want to change.

2. Create, move, and/or remove atoms in the Chip Planner.

3. Use the Resource Property Editor to edit internal properties of resources and to edit or remove connections.

4. Repeat steps 2 and 3 until you have finished making all changes.

5. View the summary and status of your changes in the Change Manager and control which changes to resource properties are implemented and/or saved. Add comments to help you reference each change.

6. Use the **Start Check & Save All Netlist Changes** command on the Processing menu to check the legality of the change for all of the other resources in the netlist.

7. Run the Assembler to generate a new programming file or run the EDA Netlist Writer to generate a new netlist.

Identifying Delays & Critical Paths With the Chip Planner

You can use the Chip Planner to view complete routing details for your design, including all possible routing paths between device resources. The Chip Planner displays all the resources of the device, such as interconnects and routing lines, logic array blocks (LABs), RAM blocks, DSP blocks, I/Os, rows, columns, and the interfaces between blocks and interconnects and other routing lines. See Figure 2.

Figure 2. Chip Planner
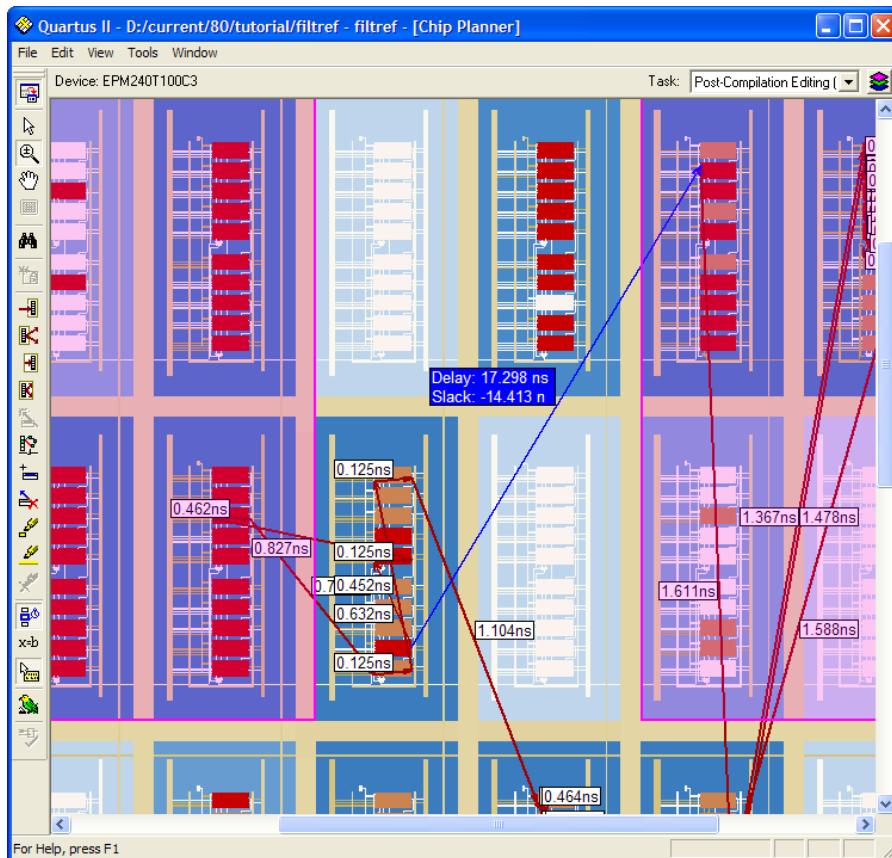
You can then use the information from the Chip Planner to determine which properties and settings you may want to edit in the Resource Property Editor. Right-click one or more resources in the Chip Planner, and then click **Locate in Resource Property Editor** to open the Resource Property Editor and make edits to the resource(s). Refer to "Modifying Resource Properties With the Resource Property Editor" on page 189 for more information.

Right-click multiple elements and click **Selected Elements Window** to locate in the Resource Property Editor or other editors to remove elements from the selection, if desired.

Editing Atoms in the Chip Planner

The Chip Planner also allows you to create new atoms, move existing atoms to other locations, or remove atoms. These changes are reflected in the Change Manager.

To create a new atom, select **Post Compilation Editing (ECO)** from the **Task** list in the Chip Planner, then right-click a resource location and click **Create Atom**. After specifying a new name for the atom, you can then right-click the atom and click **Locate in Resource Property** to modify the properties and connections for the new atom.

Modifying Resource Properties With the Resource Property Editor
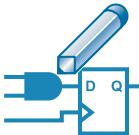
The Resource Property Editor allows you to make post-compilation edits to the properties and parameters of logic cell, I/O element, or PLL resources, as well as edit or remove connections for individual nodes. You can use the toolbar buttons to navigate forward and backward among the resources. You can also select and change multiple resources at one time. In addition, when you move the mouse pointer over a port, the Resource Property Editor highlights the fan-in and fan-out for that port.

The Resource Property Editor contains a schematic diagram of the resource you are modifying, a port connection table that lists all the input and output ports and their connected signals, and a property table that displays the properties and parameters that are available for that resource. If the port connection or property tables are not visible, you can display them with the **View Port Connections** and **View Properties** commands on the View menu. Figure 3 shows the Resource Property Editor.

Figure 3. Resource Property Editor

Viewer shows schematic diagram of resource
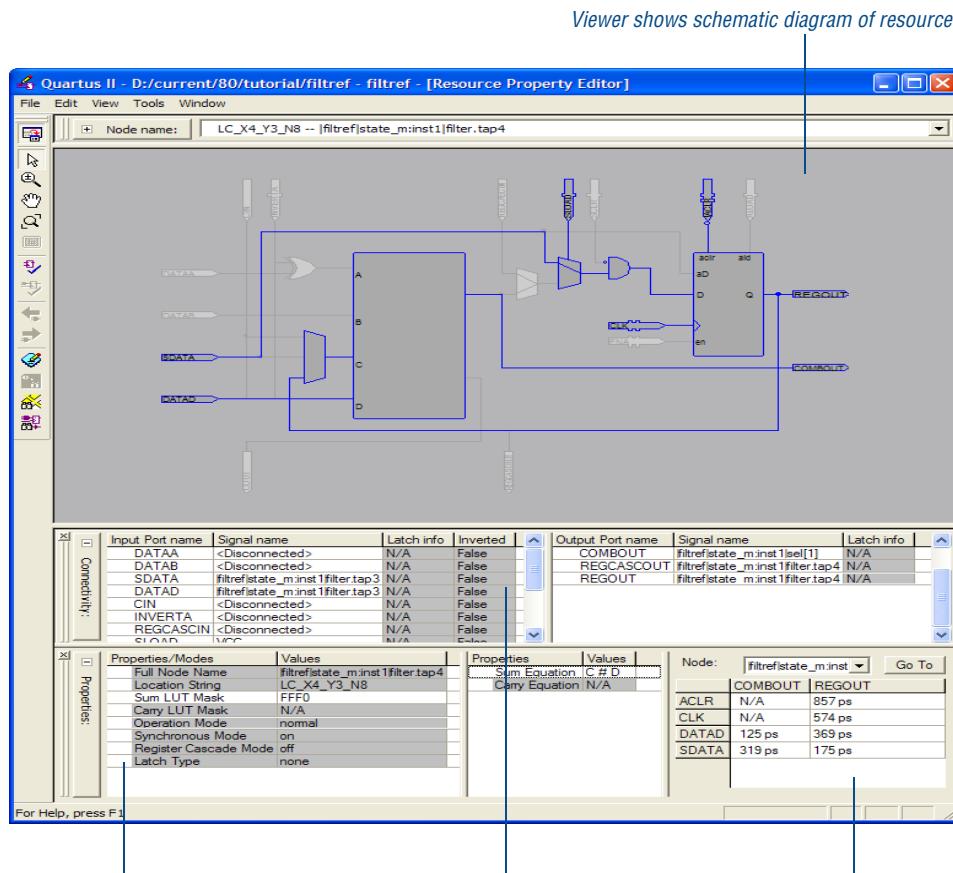
Property table displays the properties and values for the selected resource and allows you to make changes

Port connection table shows the input and output ports

Cell delay panel shows delay information for the selected node

You can make changes to the resource in the schematic, the port connection table, or the property table. If you make a change in the port connection table or property table, that change is reflected automatically in the schematic diagram. You can also view equation and cell delay information.

The Resource Property Editor allows you to right-click a node in the schematic or in the port connection table and click **Edit Connection** to specify a new signal for the connection. If you want to remove the connection, you can right-click the node and click **Remove Connection**. In the port connection table, you can create or remove output ports by

right-clicking the port and clicking **Create** or **Remove**. In the schematic, you can right-click a node and then specify one or more fan-outs to remove with the **Fan-Outs** dialog box by pointing to **Remove** and clicking **Fan-Outs**.

Once you have made a change, you can use the **Check Resource Properties** command on the Edit menu to perform simple design-rule checking on the resource. On the Processing menu point to **Start** then click **Check and Save All Netlist Changes** to save the changes you have made to atoms before you run the Assembler. You can also view a summary of your changes in the Change Manager. Refer to the next section, "Viewing & Managing Changes with the Change Manager," for more information.

For Information About	Refer To
Engineering change management and using the Resource Property Editor	*Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*
	"About the Resource Property Editor" and "About Making Post-Compilation Changes" in Quartus II Help

Viewing & Managing Changes with the Change Manager

The Change Manager window lists all the ECO changes that you have made, and allows you to select each ECO change in the list and specify whether you want to apply or delete the change. It also allows you to add comments for your reference. You can open the Change Manager by pointing to **Utility Windows** on the View menu and clicking **Change Manager**. See Figure 4.

Figure 4. Change Manager

Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
⊞ 1	state_m:inst1|filter.idle:CLK:0	Modify Source	filtref|clk	filtref|reset	filtref|reset	filtref|clk
⊞ 2	taps:inst|xn[0]:DATAD:0	Modify Source	filtref|d[0]	VCC	filtref|d[0]	filtref|d[0]
⊞ 3	taps:inst|xn[0]:DATAD:0	Modify Source	VCC	filtref|d[0]	filtref|d[0]	filtref|d[0]
⊞ 4	taps:inst|xn_1[1]:ENA:0	Modify Source	filtref|newt	filtref|clk	filtref|clk	filtref|newt

Green shading in the **Current Value** column indicates that the changes have been applied to the current value. Blue shading in the **Disk Value** column indicates that the changes have been saved successfully to disk.

After you have committed the changes you want, right-click the change and click **Check & Save All Netlist Changes** to check the legality of the change for all of the other resources in the netlist. When you choose one of the commands for exporting, you can save the exported data as a Tcl Script File (**.tcl**), which is a sequence of Chip Planner Tcl commands that can be sourced back into the Quartus II software to reproduce a set of changes if the Change Manager log has been lost or corrupted. You can also save a Comma-Separated Values File (**.csv**) or a Text File (**.txt**)—these files contain tabular representations of the data for documentation purposes.

Verifying ECO Changes

After you have made an ECO change, you should run the Assembler module of the Compiler to create a new Programmer Object File (**.pof**). You may also want to rerun the EDA Netlist Writer to generate a new netlist, or rerun timing analysis or simulation to verify that the change results in the appropriate timing improvement. Performing a full compilation, however, creates a new post-fit netlist, removing any ECO changes.

For Information About	Refer To
Engineering change management and using the Change Manager	*Engineering Change Management with the Chip Planner* chapter in volume 2 of the *Quartus II Handbook*
Using the Change Manager	"About the Change Manager" and "About Making Post-Compilation Changes" in Quartus II Help

Chapter Fifteen

Formal Verification
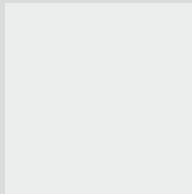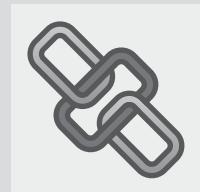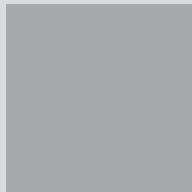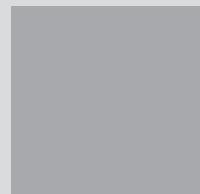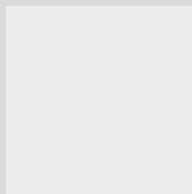
15

Introduction

The Quartus II software allows you to use formal verification EDA tools to verify the logical equivalence between source design files and Quartus II output files. Figure 1 shows the formal verification flow.

Figure 1. Formal Verification Flow
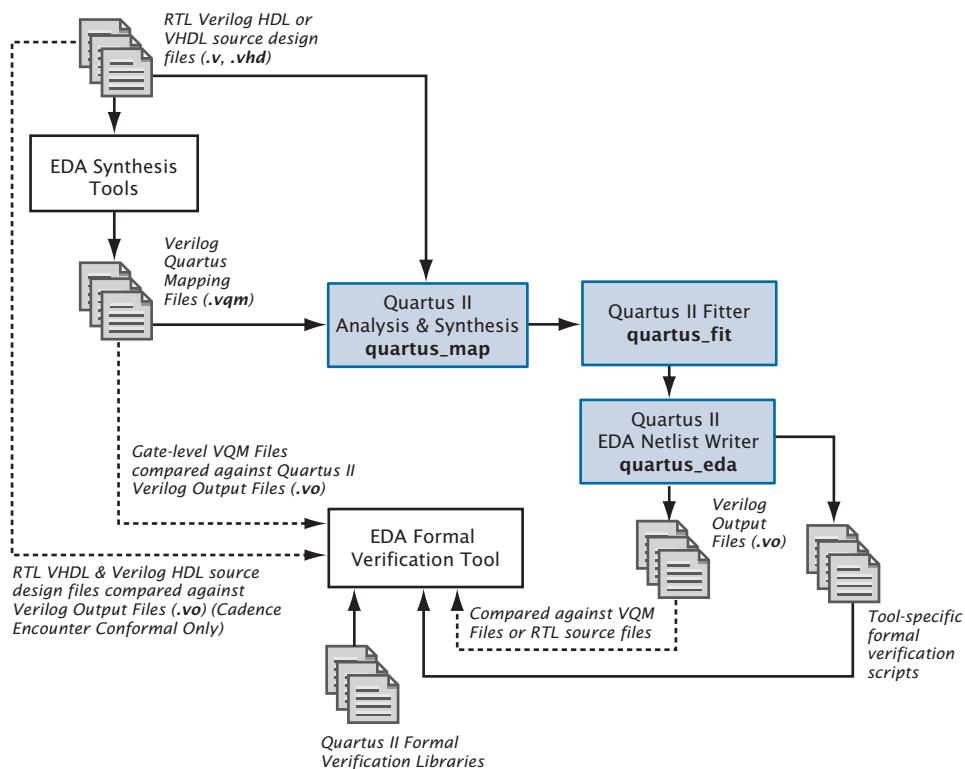
The type of formal verification supported by the Quartus II software is equivalence checking, which compares the functional equivalence of the source design with the revised design by using mathematical techniques rather than by performing simulation using test vectors. Equivalence checking greatly decreases the time to verify the design. The Quartus II software allows you to verify the logical equivalence between the synthesized gate-level Verilog Quartus Mapping Files (**.vqm**) generated by an EDA synthesis tool and the Verilog Output Files (**.vo**) generated by the Quartus II software. For the Cadence Encounter Conformal software, the Quartus II software also allows you to verify the logical equivalence

between RTL VHDL design files (**.vhd**) or Verilog HDL design files (**.v**) and Quartus II software–generated Verilog Output Files. Figure 2 shows which file types are compared in formal verification.

Figure 2. File Types Compared in Formal Verification
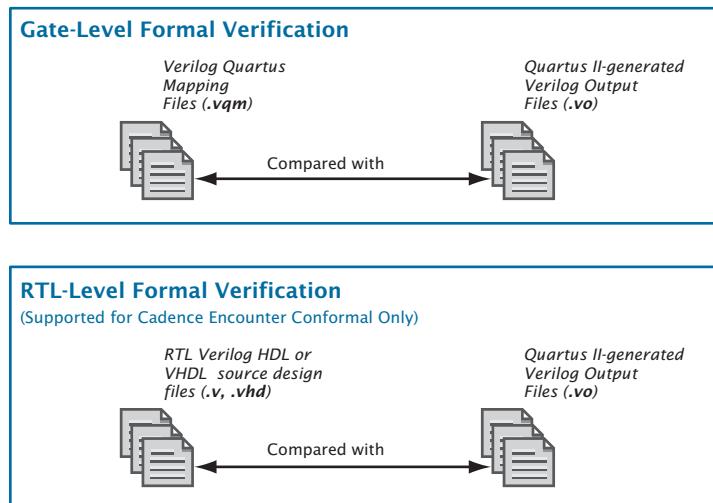
Using the Cadence Encounter Conformal Software

You can use the Cadence Encounter Conformal software to perform formal verification on your Quartus II designs. The formal verification software determines whether or not the Quartus II software correctly interprets the logic in the Verilog Quartus Mapping file or the source VHDL or Verilog HDL design file during synthesis and fitting.

In the **Formal Verification** page under **EDA Tool Settings** in the **Settings** dialog box on the Assignments menu, you can specify your formal verification tool. See Figure 3.

Figure 3. Formal Verification Page of Settings Dialog Box
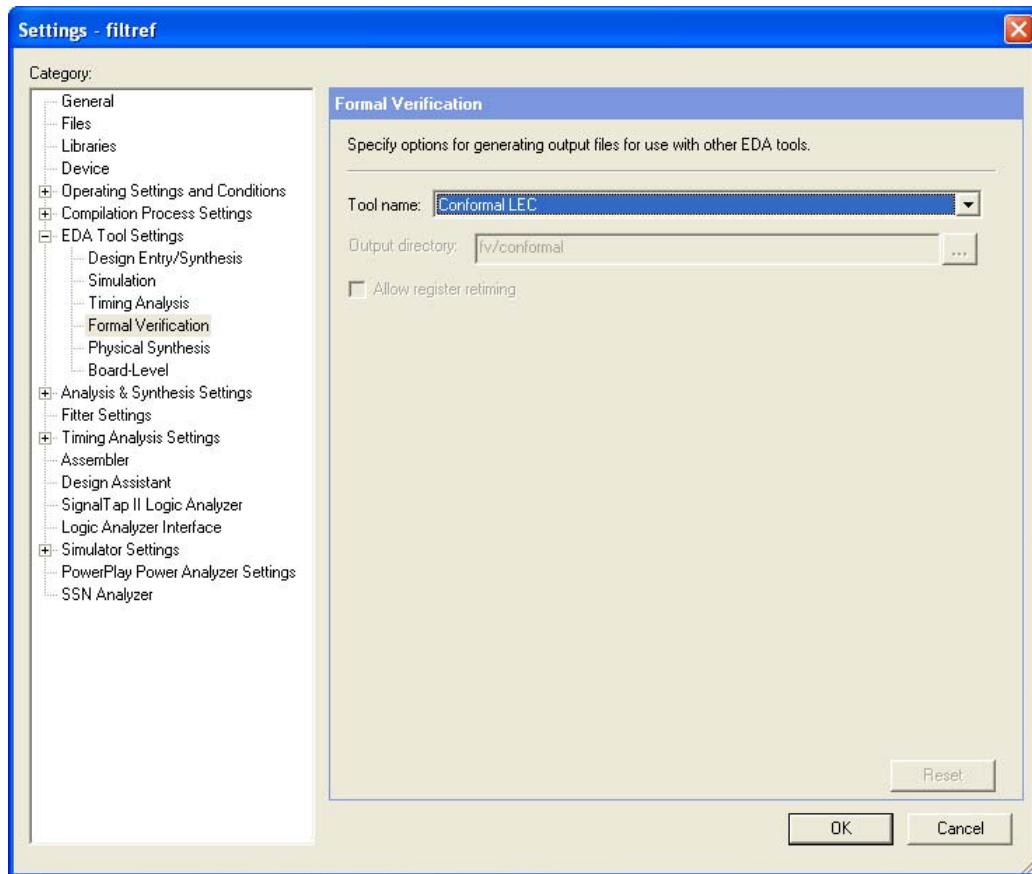
Specifying Additional Settings

When you are compiling a project to generate files for use with formal verification tools, Altera strongly recommends that you turn off the **Perform register retiming** option on the **Physical Synthesis Optimizations** page, which is under **Compilation Process Settings** in the **Settings** dialog box on the Assignments menu.

Altera recommends that you turn off the option because this option often results in moving and merging registers along the critical path, which may affect the registers in cones of logic that the formal verification tools may use as comparison points.

For Information About	Refer To
Using Cadence Encounter Conformal software	*Cadence Encounter Conformal Support* chapter in volume 3 of the *Quartus II Handbook*
	"About Using the Encounter Conformal Software with the Quartus II Software" in Quartus II Help

Chapter Sixteen

System-Level Design

16

Introduction

The Quartus II software supports the SOPC Builder and DSP Builder system-level design flows. System-level design flows allow engineers to rapidly design and evaluate system-on-a-programmable-chip (SOPC) architectures and design at a higher level of abstraction.

SOPC Builder is an automated system development tool that dramatically simplifies the task of creating high-performance SOPC designs. The tool automates the system definition and integration phases of SOPC development completely within the Quartus II software. SOPC Builder allows you to select system components, define and customize the system, and generate and verify the system before integration. Figure 1 shows the SOPC Builder design flow.

Figure 1. SOPC Builder Design Flow
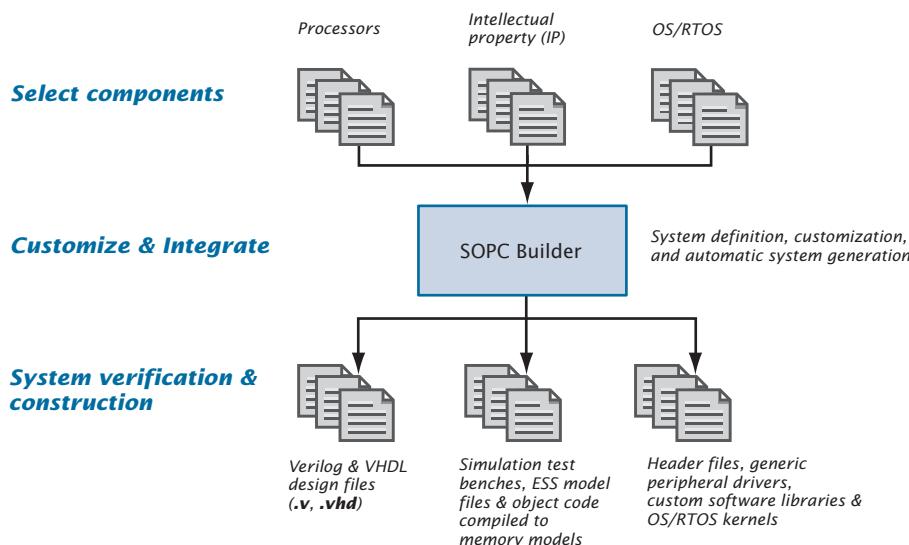
The Altera DSP Builder integrates high-level algorithm and HDL development tools by combining the algorithm development, simulation, and verification capabilities of the MathWorks MATLAB and Simulink system-level design tools with VHDL synthesis and simulation tools and the Quartus II software. Figure 2 on page 201 shows the DSP Builder design flow.

Figure 2. DSP Builder Design Flow
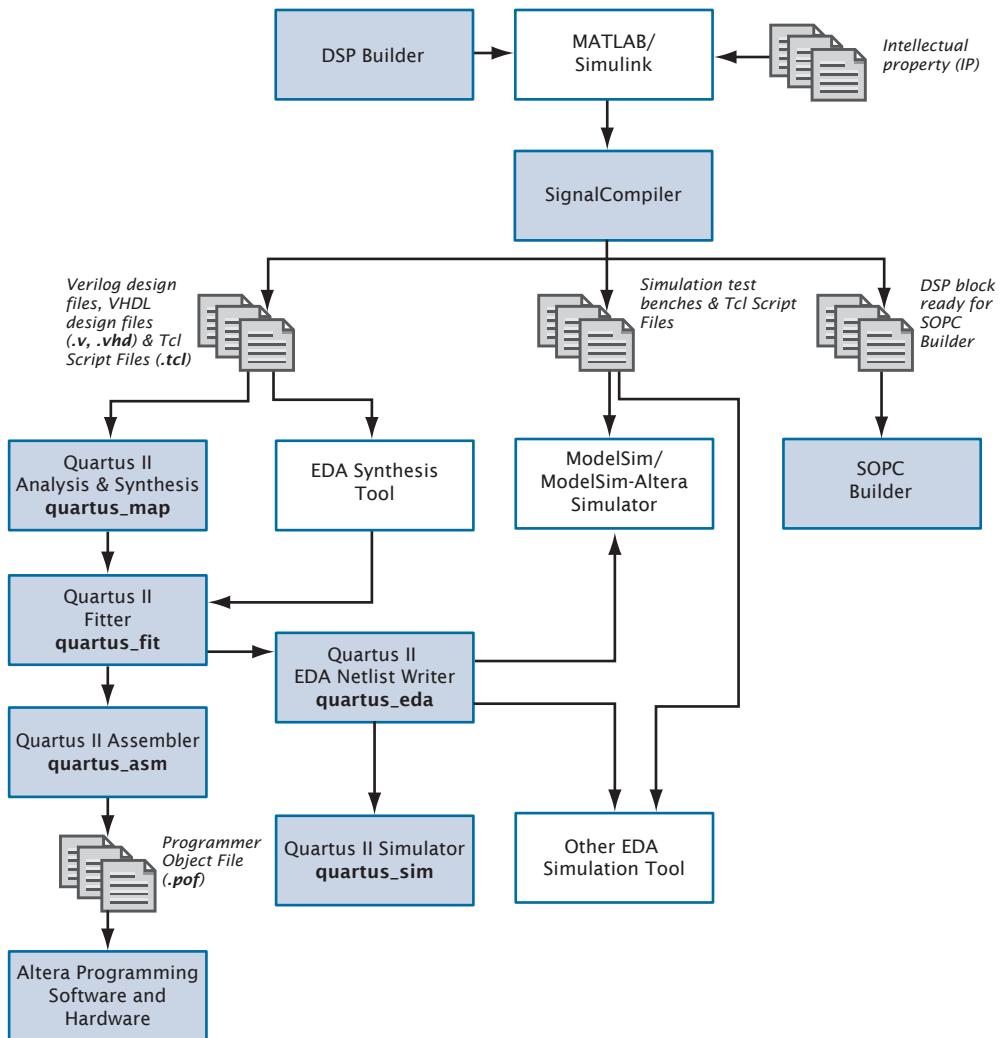
Creating SOPC Designs with SOPC Builder

SOPC Builder, which is included with the Quartus II software, provides a standardized, graphical environment for creating SOPC designs composed of components such as CPUs, memory interfaces, standard peripherals, and user-defined peripherals. SOPC Builder allows you to select and customize the individual components and interfaces of your system module. SOPC Builder combines these components and generates a single system module that instantiates these components, and automatically generates the necessary bus logic to connect them together.

The SOPC Builder library includes the following components:

- Processors
- Intellectual property (IP) and peripherals
- Memory interfaces
- Communications peripherals
- Buses and interfaces, including the Avalon™ interface
- Digital signal processing (DSP) cores
- Software
- Header files
- Generic C drivers
- Operating system (OS) kernels

You can use SOPC Builder to construct embedded microprocessor systems that include CPUs, memory interfaces, and I/O peripherals, however, you can also generate dataflow systems that do not include a CPU. It allows you to specify system topologies with multiple masters and slaves. SOPC Builder can also import or provide an interface to user-defined blocks of logic that are connected to the system as custom peripherals.

Creating the System

When building a system in SOPC Builder, you can choose either user-defined modules or modules available from the SOPC Builder library.

SOPC Builder can import or provide an interface to user-defined blocks of logic. There are four mechanisms for using an SOPC Builder system with user-defined logic: simple PIO connection, instantiation inside the system module, bus interface to external logic, and publishing a local SOPC Builder component.

SOPC Builder provides library components (modules) for download, including processors, such as the Nios® II processor, a UART, a timer, a PIO, an Avalon tri-state bridge, several simple memory interfaces, and OS/RTOS kernels. In addition, you can choose from an array of MegaCore functions, including those that support the OpenCore Plus hardware evaluation feature.

You can use the **System Contents** tab of SOPC Builder to define the system. You can select library components in the module pool and display the added components in the module table.

You can use the information in the module table of the **System Contents** tab or in a separate wizard to define the following component options:

- System components and interfaces
- Master and slave connections
- System address map
- System IRQ assignments
- Arbitration priorities for shared slaves
- Multiple master and slave clock domains

Generating the System

Each project and component in SOPC Builder contains a system description file that contains all the settings, options, and parameters entered in SOPC Builder. During system generation, the SOPC Builder uses these files to generate the source code, software components, and simulation files for the system.

Once system definition is complete, you can generate the system using the **System Generation** tab of SOPC Builder.

The SOPC Builder software automatically generates all necessary logic to integrate processors, peripherals, memories, buses, arbitrators, IP functions, and interfaces to logic and memory outside the system across multiple clock domains; and creates HDL source code that binds the components together.

SOPC Builder can also create software development kit (SDK) software components, such as header files, generic peripheral drivers, custom software libraries, and OS/real-time operating system (RTOS kernels), to provide a complete design environment when the system is generated.

For simulation, SOPC Builder creates a Mentor Graphics ModelSim simulation directory that contains a ModelSim project file, the simulation data files for all memory components, macro files to provide setup information, aliases, and an initial set of bus-interface waveforms. It also creates a simulation test bench that instantiates the system module, drives clock and reset inputs, and instantiates and connects simulation models.

A Tcl script that sets up all the files necessary for compilation of the system in the Quartus II software is also generated.

For Information About	Refer To
Using SOPC Builder	*Introduction to SOPC Builder* chapter in volume 4 of the *Quartus II Handbook*
	"About SOPC Builder" in Quartus II Help

Creating DSP Designs with the DSP Builder

DSP Builder shortens DSP design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment. DSP Builder allows system, algorithm, and hardware designers to share a common development platform. The DSP Builder is an optional software package available from Altera, and is also included with DSP Development Kits.

DSP Builder also provides support for system-level debugging using the SignalTap II block or the Hardware in the Loop (HIL) block. You can synthesize, compile and download the design, and then perform debugging, all through the MATLAB/Simulink interface. Adding the Hardware in the Loop block to your Simulink model allows you to co-simulate a Quartus II software design with a physical FPGA board implementing a portion of that design. You define the contents and function of the FPGA by creating and compiling a Quartus II project. A simple JTAG interface between Simulink and the FPGA board links the two.

Instantiating Functions

You can combine existing MATLAB functions and Simulink blocks with Altera DSP Builder blocks and MegaCore functions, including those that support the OpenCore Plus hardware evaluation feature, to link system-level design and implementation with DSP algorithm development.

Generating Simulation Files

After verifying the design in the Simulink software, you can use the DSP Builder SignalCompiler block to generate files for simulating the design in EDA simulation tools.

The SignalCompiler block translates a DSP Builder Simulink model into a VHDL or Verilog model and generates a Verilog HDL or VHDL test bench file that imports the Simulink input stimuli. You can use the Tcl script for automated simulation in the ModelSim software, or simulate in another EDA simulation tool with the Verilog HDL or VHDL test bench file.

Generating Files for Synthesis

DSP Builder provides automated and manual synthesis and compilation flows. You can use the Quartus II software to synthesize the design, or you can use the Tcl script generated by the DSP Builder SignalCompiler block to synthesize the design in Mentor Graphics Leonardo Spectrum or Synplicity Synplify software. If the DSP Builder design is the top-level design, you can use either the automated or manual synthesis flow. If the DSP Builder design is not the top-level design, you must use the manual synthesis flow.

You can use the automated flow to control the entire synthesis and compilation flow from within the MATLAB/Simulink design environment. The SignalCompiler block creates VHDL Design Files and Tcl scripts, performs synthesis in the Quartus II, LeonardoSpectrum, or Synplify software, compiles the design in the Quartus II software, and can also optionally download the design to a DSP development board. You can specify which synthesis tool to use for the design from within the Simulink software.

In the manual flow, the SignalCompiler block generates VHDL Design Files and Tcl scripts that you can then use to perform synthesis manually in an EDA synthesis tool, or the Quartus II software, which allows you to specify your own synthesis or compilation settings. When generating output files, the SignalCompiler block maps each Altera DSP Builder block to the VHDL library. MegaCore functions are treated as black boxes.

For Information About	Refer To
Using the DSP Builder	*DSP Builder User Guide* on the Altera website

Chapter
Seventeen

Installation, Licensing & Technical Support
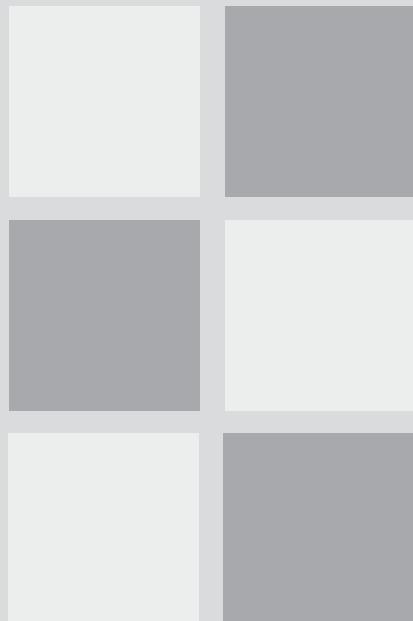
17

Installing the Quartus II Software

You can install the Quartus II software on the following platforms:

■ Pentium III (400 MHz or faster) based computer, running one of the following Windows operating systems:

– Microsoft Windows XP
– Microsoft Windows Vista

■ Opteron (AMD) or EM64T PC, running Microsoft Windows XP Professional x64 Edition

■ Pentium III (400 MHz or faster), Pentium 4 (400 MHz or faster), or AMD64/EM64T based computer running one of the following Linux operating systems:

– Red Hat Enterprise Linux 4.0
– Red Hat Enterprise Linux 5.0
– CentOS-4
– CentOS-5
– SUSE Linux Enterprise Server 9
– SUSE Linux Enterprise Server 10

For Information About	Refer To
System requirements and installation instructions	*Altera Software Installation and Licensing* manual on the Altera website
Specific information about disk space and memory	Quartus II software **readme.txt** file
Latest information on new features, EDA interface support, and known issues and workarounds for the Quartus II software	*Quartus II Software Release Notes* on the Altera website
Latest information about device support in the Quartus II software	*Quartus II Device Support Release Notes* on the Altera website

Licensing the Quartus II Software

To use Altera-provided software, you need to obtain and set up an Altera subscription license. An Altera subscription enables the following software:

- Altera Quartus II software (Includes SOPC Builder and IP Library)
- Mentor Graphics ModelSim-Altera software

Altera offers several types of software subscriptions. Table 1 shows the different license and subscription options that are available.

Table 1. Altera License and Subscription Options

License Name	Description
FIXEDPC	A stand-alone PC license tied to your Network Interface Card (NIC) number or Quartus II serial number.
FLOATALL	A floating network license for Windows, Red Hat, or SUSE Linux Enterprise Server 9 users with either a Windows, UNIX, or Linux license server. If you choose a FLOATALL license, you receive a license that allows you to enable the software on the platform of your choice.
Quartus II Web Edition	A free, entry-level version of the Quartus II software that supports selected devices. The Quartus II Web Edition software is available from the Altera website at **www.altera.com**. The Quartus II Web Edition version 8.1, ModelSim-Altera Starter Edition version 6.4a, and later versions do not require a license.

Customers who purchase selected development kits receive a free version of the Quartus II software for Windows and are given instructions on how to obtain a license for the software.

If you are using a node-locked (single-user) (FIXEDPC) version of the Quartus II software for the first time, you must use your Quartus II software serial number to obtain a license file.

The following steps describe the basic flow for licensing your software:

1. When you start the Quartus II software, if the software cannot detect a valid ASCII text license file, **license.dat**, you see a prompt with the following options:

   – **Start the 30-day evaluation period with no license file (no device programming file support).** This option allows you to evaluate the Quartus II software, without programming file support, for 30 days. After the 30-day grace period is over, you must obtain a valid license file from the Licensing section of the Altera website at **www.altera.com/licensing**, and then follow the remaining steps in this procedure.

   – **Perform automatic web license retrieval.** If you are using a node-locked (FIXEDPC) license, this option allows the Quartus II software is able to automatically retrieve a license file from the website and you can skip the remaining steps of this procedure. If you are using a network (multiuser) license, or if the Quartus II software is not able to retrieve a license file, you are guided through the licensing procedure.

   – **If you have a valid license file, specify the location of your license file.** If you have a valid license file but have not specified the location of the license file, selecting this option displays the **License Setup** page of the **Options** dialog box on the Tools menu. Choose **Specify valid license file** or **Use LM_LICENSE_FILE variable**. You can also specify the license file or LM_LICENSE_FILE variable in your **System** control panel for Windows XP, or Windows Vista (32-bit and 64-bit), or in your **.cshrc** file for Linux workstations. If you select this option, you can skip the remaining steps of the procedure.

2. If you are requesting a new license file, in the Licensing section of the Altera website, choose the link for the appropriate license type. Refer to .

3. Specify the requested information.

4. After you receive a license file by e-mail, save it to a directory on your system.

5. Modify the license file for your license.

6. Set up and configure the FLEXlm license manager server for your system.

For Information About	Refer To
Detailed information about licensing the Quartus II software, modifying the license file, and specifying the license file location	*Altera Software Installation and Licensing* manual on the Altera website
General information about Quartus II licensing	"Specifying a License File" in Quartus II Help

Getting Technical Support

The easiest way to get technical support is to use the mySupport website and register for an altera.com account and user name. Your copy of the Quartus II software is registered at the time of purchase; however, in order to use the mySupport website to view and submit service requests, you must also register for an altera.com account and user name. An altera.com account also makes it easier for you to use many other Altera website features, such as the Download Center, Licensing Center, Altera Technical Training online class registration, or Buy On-Line-Altera eStore features.

To register for an Altera.com account user name and password, follow these steps:

1. Go to the mySupport website:

   ✓ To start your web browser and connect to the mySupport website while running the Quartus II software, on the Help menu point to **Altera on the Web** and click **Quartus II Home Page**.

   or

   ✓ Point your web browser to the mySupport website at **www.altera.com/mysupport**.

2. Follow the instructions on the mySupport website to register for an Altera.com account.

If you are not a current Altera subscription user, you can still register for an Altera.com account.

For information about other technical support resources, refer to Table 2.

Table 2. Quartus II Technical Support Resources

Resource	Description
Altera website	**www.altera.com** The Altera website provides information on Altera and all of its products.
Support Center	**www.altera.com/support** The Support Center section of the Altera website gives you access to the mySupport website. In addition, it provides software and device support information as well as design examples that you can integrate into your design.
mySupport website	**www.altera.com/mysupport** The mySupport website allows you to submit, view, and update technical support service requests.
Telephone	(800) 800-EPLD (7:00 a.m. to 5:00 p.m. Pacific time, M–F) You will need your 6-digit Altera ID to access the hotline. (408) 544-8767 (7:00 a.m. to 5:00 p.m. Pacific time, M–F)

Chapter Eighteen

Documentation & Other Resources

18

Getting Online Help

The Quartus II software includes a browser-based Help system that provides comprehensive documentation for the Quartus II software and more details about the specific messages generated by the Quartus II software. You can view Help in one of the following ways:

■ Press F1 from a highlighted menu command or active dialog box for context-sensitive help.
■ Click the **Help** button when available in an active dialog box.
■ On the Help menu, click **Search**.

To print Help topics from the Contents tab, right-click the Help folder or individual Help topic that you want to print, and click Print or click the Print button on the toolbar. If you select a Help folder to print, you can choose to print all the topics in the folder. You can also use the Print command or Print button to print any individual Help topic you are viewing.

To search for a keyword in an open Quartus II Help topic, press Ctrl+F to open the **Find** dialog box, and type the search text, and then click **Find Next**.

For Information About	Refer To
Using Quartus II Help	"Using Quartus II Help Effectively" and "Help Menu Commands" in Quartus II Help
	"Using Quartus II Help" in the *Altera Software Installation and Licensing* manual.

Starting the Quartus II Interactive Tutorial

The Quartus II software includes the Flash-based Quartus II Interactive Tutorial. The modules of this tutorial teach you how to use the basic features of the Quartus II design software, including design entry, compilation, timing analysis, simulation, programming, incremental compilation, and the SignalTap II Logic Analyzer.

This tutorial includes audio and Flash animation components. For best results, use the tutorial on a system that includes a sound card, speakers, and at least 1024x768 display resolution.

To start the Quartus II Interactive Tutorial after you have successfully installed the Quartus II software:

✓ On the Help menu, click **Getting Started Tutorial**.

Once you start the tutorial, you can jump immediately to any tutorial module by clicking **Contents**. Once you select a tutorial module, you can click **Show Me**, **Guide Me**, or **Test Me** at any time to jump directly to the tutorial mode that best suits your learning style.

Other Quartus II Software Documentation

Table 1 shows the additional software documentation that is available for the Quartus II software:

Table 1. Additional Quartus II Documentation (Part 1 of 2)

Document	Description	Where to Find It
Quartus II Software Release Notes	Provides late-breaking information about new features, EDA interface support, and known issues and workarounds	The Altera website
Quartus II Device Support Release Notes	Provides information about changes to device support, including changes to timing, simulation, and power models	The Altera website
Altera Software Installation and Licensing manual	Provides detailed information about software requirements, installation, and licensing for Windows and Linux workstations	The Altera website
Quartus II Handbook	Provides comprehensive information about the programmable logic design cycle from design to verification	In Quartus II subscription packages and on the Altera website
Altera Complete Design Suite **readme.txt** file	Provides information about memory, disk space, and system requirements	On the Altera Complete Design Suite DVD and installed with the Quartus II software
Quartus II Scripting Reference Manual	Provides information about command-line and Tcl commands and scripting	The Altera website

Table 1. Additional Quartus II Documentation (Part 2 of 2)

Document	Description	Where to Find It
Quartus II Settings File Reference Manual	Provides information about Quartus II Settings File variables	The Altera website
Quartus II Software Quick Start Guide	Shows how to set up your project, set timing requirements, and compile your project for a target device	In Quartus II subscription packages and on the Altera website

Other Altera Literature

The Literature section of the Altera website at **www.altera.com** provides documentation on many subjects that are related to the Quartus II software, including the following topics:

■ Quartus II features and guidelines on using these features with your design flow
■ Altera device features, functions, structure, specifications, configuration, and pin-outs
■ Design solutions and methodologies
■ Implementing device features
■ Altera programming hardware features, use, and installation
■ Using the Quartus II software with other EDA tools
■ Using other Altera software tools
■ Implementing IP MegaCore functions and Altera megafunctions
■ Optimizing designs or improving performance
■ Synthesis, simulation, and verification guidelines
■ Product updates and notifications

The literature that is available from the Altera website is the most current information about Altera products and features; it is updated frequently, even after a product has been released. Altera continues to add new literature in order to provide more information on the latest features of Altera tools and devices, and to provide additional information that Altera customers have requested.

Index

A

Add Paths dialog box 146
AHDL 27
AHDL Include Files (**.inc**) 25
Altera Hardware Description Language
      (AHDL) 27
Altera Megafunction Partners Program
      (AMPP) 30
Altera on the Web command 211
Altera Programming Unit (APU) 160
Altera website 212
Altera.com account 212
AMPP 30
Analysis & Elaboration 64, 76
Analysis & Synthesis 3
    design flow 64
    Integrated Synthesis 65
    netlist optimization 71
    performing with EDA tools 68
    VHDL and Verilog HDL support 65
Analysis & Synthesis Settings page 72,
      142
APU 160
Assembler 3, 158, 159, 160
Assign SignalProbe Pins dialog box 178
Assignment Editor 53
Assignment Editor command 93
assignments
    importing 60
    location 93
    making 52, 140
    path-based 146
    viewing 139
attributes 71
Avalon interface 202

B

batch files 43
black-box methodology 34

Block Design Files (**.bdf**) 24, 25
Block Editor 25
Block Symbol Files (**.bsf**) 25, 27
block-based design 13
Board-Level page 133
ByteBlaster II download cable 160, 169
ByteBlasterMV download cable 160, 169

C

Chain Description Files (**.cdf**) 160
change management design flow 186
Check Resource Properties command 191
Chip Editor 189
Chip Planner 6, 92, 139, 184, 187
clear box methodology 35
command-line executables 38
Comma-Separated Value Files (**.csv**) 53,
      175
compilation flows 4, 42
compilation, incremental 12, 57, 105, 174
Compiler
    compilation flows 4, 42
    modules 3
    specifying settings 56
    starting 3
    status 85
Compiler Database Interface 3
compiler directives 71
configuring 158
Constraints
    timing 126
Convert Programming Files command 159
Copy Project command 18
Create command 190
**Create/Update > Create Jam, SVF, or ISC
      File** command 159
Create/Update > Create/Update IPS File
      command 160
Create/Update command 26, 27

D

debugging *see* SignalTap II Logic Analyzer;
        SignalProbe feature
Design Assistant 3, 74, 92
Design Assistant page 74
design constraints 52
Design Partition Planner 58
design partitions 57, 58, 105
Design Partitions window 57, 59
Design Space Explorer 97, 147
devices, programming and configuring 158
documentation conventions xi
DSE 97, 147
dse.tcl Tcl script 97, 147
DSP Builder 200, 205
        creating designs 205
        design flow 201
        generating simulation files 205
        generating synthesis files 206
        instantiating functions 205
        SignalCompiler block 205
        using with other EDA tools 206

E

Early Timing Estimate page 129
early timing estimate, performing 129
ECOs 186
        creating 189
        verifying 192
EDA interfaces 7
EDA Netlist Writer 3, 111, 113, 134
EDA Tool Settings page 8, 69, 195
EDA tools 133
        formal verification 195
        functional simulation 114
        simulation 111
        specifying settings 8, 56, 69, 112, 195
        starting synthesis tools 70
        supported tools 8, 68, 111, 133
        synthesis 68
        timing analysis 133
        timing simulation 115
EDIF Input Files (**.edf**) 24

EDIF netlist files (**.edf**) 64, 68
Edit Connection command 190
engineering change orders *see* ECOs
EthernetBlaster download cable 160
executables 38
Export Database command 23

F

Files page 18
Fitter 3, 84
Fitter Settings page 93
fitting
        analyzing 87
        design flow 84
        optimization 92, 143
        performing an early timing
                estimate 129
Flash Loader Hexadecimal File (**.flhex**) 159
FLEX Chain File (**.fcf**) 160
flows for compilation 4, 42
formal verification
        design flow 194
        performing with EDA tools 195
        specifying settings 197
full compilation 3
functional simulation
        EDA tools 114
        Quartus II Simulator 117

G

Graphic Editor *see* Block Editor
graphical user interface 3

H

Help, getting 214
Hexadecimal (Intel-Format) Files (**.hex**) 163
Hexadecimal (Intel-Format) Output Files
        (**.hexout**) 159, 162
Hierarchy Display *see* Project Navigator

Altera Corporation
101 Innovation Drive
San Jose, California 95134 USA
www.altera.com
www.altera.com/mysupport