Introduction to the Quartus® II Software

Version 10.0

QUARTUS® II

Introduction to the
Quartus® II
Software

QUARTUS®II

Contents

Preface

This manual is designed for the novice Altera® Quartus® II design software user and provides an overview of the capabilities of the Quartus II software in programmable logic design. The Altera Quartus II software is the most comprehensive environment available for system-on-a-programmable-chip (SOPC) 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, the Quartus II interactive tutorial, application notes, 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.

Chapter One

Design Flow
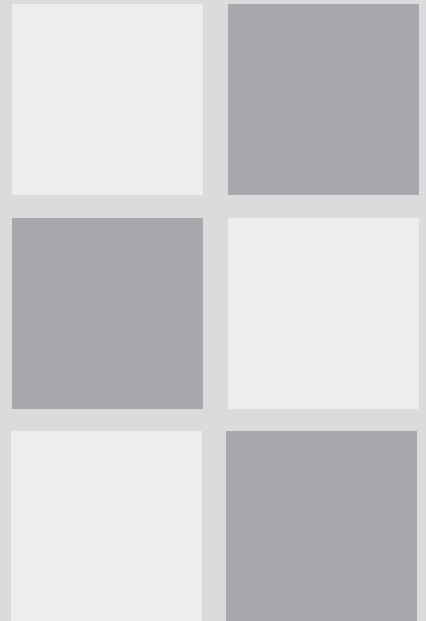
1

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

Design Entry — *Includes block-based design, system-level design & software development*

Synthesis → Power Analysis

Place & Route → Debugging

Timing Analysis → Engineering Change Management

Simulation → Timing Closure

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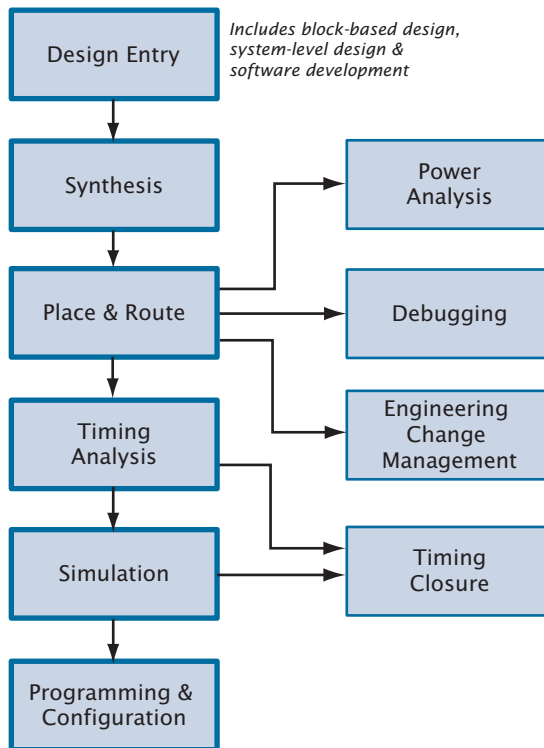
In addition, the Quartus II software allows you to use the Quartus II graphical user interface and command-line interface for each phase of the design flow. You can use one of these interfaces for the entire flow, or you can use different options at different phases.

Graphical User Interface Design Flow

You can use the Quartus II software graphical user interface (GUI) 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
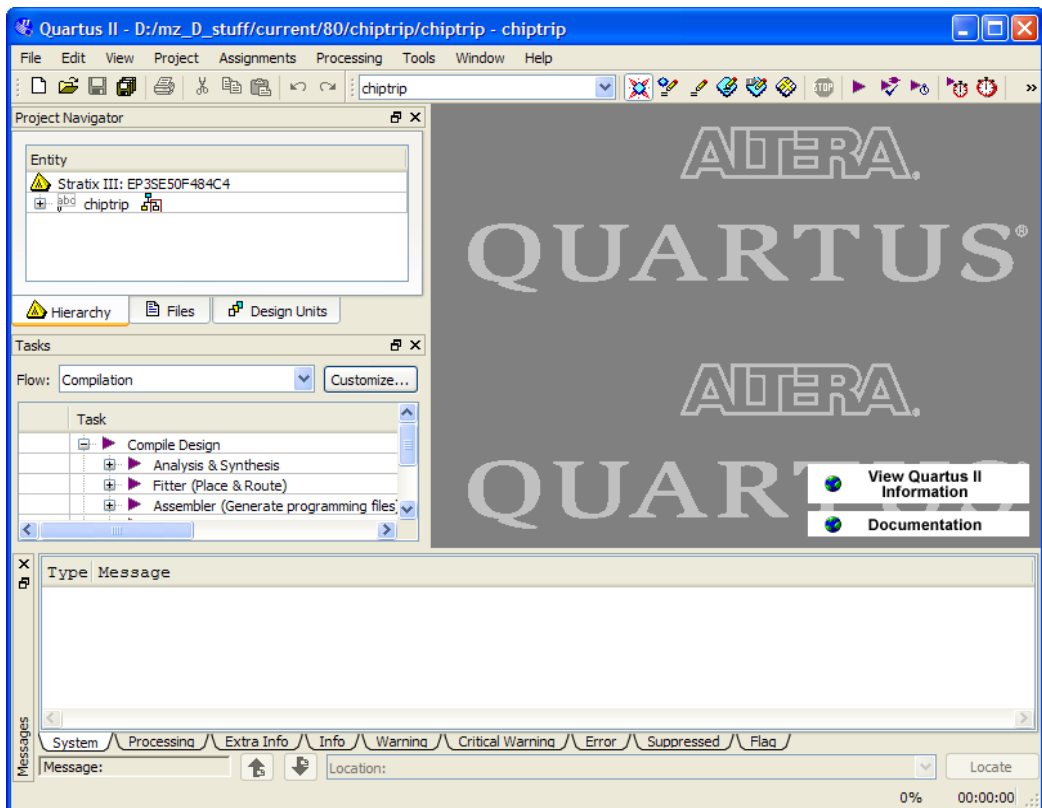
The Quartus II software includes a modular Compiler. The Compiler includes the following modules (modules marked with an asterisk are optional during a compilation, depending on your settings):

- 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.

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

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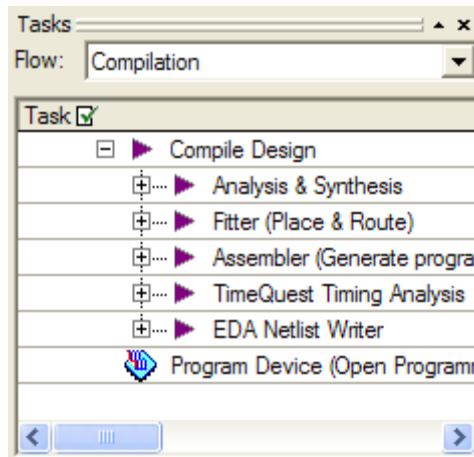
The Quartus II software also provides other predefined compilation flows that you can use with commands on the Processing menu. Table 1 lists the commands for some of the most common compilation flows.

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
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 GUI:

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 **Device** dialog box, the Chip Planner, 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 an EDA simulation tool.

10. Place and route the design with the Fitter.

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

12. Use an EDA simulation tool 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.

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 running. You can use these executables with standard command-line commands and scripts, with Tcl scripts, and in makefiles. See Table 2 for a list of all available command-line executables.

Figure 4. Command-Line Design Flow
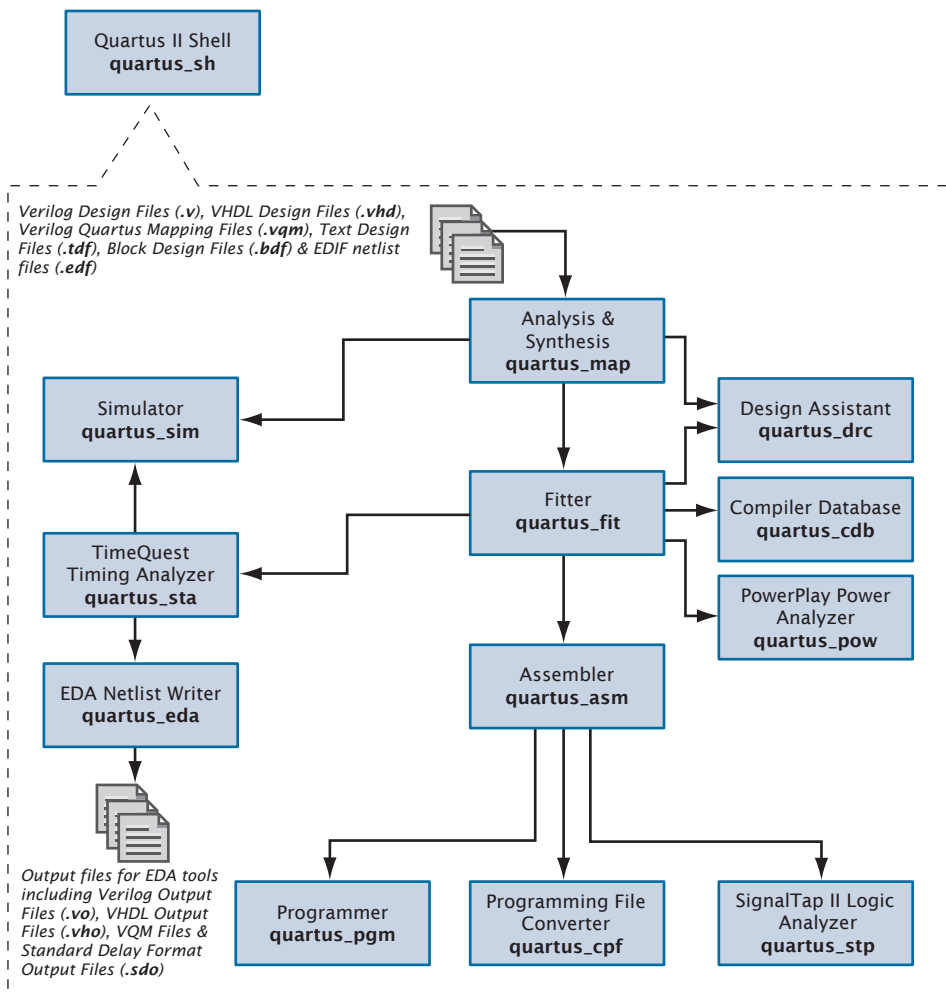
Table 2. Command-Line Executables (Part 1 of 2)

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.
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.

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

Executable Name	Title	Function
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.
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.

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 5 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 5 from a command prompt in the new project directory, or you can store them in a batch file or shell script.

Figure 5. 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 6 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 6. 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
```

The Quartus II software also supports makefile scripts that use the Quartus II executables, which allow you to integrate your scripts with a wide variety of scripting languages.

For Information About	Refer To
Using command-line executables	About Quartus II Scripting
	Command-Line Scripting chapter in volume 2 of the *Quartus II Handbook*
Using compilation flows	"About Compilation Flows" in Quartus II Help

Using Tcl Commands

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 **Insert 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 7 shows an example of a Tcl script.

Figure 7. Example of a Tcl Script

```
## 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 CYCLONE

# Set Device
set_global_assignment -name device ep1c6f256c6

# 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

# 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

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 incremental compilation design flows and block-based design flows. You can use these design flows with or without EDA design entry and synthesis tools.

Incremental Design Flows

Your design flow affects how much impact design partitions have on design optimization, and how much design planning may be required to obtain optimal results. In the standard incremental compilation flow, the design is divided into partitions, which can be compiled and optimized together as parts of one Quartus II project. If another team member or IP provider is developing source code for the design, they can functionally verify their partition independently, and then simply provide source code for the partition to the project lead for integration into the larger design. If the project lead wants to compile the larger design when source code is not yet complete for a partition, they can create an empty placeholder for the partition to facilitate compilation until the actual partition code is ready.

Compiling all design partitions in a single Quartus II project ensures that all design logic is compiled with a consistent set of assignments and allows the software to perform global placement and routing optimizations. Compiling all design logic together is beneficial for FPGA design flows because in the end all parts of the design must use the same shared set of device resources.

If required for third-party IP delivery, or in cases where designers can not access a shared or copied top-level project framework, you can create and compile a design partition logic in isolation and export a partition that is included in the top-level project. If this type of design flow is necessary, planning and rigorous design guidelines may be required to ensure that designers have a consistent view of project assignments and resource allocations. Therefore developing partitions in completely separate Quartus II projects can be more challenging than having all source code within one project or developing design partitions within the same top-level project framework.

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

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.

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.

You can also use the **LogicLock Regions Properties** dialog box to edit existing LogicLock regions, view information about 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.

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 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.

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 "Using Incremental Compilation" on page 53 in Chapter 4, "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

Chapter Two

Design Entry
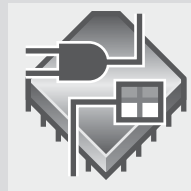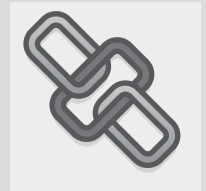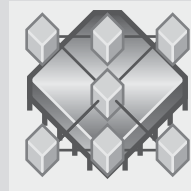
2

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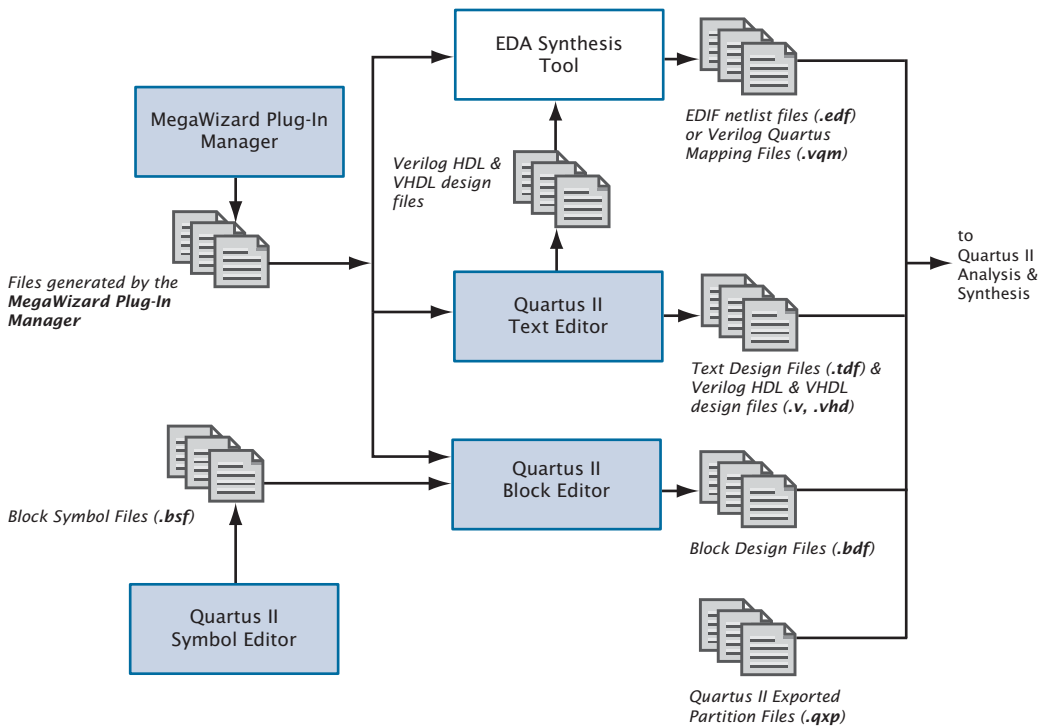
The Quartus II software also supports system-level design entry flows with the Altera SOPC Builder and DSP Builder software. For more information about these methods, refer to "Chapter 16: System-Level Design" on page 199.

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.

The Project Navigator provides a graphical representation of the project hierarchy, files, and design units, and shortcuts to various menu commands.

Figure 2. Project Navigator Window
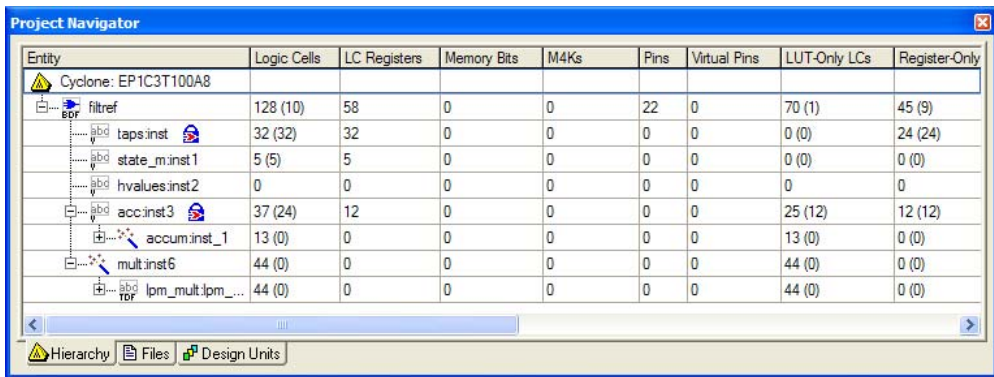
The Project Navigator also allows you to assign design partitions. For more information, see "Creating Design Partitions" on page 57.

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

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 "EDA Synthesis Tools" on page 108.

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.

Using the Quartus II Symbol Editor

The Quartus II 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. You can customize these Block Symbol Files and then add the symbols to schematics created with the Block Editor.

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 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 41 in Chapter 3, "Synthesis."

AHDL is a high-level, modular language that is completely integrated into the Quartus II software. 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 1 shows the types of Altera-provided megafunctions and LPM functions that you can create with the **MegaWizard Plug-In Manager**.

Table 1. 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 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.

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, 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

For Information About	Refer To
Using the **MegaWizard Plug-In Manager** and Altera-provided megafunctions and LPM functions	"About the MegaWizard Plug-In Manager" in Quartus II Help
	Command-Line Scripting chapter in volume 2 of the *Quartus II Handbook*
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*

Constraint Entry

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 3 shows the constraint and assignment entry flow.

Figure 3. Constraint & Assignment Entry Flow
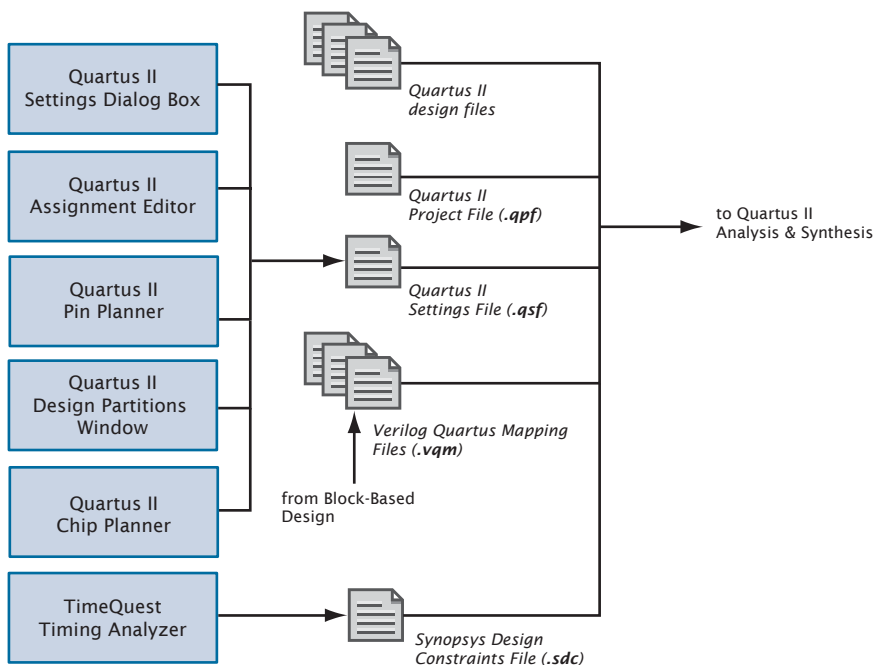
Using the Assignment Editor

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 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.

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.

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 4 shows the Pin Planner.

Figure 4. Pin Planner
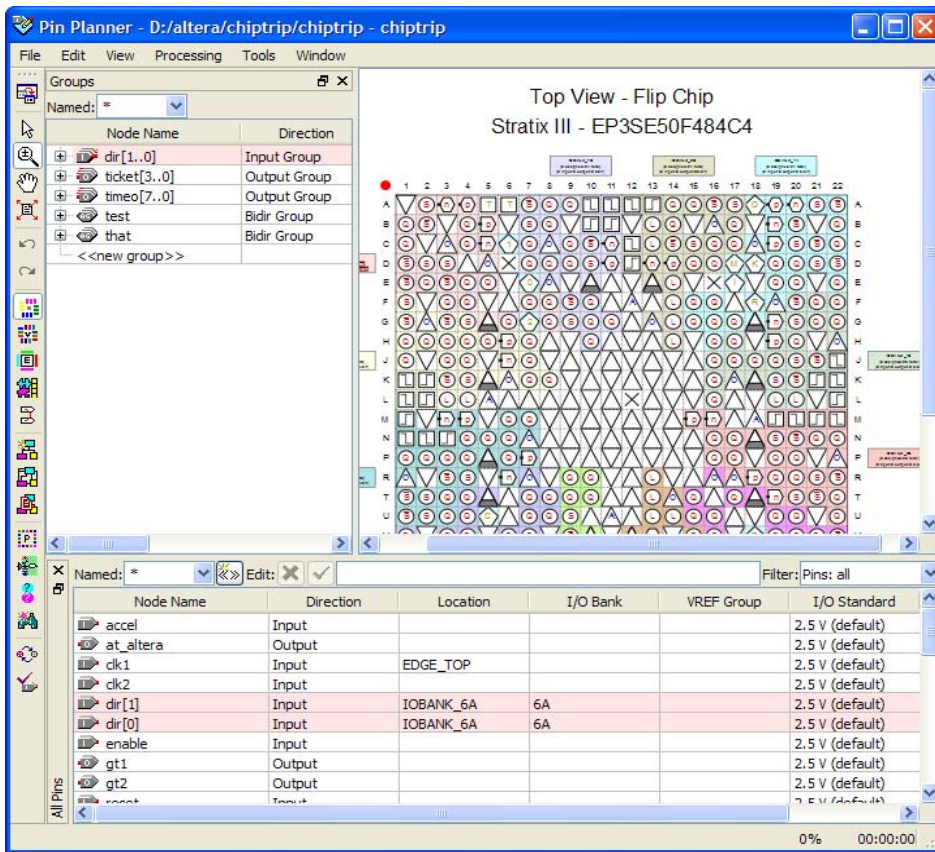
By default, the Pin Planner displays a **Groups** list, an **All Pins** list, and a package view diagram of the device. You can make pin assignments by dragging pins from the **Groups** list and **All Pins** list to available pin or I/O bank locations in the package diagram. In the **All Pins** list, 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.

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 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 5: Timing Analysis and Design Optimization" on page 65.

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

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**.

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

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 **Design Partition 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.

Creating Design Partitions with the Design Partitions 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 5. Design Partition Planner
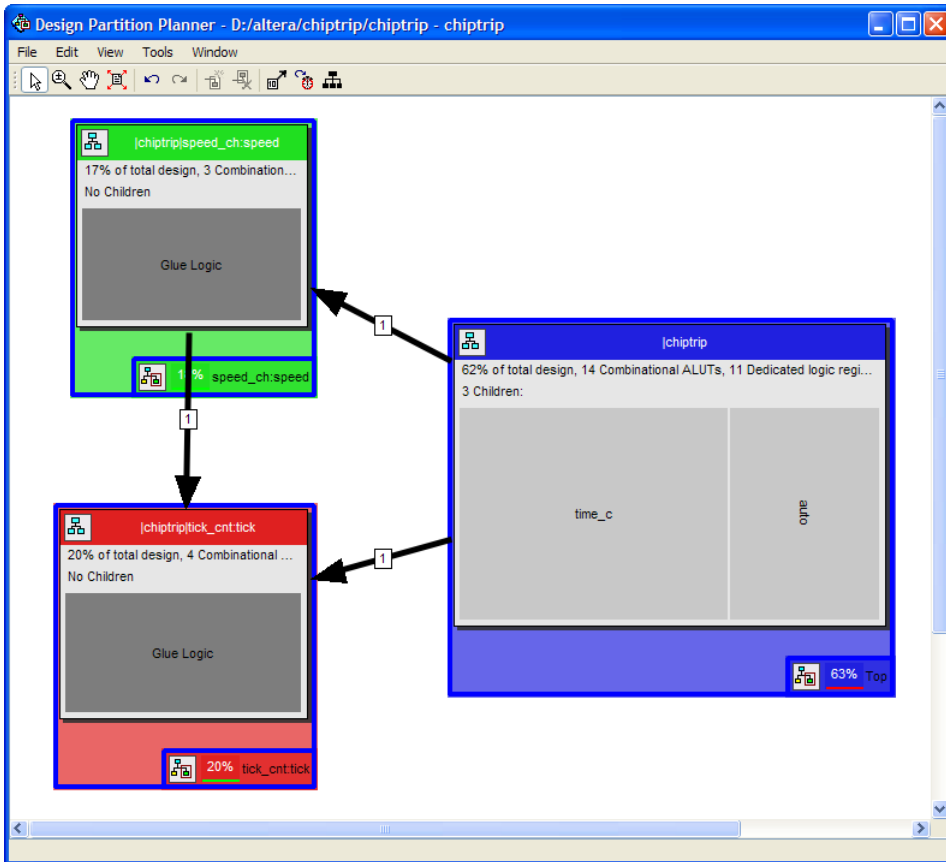
For Information About	Refer To
Assigning design partitions and using incremental compilation	*Quartus II Incremental Compilation for Hierarchical & Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*
	Best Practices for Incremental Compilation Partitions and Floorplan Assignments chapter in volume 1 of the *Quartus II Handbook*
	"About Incremental Compilation" in Quartus II Help

Chapter
Three

Synthesis
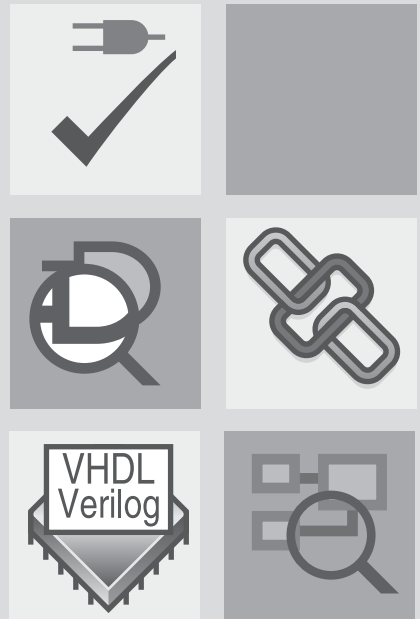
3

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
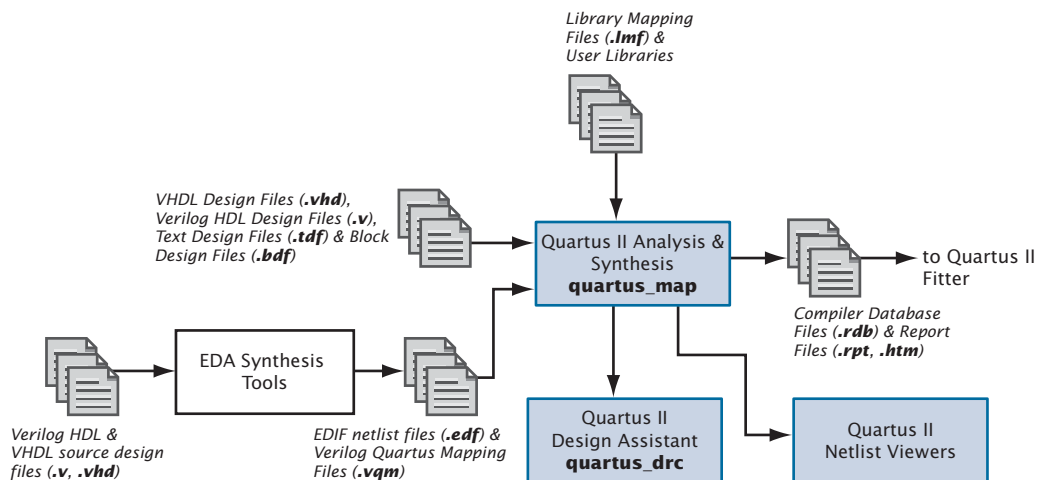
You can start a full compilation in the Quartus II software, which includes the Analysis & Synthesis module, or you can start Analysis & Synthesis separately. You can perform an Analysis & Elaboration to check a design for syntax and semantic errors without performing a complete Analysis & Synthesis 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.

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

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 27 and "Instantiating Megafunctions in EDA Tools" on page 28 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.

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. 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.

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
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 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.

> ☞ **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 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 2).

Figure 2. RTL Viewer
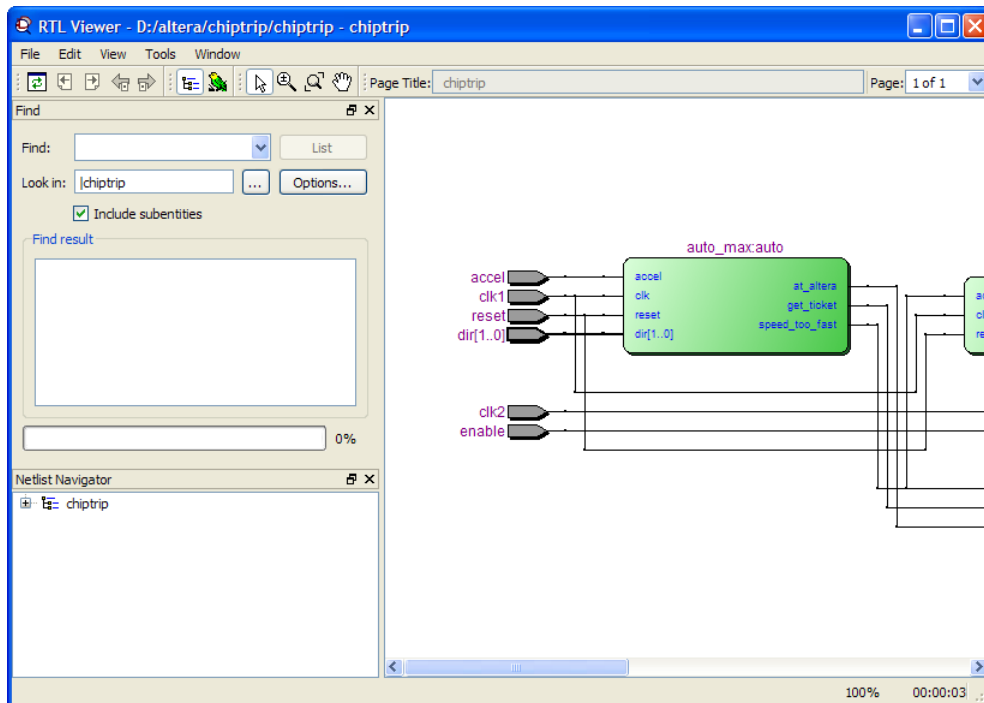
The RTL Viewer displays the Analysis & Elaboration results for Verilog HDL or VHDL designs, and AHDL Text Design Files (**.tdf**), Block Design Files (**.bdf**), and Graphic Design Files (**.gdf**). For Verilog Quartus Mapping Files or EDIF netlist files that were generated from other EDA synthesis tools, the RTL Viewer displays the hierarchy for the atom representations of WYSIWYG primitives.

You can select one or more items in the Netlist Navigator 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.

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 3).

Figure 3. State Machine Instance in RTL Viewer

Double-clicking a state machine instance symbol in the RTL Viewer opens the State Machine Viewer window
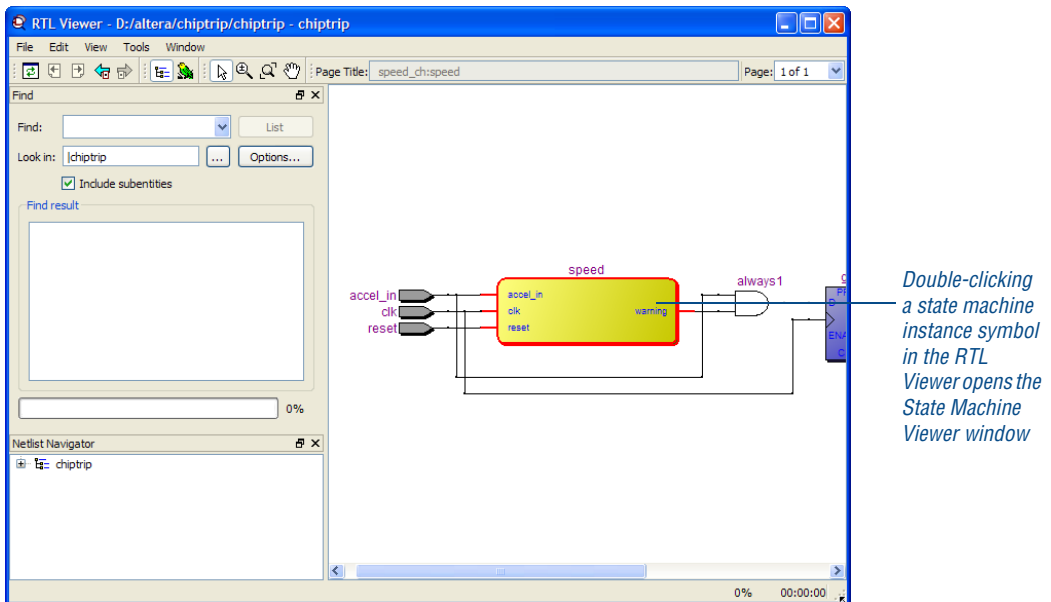
The State Machine Viewer includes a schematic view and a State Table (Figure 4).

Figure 4. State Machine Viewer

Schematic view

State Table shows source and destination states and transition conditions
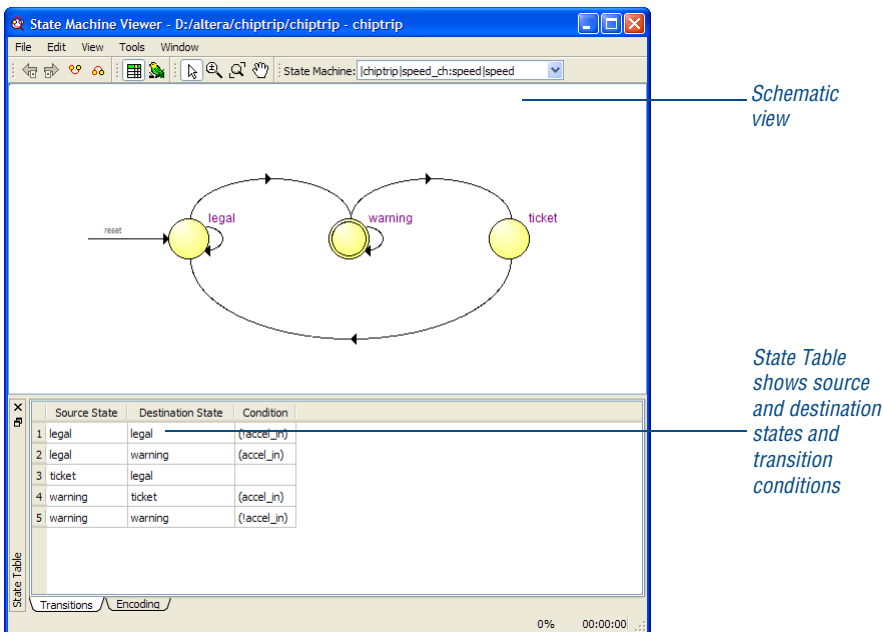
When you select a cell in a transition table, the corresponding state or transition is highlighted in the schematic view. Likewise, when you select a state or transition in the schematic view, the corresponding cell is highlighted in the transition table. The schematic view allows you to zoom in and 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 5).

Figure 5. Technology Map Viewer
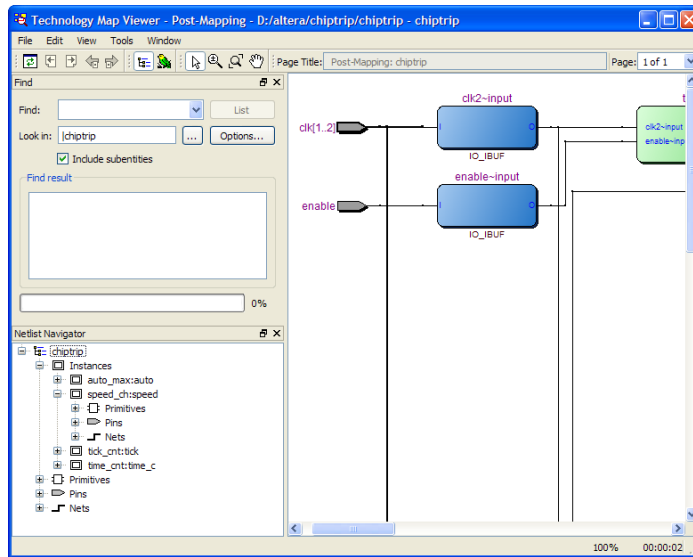
You can also use the Technology Map Viewer to display post-Analysis & Synthesis mapping and compare those results to the results from a full compilation. Display the results from Analysis & Synthesis by pointing to **Netlist Viewers** on the Tools menu, and then clicking **Technology Map Viewer (Post-Mapping)**.

☞ **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 45. 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 72 in Chapter 5, "Timing Analysis and Design Optimization."
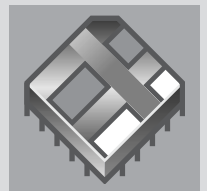
For Information About	Refer To
Using the Quartus II Technology Map Viewer	*Analyzing Designs with Quartus II Netlist Viewers* chapter in volume 1 of the *Quartus II Handbook*
	"About the Netlist Viewers" in Quartus II Help

Chapter

Four

Place and Route
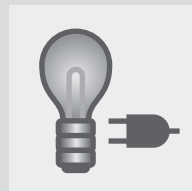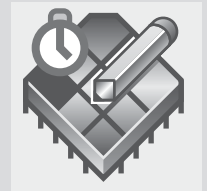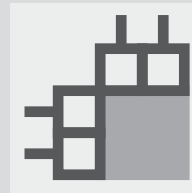
4

Introduction
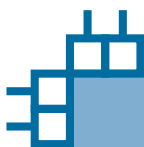
The Quartus II Fitter places and routes your design, which is also referred to as "fitting" in the Quartus II software. Using the database that has been created by Analysis & Synthesis, the Fitter matches the logic and timing requirements of the project with the available resources of the target device. 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
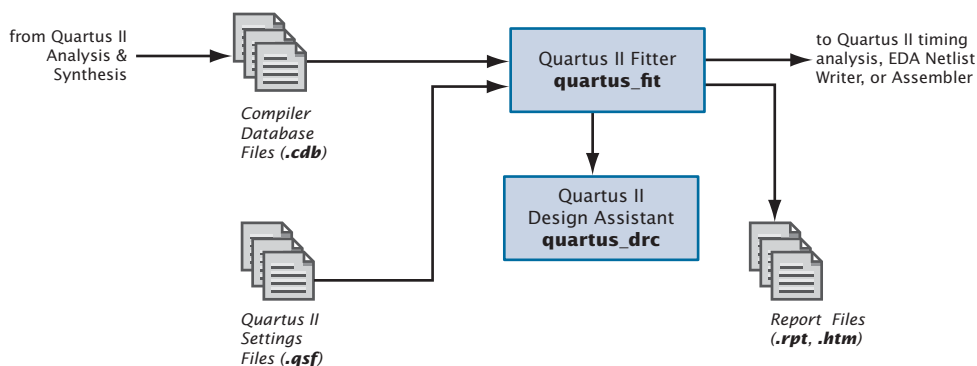
If you have made resource assignments in your design, the Fitter attempts to match those resource assignments with the resources on the device, tries to meet any other constraints you have set, and then attempts to optimize the remaining logic in the design. If you have not set any constraints on the 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 perform 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> ↵
> ```

Using Incremental Compilation

The Quartus II software performs incremental compilation to reuse previous compilation results for unchanged entities in the design. For more information, refer to "Design Methodologies and Planning" on page 14 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 the appropriate **Netlist Type** for the partitions. To preserve compilation and placement results, set the **Netlist Type** for the partitions to **Post-Fit**.

4. Assign each partition to a physical location on the device by using the Chip Planner and LogicLock assignments.

5. Compile the design.

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

7. 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 2 shows the Messages window.

Figure 2. Messages Window
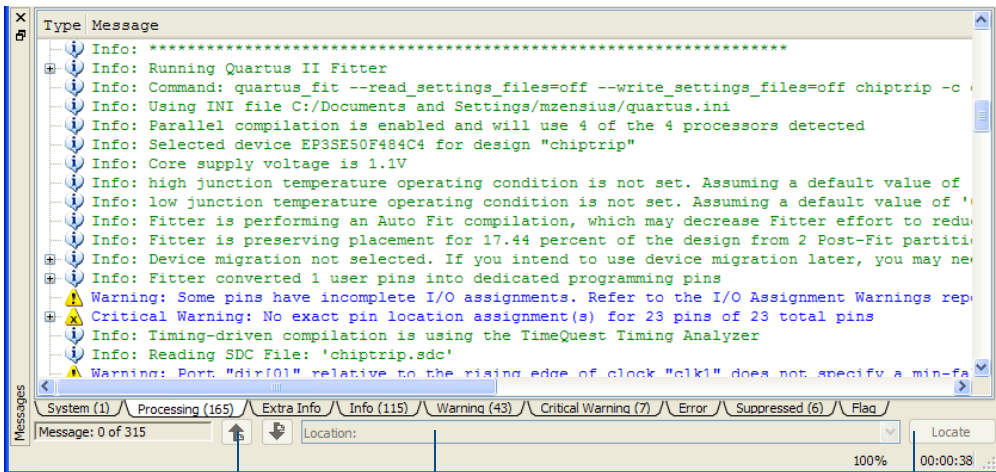
Arrow buttons allow you to select next and previous messages

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.

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.

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 3).

Figure 3. Chip Planner
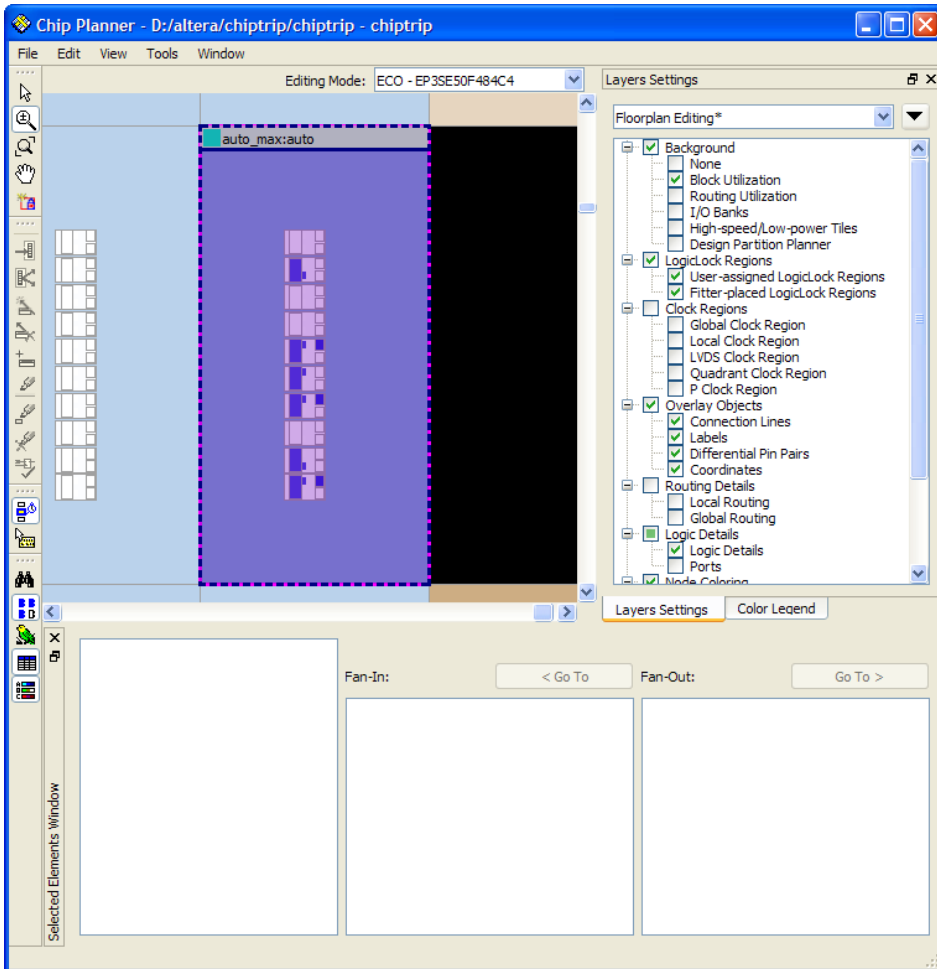
Resource usage in the Chip Planner is color coded. Different colors represent different resources, such as unassigned and assigned pins and logic cells, unrouted items, and row FastTrack® fan-outs. The Chip Planner also allows you to customize the floorplan view using filters to show pins and the interior 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 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.

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. 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 .

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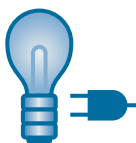
The Resource Optimization Advisor offers recommendations for optimizing your design for resource usage in the following areas:

- 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.

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 4.

Figure 4. Resource Optimization Advisor Recommendation Page
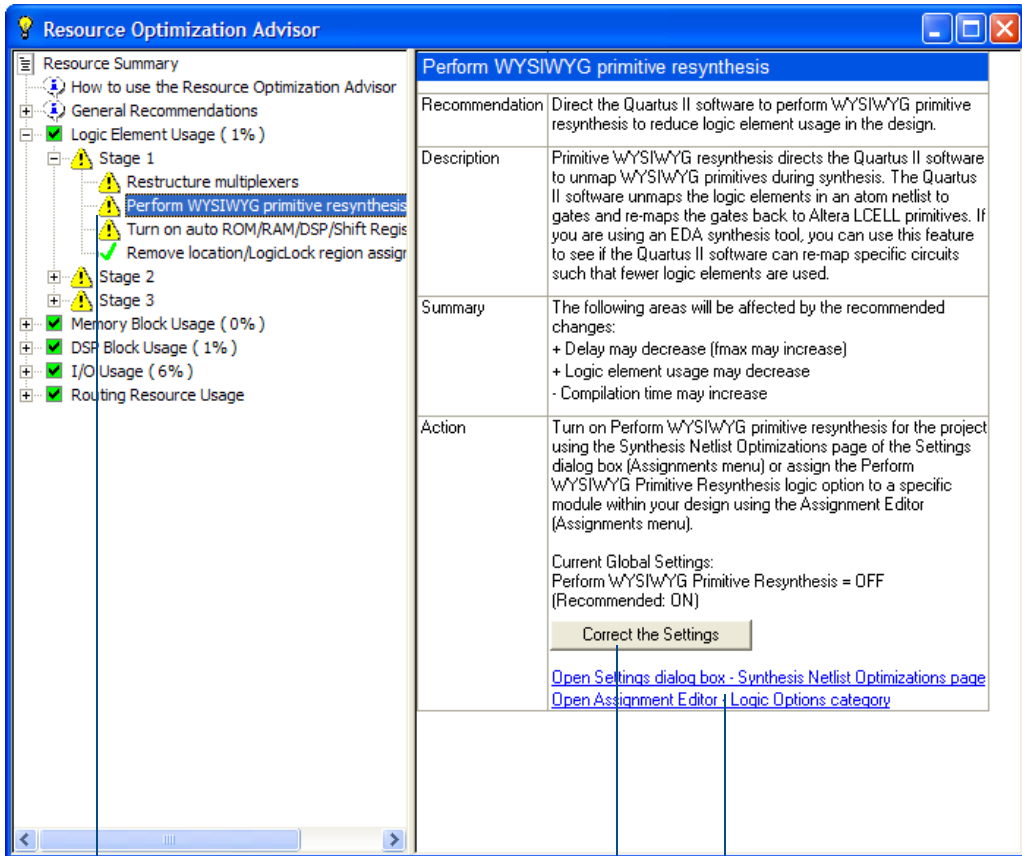
Hierarchical list of recommendations—icons indicate potential problem areas

Some recommendations include buttons that make the reccomended changes 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 DSE 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
Parameters and settings for optimizing performance	*Area and Timing Optimization* chapter in volume 2 of the *Quartus II Handbook*

Chapter Five

Timing Analysis and Design Optimization
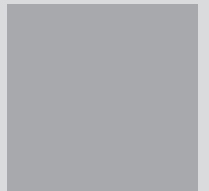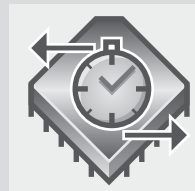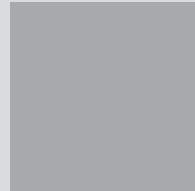
5

Introduction

The Quartus II TimeQuest Timing Analyzer allows you to analyze the timing characteristics of your design. 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

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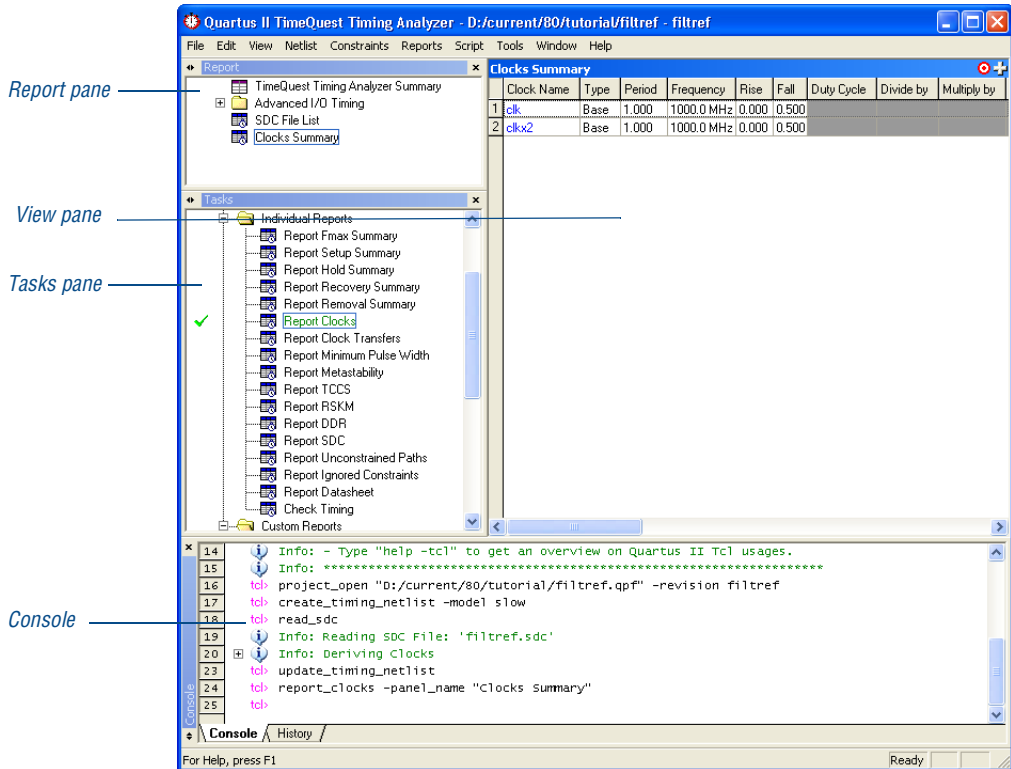
The **View** pane displays timing analysis results, including any summary reports, custom reports, or histograms. Figure 2 shows the **View** pane when you use the **Report Clocks** command in the **Tasks** pane for a design that includes two defined clocks, clk and clkx2.

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

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

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 3).

Figure 3. Report Timing Dialog Box
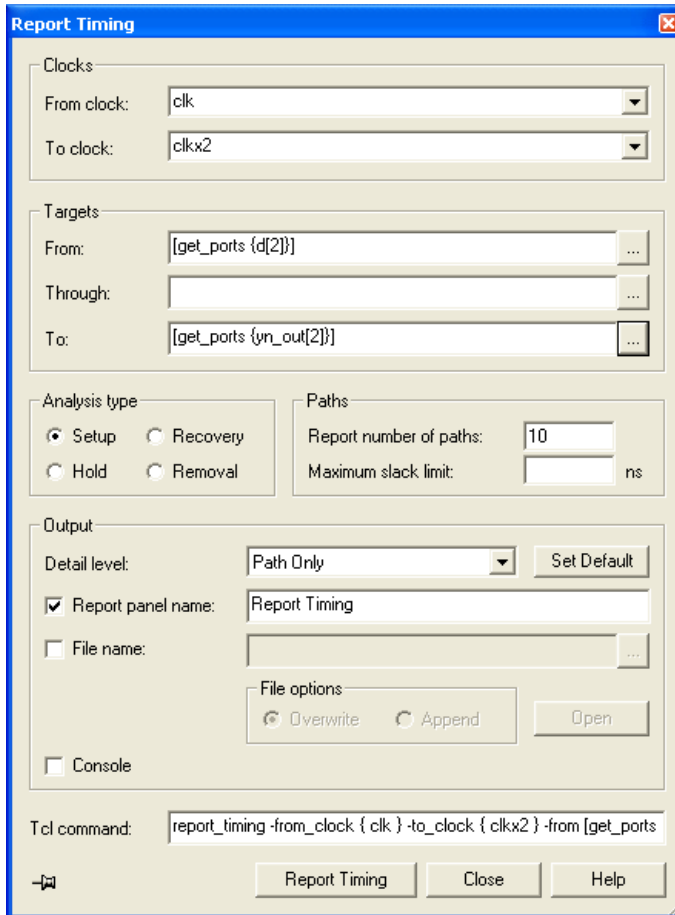
When information about a path is reported by the TimeQuest analyzer, you can use the **Locate Path** command directly from the timing analyzer reports to view path information in the Chip Planner, Technology Map Viewer, and RTL Viewer.

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 4).

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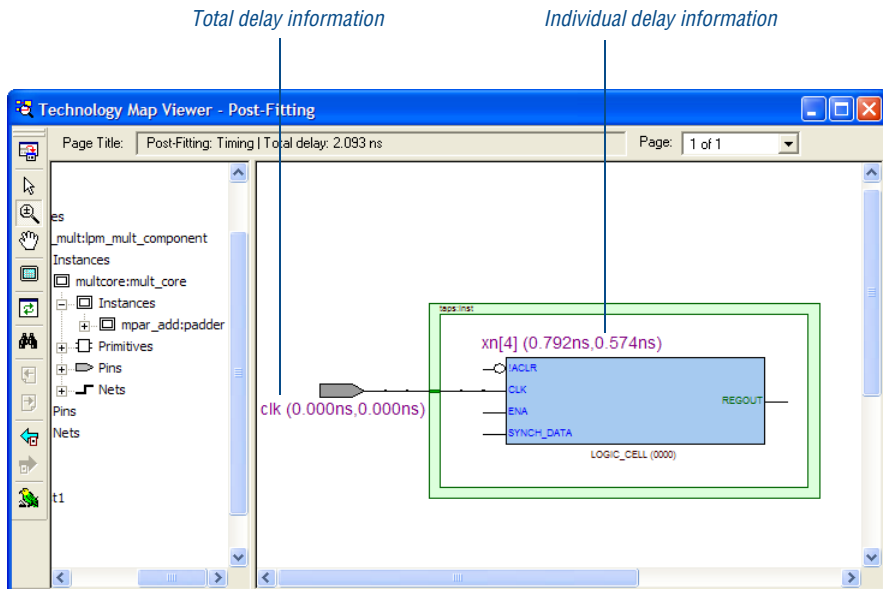
For Information About	Refer To
Using the Quartus II Technology Map Viewer	*Analyzing Designs with Quartus II Netlist Viewers* chapter in volume 1 of the *Quartus II Handbook*

Timing Closure

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 the 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 5 shows the timing closure flow.

Figure 5. Timing Closure Flow

ALTERA CORPORATION
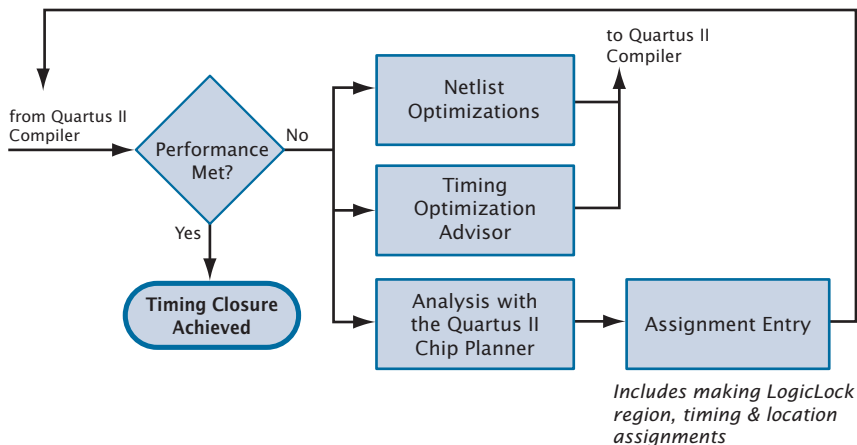
Using the Chip Planner

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, and regions.. 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 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, the Timing Optimization Advisor can provide specific timing recommendations for the project, based on the project information and current settings.

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.

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**).

- **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.

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.

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 DSE, 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 63 in Chapter 4, "Place and Route."

Power Analysis with the PowerPlay Power Analyzer

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. The PowerPlay power analysis flow is shown in Figure 6.

Figure 6. PowerPlay Power Analysis Flow
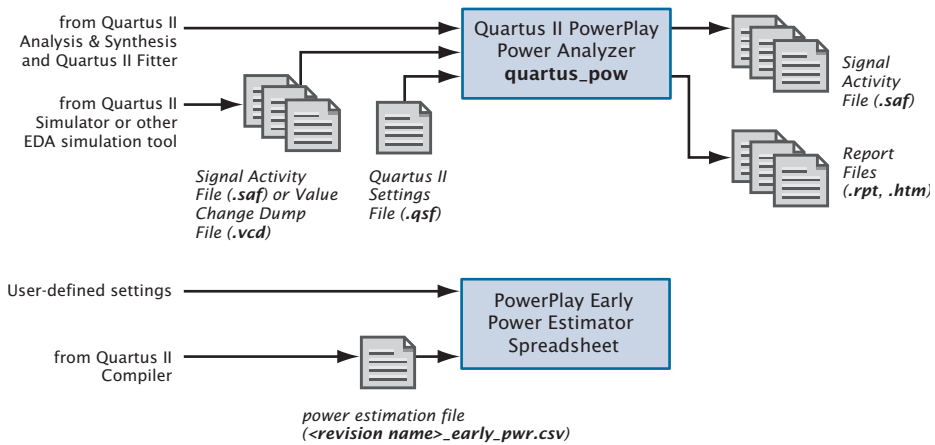
You can use the **PowerPlay Power Analyzer Tool** command on the Processing menu after running Analysis & Synthesis and the Fitter successfully. You can specify whether you want to use an input file, such as a Signal Activity File (**.saf**) or Value Change Dump File (**.vcd**) generated by the Quartus 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.

☞ **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

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. 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
Six

Programming &
Configuration

6

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
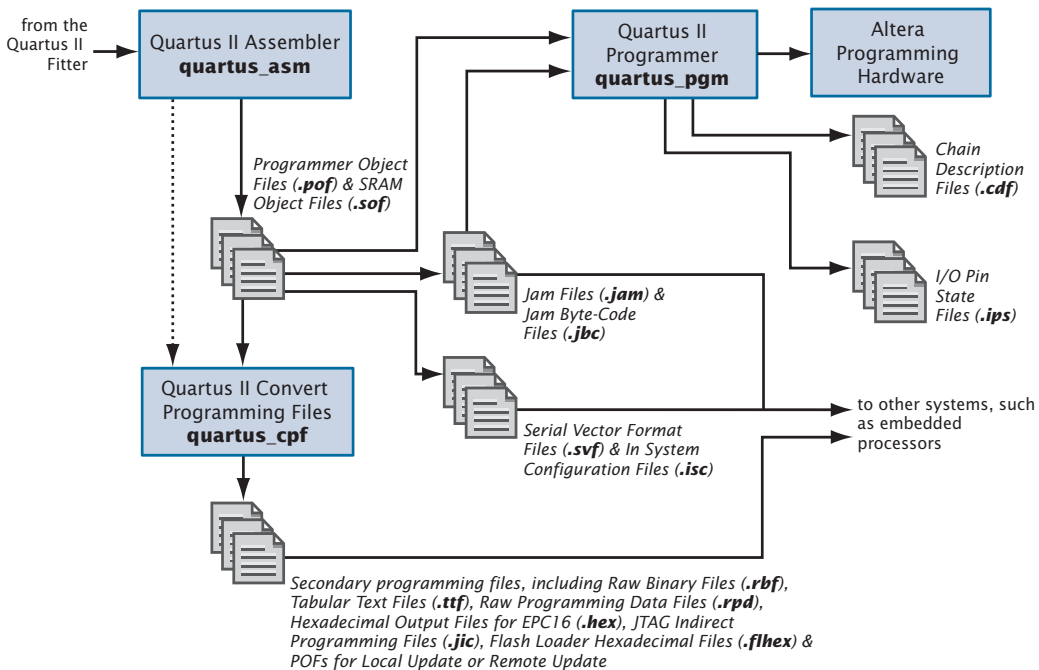
Creating and Using Programming Files

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 you can view 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> ↵
```

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
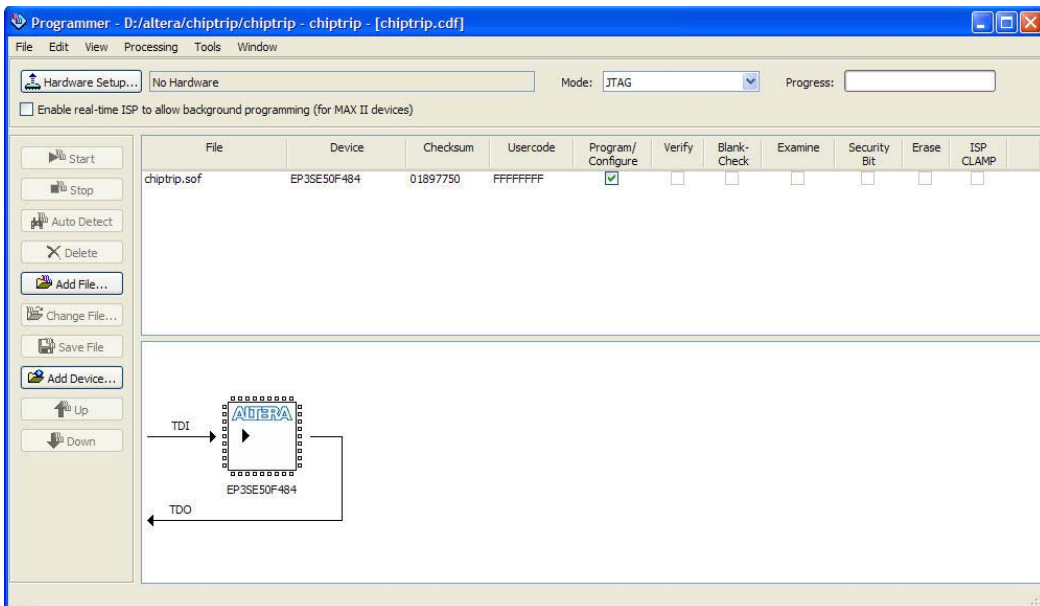
> ☞   **Using the quartus_pgm executable**
>
> You can also run the Programmer separately at the command prompt or in a script by using the **quartus_pgm** executable. You may need to run the Assembler executable, **quartus_asm**, in order to produce a programming file before running the Programmer.
>
> 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

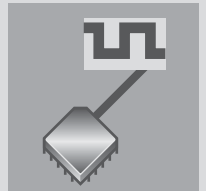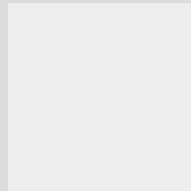| For Information About	Refer To
Altera programming hardware	*MasterBlaster Serial/USB Communications Cable User Guide, ByteBlaster II Download Cable User Guide, ByteBlasterMV Download Cable User Guide, USB-Blaster Download Cable User Guide*, and *EthernetBlaster Communications Cable User Guide* on the Altera website
Device-specific programming information	The *Configuration Handbook* on the Altera website

Chapter Seven

Debugging and Engineering Change Managment

7

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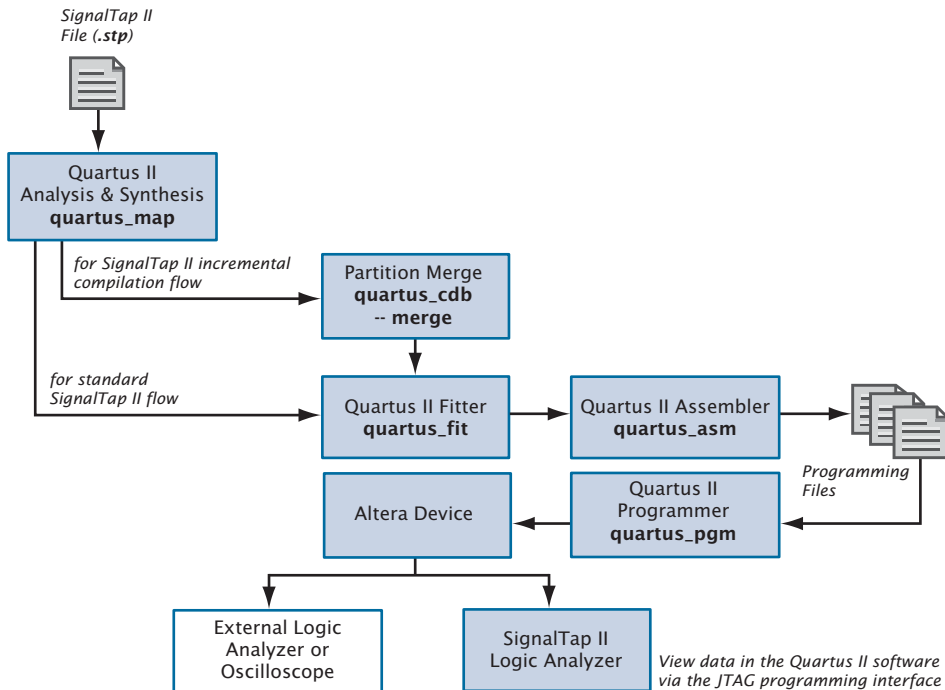
Figure 2. SignalProbe Debugging Flow
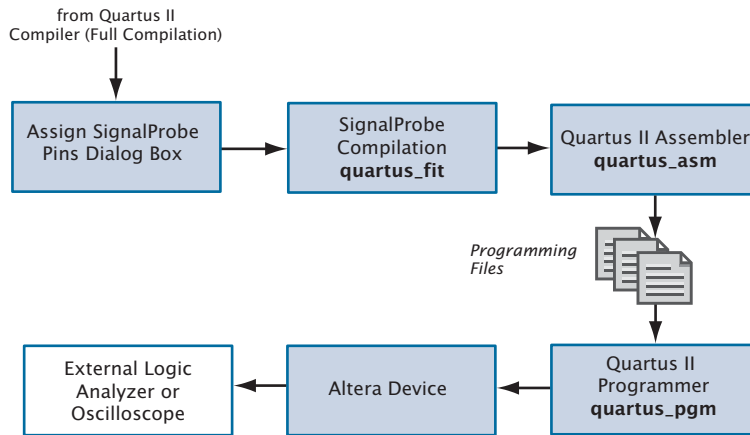
Using the SignalTap II Logic Analyzer

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.

shows the SignalTap II Logic Analyzer.

Figure 3. The SignalTap II Logic Analyzer
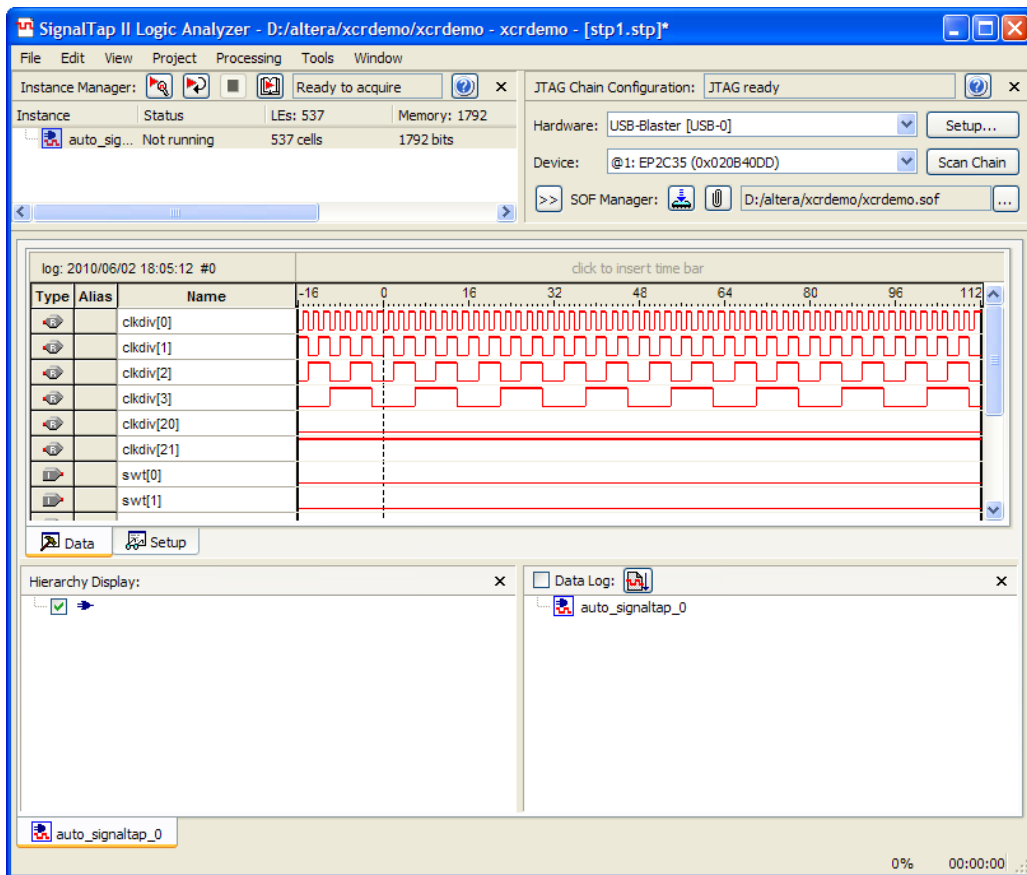
Analyzing SignalTap II Data

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.

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.

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 recompile the design. 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.

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.

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.

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 **In-System 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.

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 page 48 in Chapter 3, "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.

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 4 shows the engineering change management design flow.

Figure 4. Engineering Change Management Design Flow
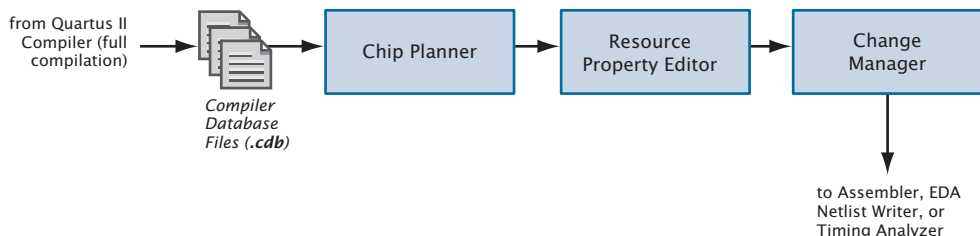
The following steps describe the design flow for engineering change management in the Quartus II software.

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 5.

Figure 5. Chip Planner
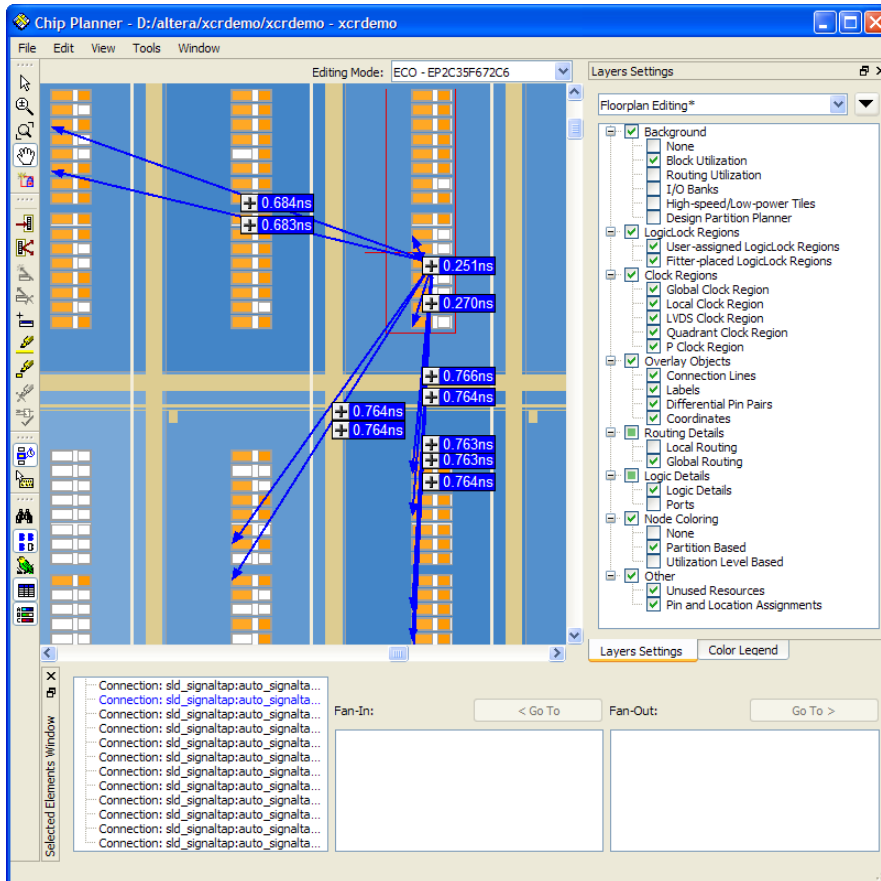
You can then use the information from the Chip Planner to determine which properties and settings you may want to edit in the Resource Property Editor. Right-click one or more resources in the Chip Planner, and then click **Locate in Resource Property Editor** to open the Resource Property Editor and make edits to the resource(s). Refer to "Modifying Resource Properties With the Resource Property Editor" on page 101 for more information.

Modifying Resource Properties With the Resource Property Editor
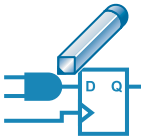
The Resource Property Editor allows you to make post-compilation edits to the properties and parameters of logic cell, I/O element, or PLL resources, as well as edit or remove connections for individual nodes. You can use the toolbar buttons to navigate forward and backward among the resources. You can also select and change multiple resources at one time. In addition, when you roll over or point to 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 6 shows the Resource Property Editor.

Figure 6. Resource Property Editor

Viewer shows schematic diagram of resource

Connectivity panel
shows the input and
output ports

Delay Information panel
shows delay information
for the selected node
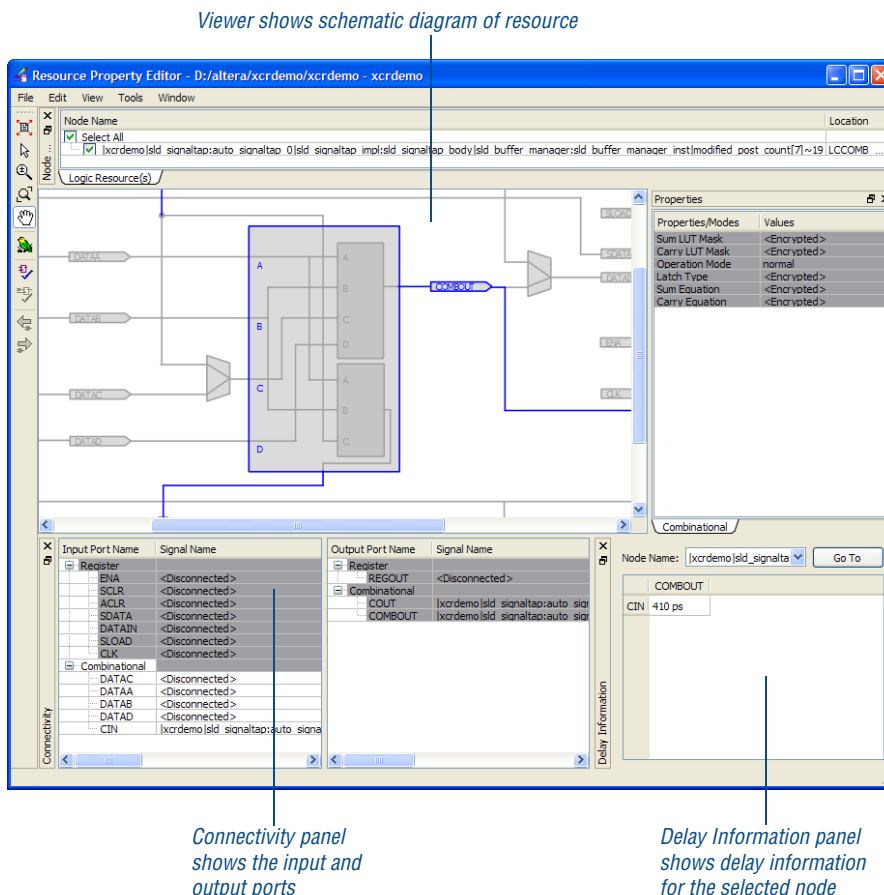
The Resource Property Editor allows you to right-click a node in the schematic or in the port connection table and click **Edit Connection** to specify a new signal for the connection. If you want to remove the connection, you can right-click the node and click **Remove Connection**. In the port connection 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 7.

Figure 7. Change Manager

Index ∕	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
⊞ 1	state_m:inst1filter.idle:CLK:0	Modify Source	filtref|clk	filtref|reset	filtref|reset	filtref|clk
⊞ 2	taps:instxn[0]:DATAD:0	Modify Source	filtref|d[0]	VCC	filtref|d[0]	filtref|d[0]
⊞ 3	taps:instxn[0]:DATAD:0	Modify Source	VCC	filtref|d[0]	filtref|d[0]	filtref|d[0]
⊞ 4	taps:instxn_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.

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
Eight

EDA Tool Support
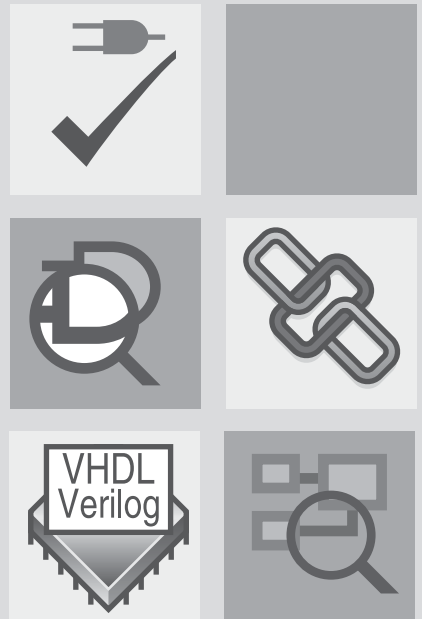
8

Introduction

The Quartus II software allows you to use the EDA tools you are familiar with for various stages of the design flow, including synthesis, simulation, and formal verification. Figure 1 shows the EDA tool design flow.

Figure 1. EDA Tool Design Flow
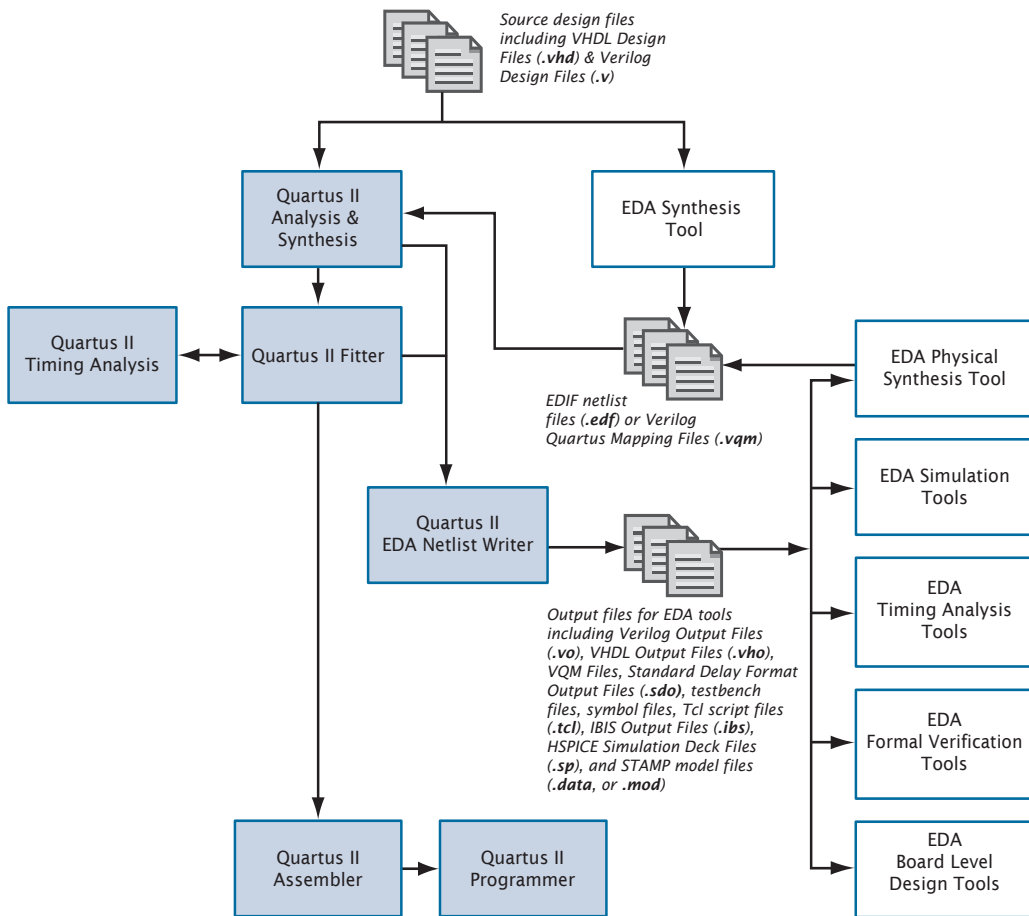
The following steps describe the basic design flow for using other EDA tools with the Quartus II software:

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 II post-fit netlist is equivalent to that of the synthesized netlist.

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

☞   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> ↵
```

Table 1 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 1. EDA Tools Supported by the Quartus II Software

Function	Supported EDA Tools
Synthesis	Mentor Graphics® LeonardoSpectrum
	Mentor Graphics Precision RTL Synthesis
	Synopsys Synplify
	Synopsys Synplify Pro
	Magma Blast FPGA
Simulation	Cadence Incisive Enterprise Simulator
	Mentor Graphics ModelSim®
	Mentor Graphics ModelSim-Altera
	Mentor Graphics QuestaSim
	Synopsys VCS MX
	Synopsys VCS
	Aldec Active-HDL
Timing Analysis	Mentor Graphics Tau (through Stamp)
	Synopsys PrimeTime
Formal Verification	Cadence Encounter Conformal

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.

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*

EDA Simulation Tools

You can perform functional and timing simulation of your design by using EDA simulation tools. 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 2 shows the simulation flow with EDA simulation tools.

Figure 2. Simulation Flow
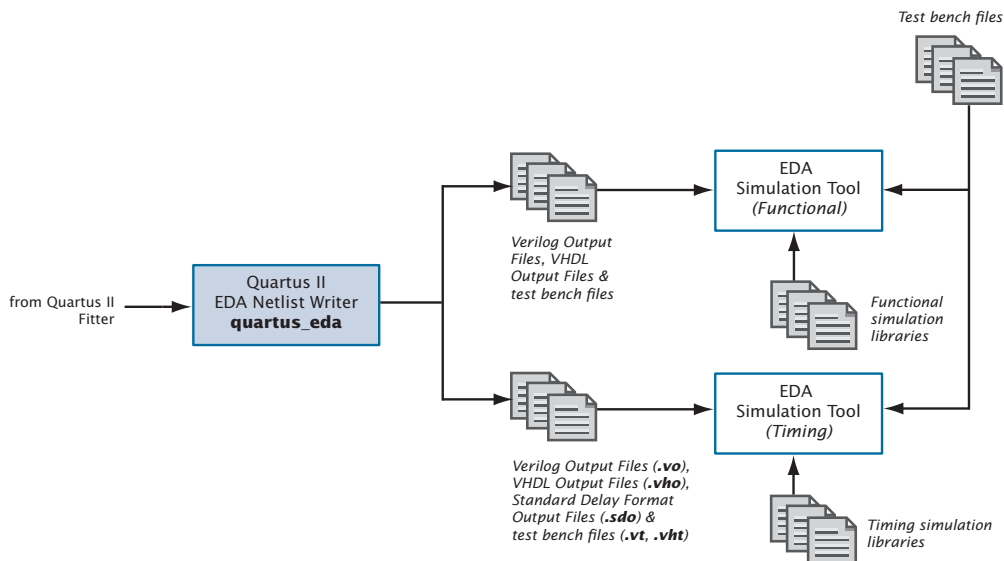
The EDA Netlist Writer module of the Quartus II software generates VHDL Output Files (**.vho**) and Verilog Output Files (**.vo**) for performing functional or timing simulation, and Standard Delay Format Output Files (**.sdo**) that are required for performing timing simulation with EDA simulation tools. 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 to launch EDA simulation tools from within the Quartus II software.

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.

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.

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 pre-compiled 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 Cadence Incisive Enterprise Simulator 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*

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.

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 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

Formal Verification

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 3 shows the formal verification flow.

Figure 3. Formal Verification Flow
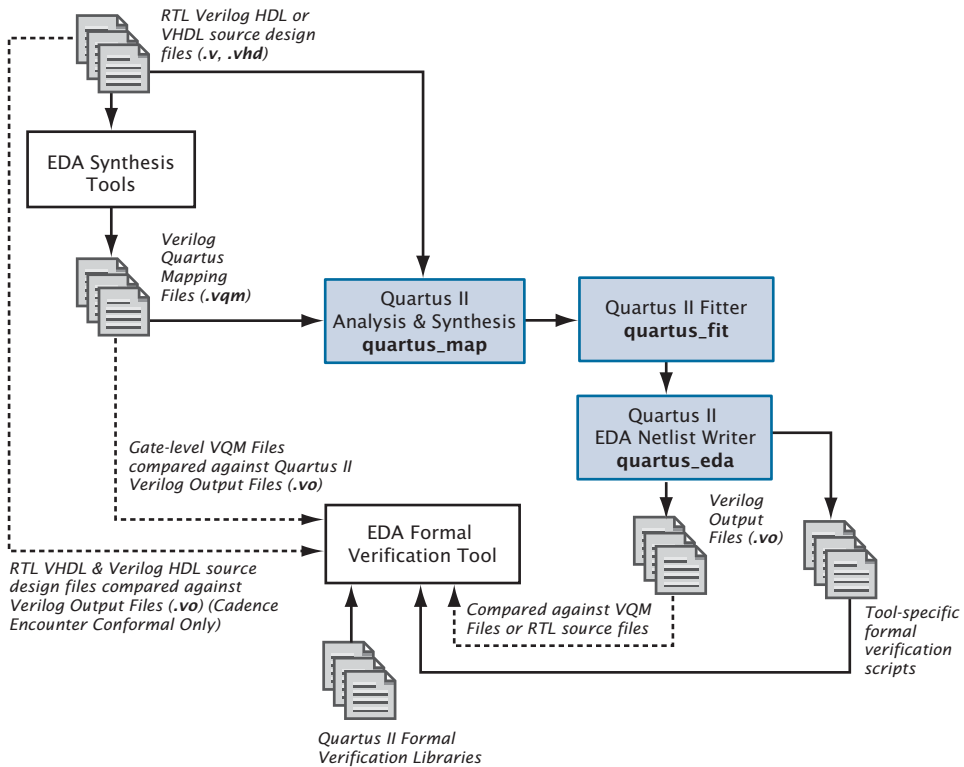
The type of formal verification supported by the Quartus II software is equivalence checking, which compares the functional equivalence of the source design with the revised design by using mathematical techniques rather than by performing simulation using test vectors. Equivalence checking greatly decreases 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 4 shows which file types are compared in formal verification.

Figure 4. File Types Compared in Formal Verification
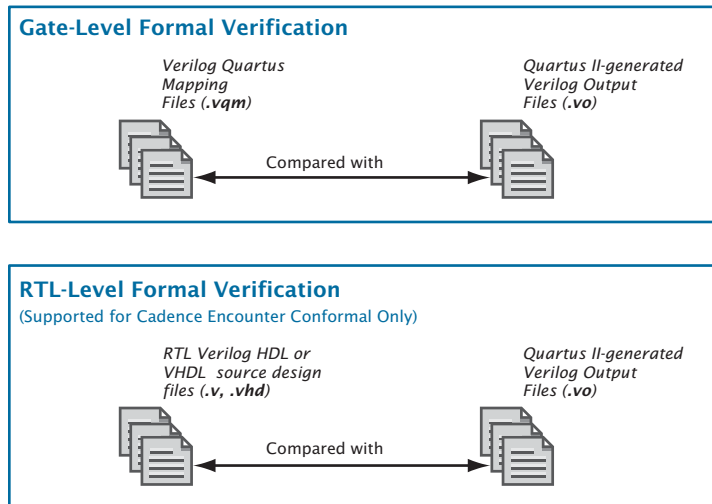
Using the Cadence Encounter Conformal Software

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.

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 Nine

System Requirements, Licensing & Technical Support

9

Installing the Quartus II Software

You can install the Quartus II software on the following platforms:

■ Pentium III (866 MHz or faster) based computer, running Microsoft Windows.
  – PCs running Windows XP are capable of running the 32-bit version of the Quartus II software with access to virtual memory of 2 GB.
  – PCs running Windows XP Professional x64 Edition or Windows Vista are capable of running the 32-bit version of the Quartus II software with access to virtual memory of up to 4 GB and the 64-bit version of the Quartus II software with access to virtual memory of more than 4 GB.

■ One of the following Linux workstations:
  – Intel Pentium III or compatible processor-based PC operating at 450 MHz or faster with 256 MB of system memory, running Red Hat Enterprise Linux; CentOS; or SUSE Linux Enterprise (32-bit).
  – AMD64 processor, Intel EM64T processor, or compatible processor-based PC with 1 GB of system memory, running Red Hat Enterprise Linux; CentOS; or SUSE Linux Enterprise Server (64-bit).

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	Altera Complete Design Suite **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 Type	Description
Fixed License	A stand-alone PC license tied to your Network Interface Card (NIC) number or Quartus II serial number.
Floating License	A floating network (multiuser) license. Floating licenses are not operating system-specific.

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.

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 a myAltera 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 a myAltera account and user name. A myAltera account also makes it easier for you to use many other Altera website features, such as the Download Center, Self Service Licensing Center, Altera Technical Training online class registration, or Buy On-Line-Altera eStore features.

To register for a myAltera 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 a myAltera account.

If you are not a current Altera subscription user, you can still register for an myAltera account.

For information about other technical support resources, refer to Table 2.

Table 2. Quartus II Technical Support Resources (Part 1 of 2)

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.

Table 2. Quartus II Technical Support Resources (Part 2 of 2)

Resource	Description
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)

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:

■   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 individual Help topic that you want to print and click Print, or click the Print button on the toolbar. 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, 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 3 shows the additional software documentation that is available for the Quartus II software:

Table 3. 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 3. 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.

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.

Altera Corporation
101 Innovation Drive
San Jose, California 95134 USA
www.altera.com
www.altera.com/mysupport