Introduction to Quartus® II

Altera Corporation
101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

Introduction to Quartus II
Version 3.0 Revision 1
June 2003

P25-09235-00

Preface

You hold in your hands the *Introduction to Quartus II* manual. The Altera® Quartus® II design software is the most comprehensive environment available for system-on-a-programmable-chip (SOPC) design. If you have primarily used the MAX+PLUS® II software, other design software, or ASIC design software in the past, and are thinking of making the switch to the Quartus II software, or, if you are somewhat familiar with the Quartus II software but would like to gain a greater knowledge of its capabilities, this manual is for you.

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

The first chapter gives 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. It shows how to integrate the Quartus II software with your existing EDA tool and command-line design flows. In addition, the manual refers you to other resources that are available to help you use the Quartus II software, such as Quartus II online Help and the Quartus II online tutorial, application notes, white papers, and other documents and resources that are available on the Altera web site.

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

Documentation Conventions

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

Typographic Conventions

Quartus II documentation uses the following typographic conventions:

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 names, project names, disk drive names, file names, file name extensions, software utility names, software executable names, 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, 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: *FLEXlm End Users Guide*.
italics	Variables are enclosed in angle brackets (< >) and shown in italics. For example: *<file name>*, *<CD-ROM drive>*.
`Courier font`	Anything that must be typed exactly as it appears is shown in Courier. For example: `\quartus\bin\lmulti lmhostid`.
↵	Enter or return key.
■	Bullets are used in a list of items when the sequence of the items is not important.

Visual Cue:	Meaning:
	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

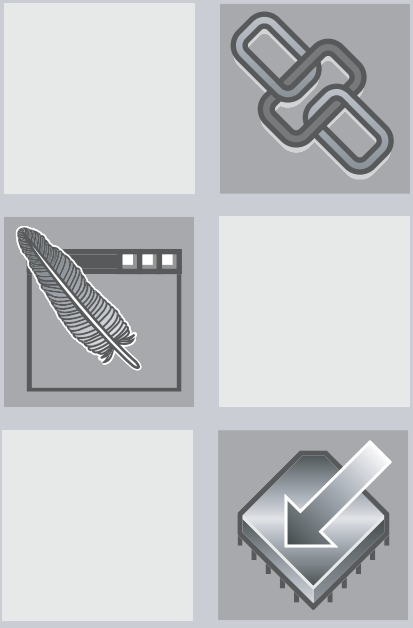
The following terminology is used throughout the *Introduction to Quartus II* manual:

Term:	Meaning:
"click"	Indicates a quick press and release of the left mouse button.
"double-click"	Indicates two clicks in rapid succession.
"choose"	Indicates that you need to use a mouse or key combination to start an action.
"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 click **OK**.
"turn on"/"turn off"	Indicates that you must click a check box to turn a function on or off.

Contents

Chapter One

Programmable Logic Design Flow



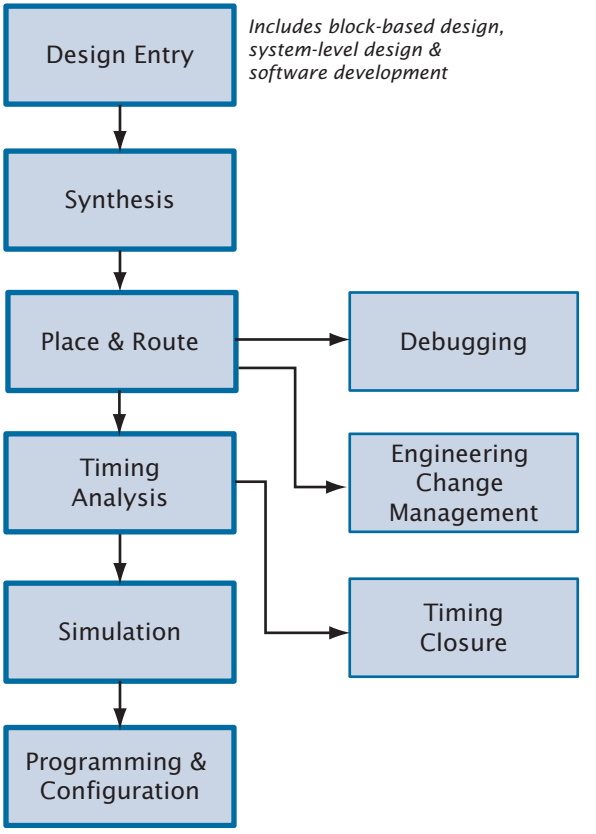
1

What's in Chapter 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. See Figure 1 for an illustration of the Quartus II design flow.

Figure 1. Quartus II Design Flow



In addition, the Quartus II software allows you to use the Quartus II graphical user interface, EDA tool interface, or 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 of the design

flow. This chapter explains the options that are available for each of the design flows. The remaining chapters in this manual describe individual stages of the design flow in more detail.

Graphical User Interface Design Flow

You can use the Quartus II software to perform all stages of the design flow; it is a complete, easy-to-use, stand-alone solution. Figure 2 shows the features that the Quartus II graphical user interface provides for each stage of the design flow.

Figure 2. Quartus II Graphical User Interface Features

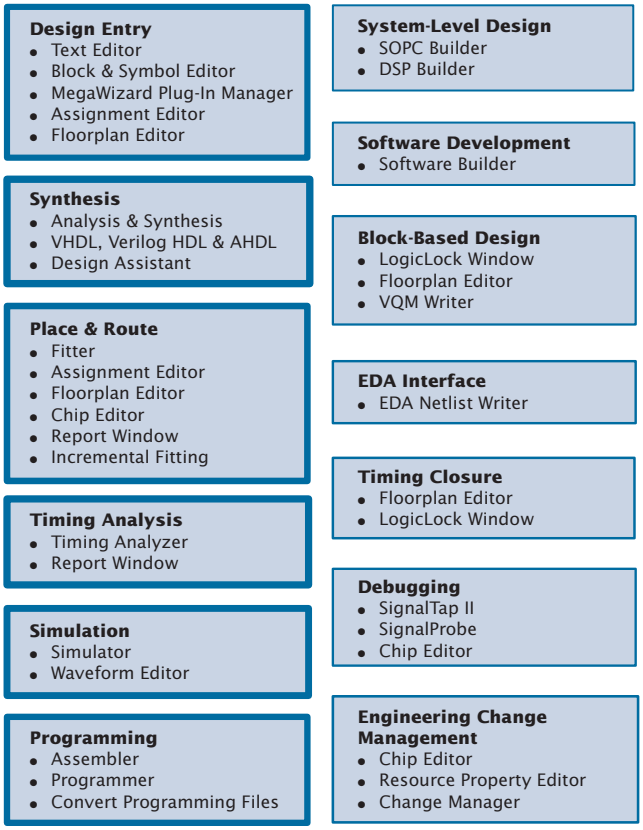
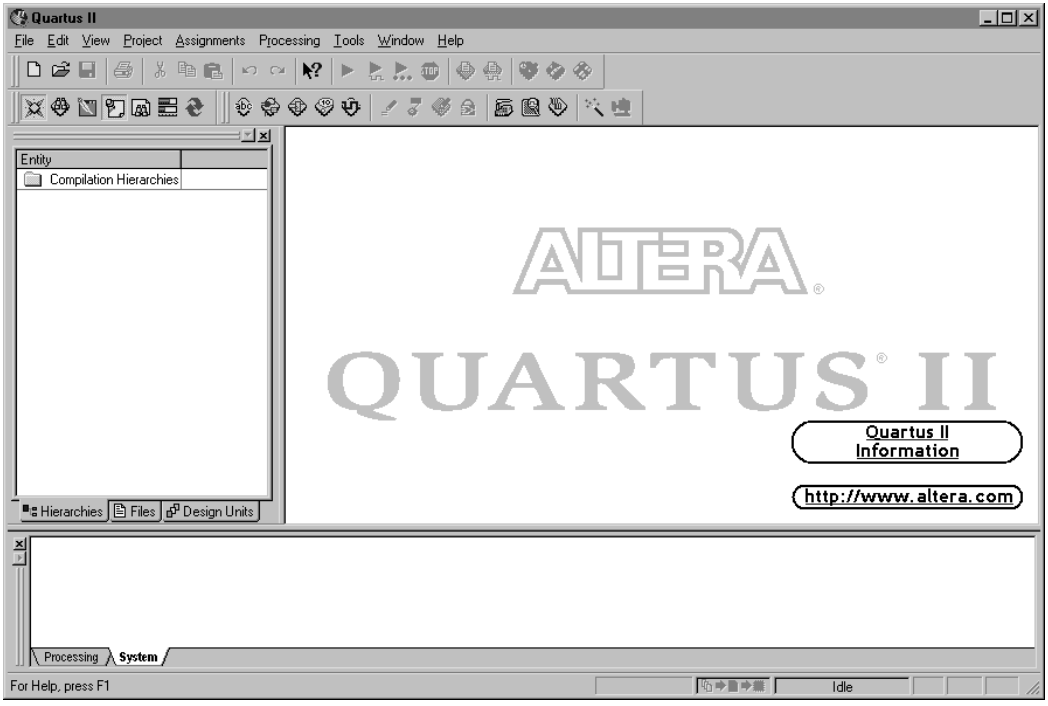


Figure 3 shows the Quartus II graphical user interface as it appears when you first start the software.

Figure 3. Quartus II Graphical User Interface



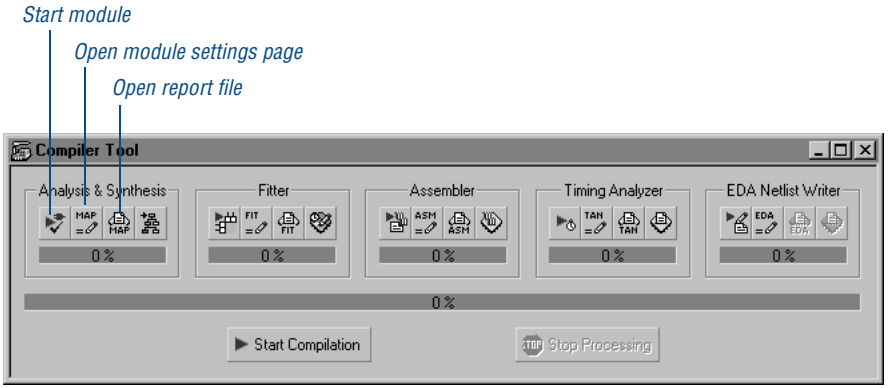
The Quartus II software includes a modular Compiler. The Compiler includes the following modules (modules marked with an asterisk are optional during compilation, depending on your settings):

- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
- Design Assistant*
- EDA Netlist Writer*
- Compiler Database Interface*

You can run all Compiler modules as part of a full compilation by choosing **Start Compilation** (Processing menu). You can also run each module individually by choosing **Start** (Processing menu) and then choosing the command for the module you want to start from the **Start** submenu.

In addition, you can start the Compiler modules by choosing **Compiler Tool** (Tools menu) and running the module in the Compiler Tool window. The Compiler Tool window also allows you to open the settings file or report file for the module, or to open other related windows.

Figure 4. Compiler Tool Window



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

1. Create a new project and specify a target device or device family by using the **New Project Wizard** (File menu).

2. Create a Verilog HDL, VHDL, or Altera Hardware Description Language (AHDL) design by using the Text Editor. If you want, you can use the Block Editor to create a block diagram with symbols that represent other design files, or to create a schematic. You can also use the **MegaWizard® Plug-In Manager** to generate custom variations of megafunctions and IP cores to instantiate in your design.

3. (Optional) Specify initial design constraints using the Assignment Editor, the **Settings** dialog box (Assignments menu), the Floorplan Editor, and/or the LogicLock™ feature.

4. (Optional) Create a system-level design by using the SOPC Builder or DSP Builder.
5. (Optional) Create software and programming files for Excalibur™ device processors or Nios® embedded processors by using the Software Builder.
6. Synthesize the design by using Analysis & Synthesis.
7. (Optional) Perform functional simulation on the design by using the Simulator.
8. Perform place and route on the design by using the Fitter. If you have made a small change to the source code, you can also use incremental fitting.
9. Perform timing analysis on the design by using the Timing Analyzer.
10. Perform timing simulation on the design by using the Simulator.
11. (Optional) Make timing improvements to achieve timing closure by using physical synthesis, the Timing Closure floorplan, the LogicLock feature, the **Settings** dialog box, and the Assignment Editor.
12. Create programming files for your design by using the Assembler.
13. Program the device by using programming files, the Programmer, and Altera hardware; or convert programming files to other file formats for use by other systems, such as embedded processors.
14. (Optional) Debug the design by using the SignalTap® II Logic Analyzer, the SignalProbe™ feature, or the Chip Editor.
15. (Optional) Manage engineering changes by using the Chip Editor, the Resource Property Editor, and the Change Manager.

EDA Tool Design Flow

The Quartus II software allows you to use the EDA tools you are familiar with for various stages of the design flow. You can use these tools together with the Quartus II graphical user interface or with Quartus II command-line executables. See Figure 5 on page 7.

Figure 5. EDA Tool Design Flow

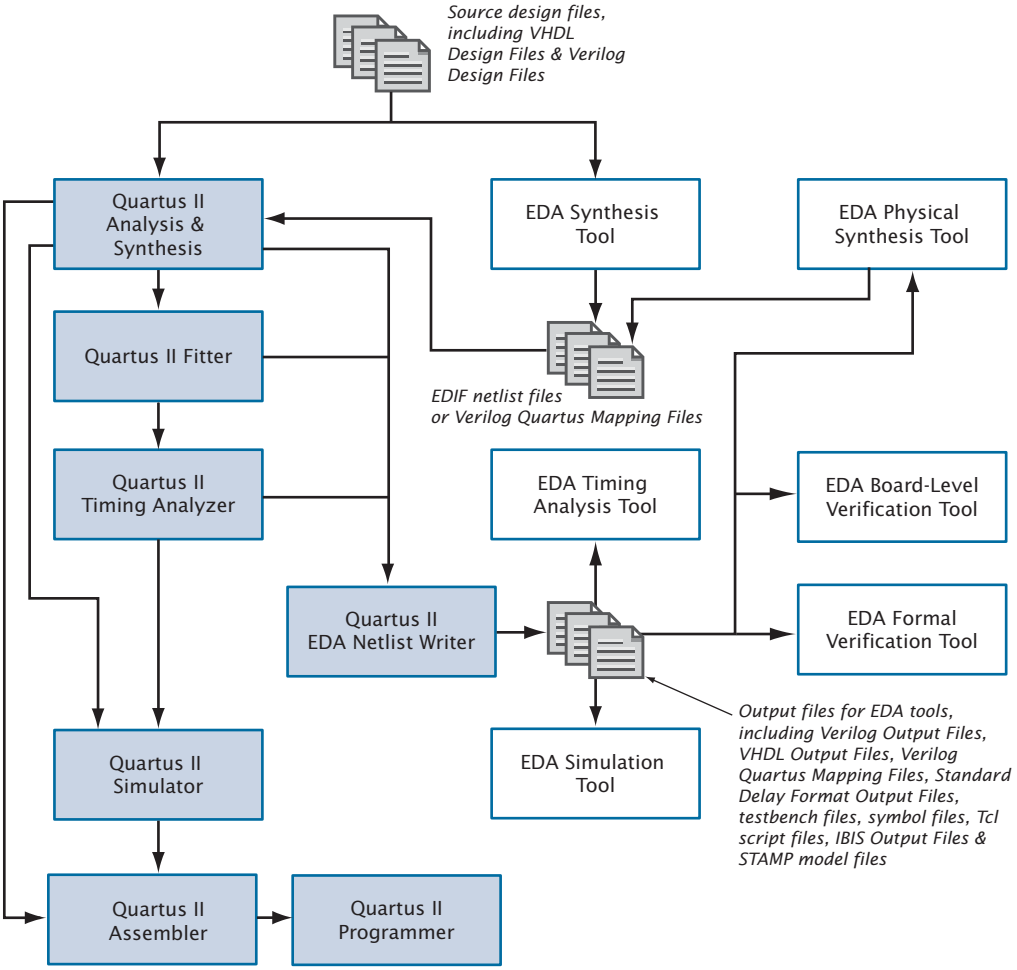


Table 1 shows the EDA tools that are supported by the Quartus II software, and indicates which EDA tools have NativeLink® support. NativeLink technology facilitates the seamless transfer of information between the Quartus II software and other EDA tools and allows you to run the EDA tool automatically from within the Quartus II software.

Table 1. EDA Tools Supported by the Quartus II Software (Part 1 of 2)

Function	Supported EDA Tools	NativeLink Support
Synthesis	Mentor Graphics Design Architect	
	Mentor Graphics LeonardoSpectrum	✓
	Mentor Graphics Precision RTL Synthesis	✓
	Mentor Graphics ViewDraw	
	Synopsys Design Compiler	
	Synopsys FPGA Express	
	Synopsys FPGA Compiler II	✓
	Synplicity Synplify	✓
	Synplicity Synplify Pro	
Simulation	Cadence NC-Verilog	✓
	Cadence NC-VHDL	✓
	Cadence Verilog-XL	
	Model Technology™ ModelSim®	✓
	Model Technology ModelSim-Altera	✓
	Synopsys Scirocco	✓
	Synopsys VSS	
	Synopsys VCS	
Timing Analysis	Mentor Graphics Blast (through Stamp)	
	Mentor Graphics Tau (through Stamp)	
	Synopsys PrimeTime	✓

Table 1. EDA Tools Supported by the Quartus II Software (Part 2 of 2)

Function	Supported EDA Tools	NativeLink Support
Board-Level Verification	Hyperlynx (through Signal Integrity IBIS)	
	XTK (through Signal Integrity IBIS)	
	ICX (through Signal Integrity IBIS)	
	SpectraQuest (through Signal Integrity IBIS)	
	Mentor Graphics Symbol Generation (Viewdraw)	
Formal Verification	Verplex Conformal LEC	
Resynthesis	Aplus Design Technologies (ADT) PALACE	✓
	Synplicity Amplify	

The following steps describe the basic design flow for using other EDA tools with the Quartus II software. Refer to Table 1 on page 8 for a list of the supported EDA tools.

1. Create a new project and specify a target device or device family.
2. Create a VHDL or Verilog HDL design file by using a standard text editor. If you want, instantiate functions from libraries, or use the **MegaWizard Plug-In Manager** (Tools menu) to create custom variations of megafunctions.
3. Synthesize your design by using one of the Quartus II–supported EDA synthesis tools, and generate an EDIF netlist file (**.edf**) or a Verilog Quartus Mapping File (**.vqm**).
4. (Optional) Perform functional simulation on your design by using one of the Quartus II–supported simulation tools.
5. In the Quartus II **Settings** dialog box (Assignments menu), specify options and Library Mapping Files for processing EDIF netlist files (**.edf**), VHDL Design Files (**.vhd**), Verilog Design Files (**.v**), Verilog Quartus Mapping Files (**.vqm**), and AHDL Text Design Files (**.tdf**) that were generated by other design entry or synthesis tools or by the **MegaWizard Plug-In Manager**.

6. (Optional) In the Quartus II **Settings** dialog box, specify options for generating VHDL Output Files (**.vho**), Verilog Output Files (**.vo**), Standard Delay Format Output Files (**.sdo**), Stamp model files, PartMiner XML-Format Files (**.xml**), and IBIS Output Files (**.ibs**).

7. Compile your design and perform place and route by using the Quartus II software. You can perform a full compilation, or you can run the Compiler modules individually:

   – Run Analysis & Synthesis to process your design and map the functions in your design to the correct library module.

   – Run the Fitter to place and route your design.

   – Run the Timing Analyzer to perform timing analysis on your design.

   – Run the EDA Netlist Writer to generate output files for use with other EDA tools.

   – Run the Assembler to create programming files for your design.

8. (Optional) Perform timing analysis on your design by using one of the Quartus II–supported EDA timing analysis tools.

9. (Optional) Perform timing simulation on your design by using one of the Quartus II–supported EDA simulation tools.

10. (Optional) Perform board-level verification by using one of the Quartus II–supported EDA board-level verification tools.

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

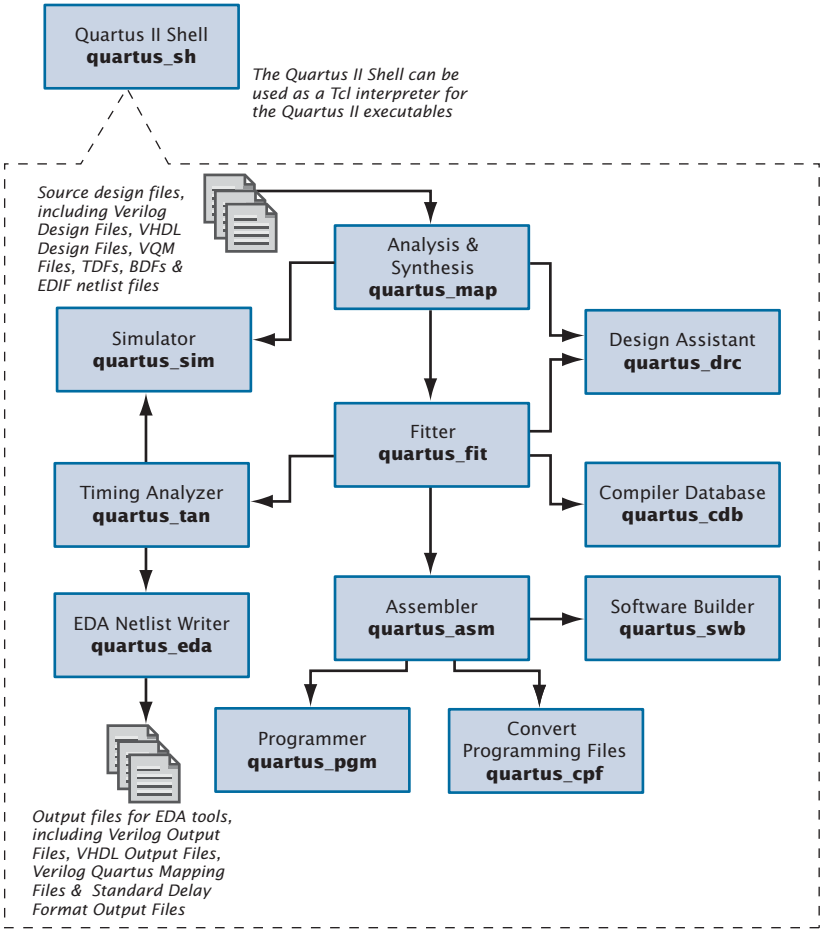
12. (Optional) Perform board-level resynthesis by using one of the Quartus II–supported EDA resynthesis tools.

13. Program the device by using programming files, the Programmer, and Altera hardware; or convert programming files to other file formats for use by other systems, such as embedded processors.

Command-Line Design Flow

The Quartus II software offers a complete command-line interface solution. It allows you to perform every stage of the design flow by using command-line executables and options. Using the command-line flow allows you to reduce memory requirements, control the Quartus II software by using scripts or standard command-line options and commands, including Tcl commands, and create makefiles. See Figure 6 for an illustration of the command-line design flow.

Figure 6. Command-Line Design Flow



Command-Line Executables

The Quartus II software includes separate executables for each stage of the design flow. Each executable occupies memory only while it is being run. You can use these executables with standard command-line commands and scripts, with Tcl scripts, and in makefile scripts. See Table 2 for a list of all of the available command-line executables.

Stand-Alone Graphical User Interface Executables

The Quartus II software also provides some stand-alone graphical user interface (GUI) executables. The **qmegawiz** executable provides a stand-alone GUI version of the **MegaWizard Plug-In Manager**. The **quartus_pgmw** executable provides a stand-alone GUI interface for the Programmer.

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 the project's design files.
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. Either Analysis & Synthesis or the Fitter must be run successfully before running the Design Assistant.
quartus_tan	Timing Analyzer	Analyzes the speed performance of the implemented circuit. The Fitter must be run successfully before running the Timing Analyzer.
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.

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

Executable Name	Title	Function
quartus_cdb	Compiler Database Interface (including VQM Writer)	Generates internal netlist files, including VQM Files for the Quartus II Compiler Database, so they can be used for back-annotation and for the LogicLock feature. Either the Fitter or Analysis & Synthesis must be run successfully before running the Compiler Database Interface.
quartus_sim	Simulator	Performs functional or timing simulation on your design. Analysis & Synthesis must be run before performing a functional simulation. The Timing Analyzer must be run before performing a timing simulation.
quartus_pgm	Programmer	Programs Altera devices.
quartus_cpf	Convert Programming Files	Converts programming files to secondary programming file formats.
quartus_swb	Software Builder	Processes a design for an Excalibur embedded processor.
quartus_sh	Tcl Shell	Provides a Tcl scripting shell for the Quartus II software.

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 run each executable individually, but you can also run all the Compiler executables at once by using the following command:

```
quartus_sh --flow compile <project name>
     [-c <Compiler Settings File name>] ↵
```

This command will run the **quartus_map**, **quartus_fit**, **quartus_asm**, and **quartus_tan** executables as part of a full compilation. Depending on your settings, it may also run the optional **quartus_drc**, **quartus_eda**, and **quartus_cdb** executables.

The quartus_cmd Executable

If you have used the **quartus_cmd** executable to perform project compilation in previous versions of the Quartus II software, this executable is still supported for backward compatibility; however, Altera recommends that for all new designs, you do not use the **quartus_cmd** executable, but use the executables that are listed in Table 2 on page 12. If you are using the **quartus_cmd** executable to compile a design, you should type the following command and options:

```
quartus_cmd <project name> -c <Compiler Settings File name>.csf ↵
```

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

***<project name or settings name>.<abbreviated executable name>*.rpt**

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

```
quartus_map chiptrip ↵
```

The **quartus_map** executable will perform analysis and synthesis and will produce a report file with the name **chiptrip.map.rpt**.

Using Settings Files with Quartus II Executables

Altera recommends that you name your settings files with the same name as the project when using the Quartus II executables.

If, however, you do have a settings file name that is different from the project name, you can use the `-c` option to specify the settings file name to use. For example, if you want to run the **quartus_map** executable for the **chiptrip** project with **speed_ch** settings, you could type the following command at the command prompt:

```
quartus_map chiptrip -c speed_ch ↵
```

The **quartus_map** executable performs analysis and synthesis and produces a report file with the name **speed_ch.map.rpt**.

Using Standard Command-Line Commands & Scripts

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

Figure 7 on page 16 shows an example of a standard command-line script. The example demonstrates how to create a project, perform analysis and synthesis, perform place and route, perform timing analysis, and generate programming files for the **filtref** tutorial design that is included with the Quartus II software. If you have installed the tutorial design, it is in the **/<*Quartus II system directory*>/qdesigns/tutorial** directory. Altera recommends that you create a new directory and copy all the design files (***.v, *.bsf, *.bdf**) from the **/<*Quartus II system directory*>/qdesigns/tutorial** directory to the new directory, in order to compile the design flow example. You can run the four commands in Figure 7 from a command prompt in the new project directory, or you can store them in a batch file or shell script. These examples assume that the **/<*Quartus II system directory*>/bin** directory (or the **/<*Quartus II system directory*>/<*platform*>** directory on UNIX or Linux workstations, where **<*platform*>** can be **solaris**, **linux**, or **hp_II**) is included in your `PATH` environment variable.

Figure 7. Example of a Command-Line Script

```
quartus_map filtref --family=Stratix

quartus_fit filtref --part=EP1S10F780C5 --fmax=80MHz --tsu=8ns

quartus_tan filtref

quartus_asm filtref
```

Creates a new Quartus II project targeting the Stratix device family

Performs fitting for the EP1S10F780C5 device and specifies global timing requirements

Performs timing analysis

Generates programming files

Figure 8 shows an excerpt from a sample **quartus_sh** command-line script for use on a UNIX 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 8. Example of a UNIX Command-Line Shell Script

```
#!/bin/sh

FILES_WITH_ERRORS=""

for filename in `ls *.bdf *.v`
do
        quartus_map fir_filter --csf=filtref.csf --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
```

Using Tcl Commands

In the Quartus II software, you can run Tcl commands or create and run Tcl scripts with the Quartus II executables to do the following tasks in a Quartus II project. The Tcl API functions include the following categories:

- Project & assignment functions
- Device functions
- Advanced device functions
- Flow functions
- Timing functions
- Advanced timing functions
- Simulator functions
- Report functions
- Timing report functions
- Back-annotate functions
- LogicLock functions
- Chip Editor Functions
- Miscellaneous functions

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

The **Templates** command (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, the **Generate Tcl File for Project** command (Project menu) can generate a Tcl Script File for the project.

You can run Tcl scripts in command-line mode with the **quartus_sh** executable, in the Quartus II Tcl Console window, or from the **Tcl Scripts** dialog box (Tools menu).

Getting Help On Tcl Commands

The Quartus II software includes a 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 ↵
```

Figure 9 shows an example of a Tcl Script.

Figure 9. Example of a Tcl Script (Part 1 of 2)

```
# Since ::quartus::report is not pre-loaded
# by quartus_sh, load this package now
# before using the report Tcl API
package require ::quartus::report

# Since ::quartus::flow is not pre-loaded
# by quartus_sh, load this package now
# before using the flow Tcl API
# Type "help -pkg flow" to view information
# about the package
package require ::quartus::flow

#------ Get Actual Fmax data from the Report File ------#

proc get_fmax_from_report {} {
#------------------------------------------------------#
      global project_name

      # Load the project report database
      load_report $project_name

      # Find the "Timing Analyzer Summary" panel name containing
      # the Actual Fmax data by traversing the panel names
      # Then set the panel row containing the Actual Fmax
      # information

      set fmax_panel_name "Timing Analyzer Summary"
      foreach panel_name [get_report_panel_names] {
      if { [string match "*$fmax_panel_name*" "$panel_name"] } {
      # Fmax is sorted so we just need to go to Row 1
      set fmax_row [get_report_panel_row "$panel_name" -row 1]
      }
```

Figure 9. Example of a Tcl Script (Part 2 of 2)

```
      # Actual Fmax is found on the fourth column
      # Index starts at 0
      set actual_fmax [lindex $fmax_row 3]

      # Now unload the project report database
      unload_report $project_name

      return $actual_fmax
}

#------ Set the project name to chiptrip ------#
set project_name chiptrip

#------ Create or open project ------#
if {project_exists $project_name} {

#------ Project already exists -- open project -------#
      project_open $project_name} {
}else {

#------ Project does not exist -- create new project ------#
      project_new $project_name
}

#------ Fmax requirement: 155.55MHz ------#
set required_fmax 155.55MHz

#------ Make global assignments ------#
set_global_assignment -name family STRATIX
set_global_assignment -name device EP1S10F484C5
set_global_assignment -name fmax_requirement $required_fmax
set_global_assignment -name tsu_requirement 7.55ns

#------ Make instance assignments ------#
# The following is the same as doing:
#   "set_instance_assignment -name location -to clock Pin_M20"
set_location -to clock Pin_M20

#------ Compile using ::quartus::flow ------#
execute_flow -compile

#------ Report Fmax from report ------#
set actual_fmax [get_fmax_from_report]
puts ""
puts "-----------------------------------------------------"
puts "Required Fmax: $required_fmax Actual Fmax: $actual_fmax"
puts "-----------------------------------------------------"
```

Creating Makefile Scripts

The Quartus II software supports makefile scripts that use the Quartus II executables, which allow you to integrate your scripts with a wide variety of scripting languages. Figure 10 shows an excerpt from a standard makefile script.

Figure 10. Excerpt from Makefile Script (Part 1 of 2)

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

PROJECT = chiptrip
SOURCE_FILES = auto_max.v chiptrip.v speed_ch.v tick_cnt.v time_cnt.v
ASSIGNMENT_FILES = chiptrip.quartus chiptrip.psf chiptrip.csf

######################################################################
# Main Targets
#
# all: build everything
# clean: remove output files and database
# clean_all: removes settings files as well as clean.
######################################################################

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

clean:
      rm -rf *.rpt *.chg smart.log *.htm *.eqn *.pin *.sof *.pof db
clean_all: clean
      rm -rf *.ssf *.csf *.esf *.fsf *.psf *.quartus *.qws

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

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

MAP_ARGS = --family=Stratix
FIT_ARGS = --part=EP1S20F484C6
ASM_ARGS =
TAN_ARGS =
```

Figure 10. Excerpt from Makefile Script (Part 2 of 2)

```
######################################################################
# Target implementations
######################################################################

STAMP = echo done >

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

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

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

$(PROJECT).tan.rpt: tan.chg $(PROJECT).fit.rpt
      quartus_tan $(TAN_ARGS) $(PROJECT)

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

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

$(ASSIGNMENT_FILES):
      quartus_sh --tcl_eval project_new $(PROJECT) -overwrite

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

For Information About	Refer To
Using Command-Line Executables	"Overview: Using Command-Line Executables" in Quartus II Help *Application Note 309 (Command-Line Scripting in the Quartus II Software)* on the Altera web site
Tcl Commands and Tcl Scripting	"Overview: Using Tcl from the User Interface," "Overview: Using Tcl Scripting," and "API Functions for Tcl" in Quartus II Help *Application Note 195 (Scripting with Tcl in the Quartus II Software)* on the Altera web site

Chapter Two

Design Entry

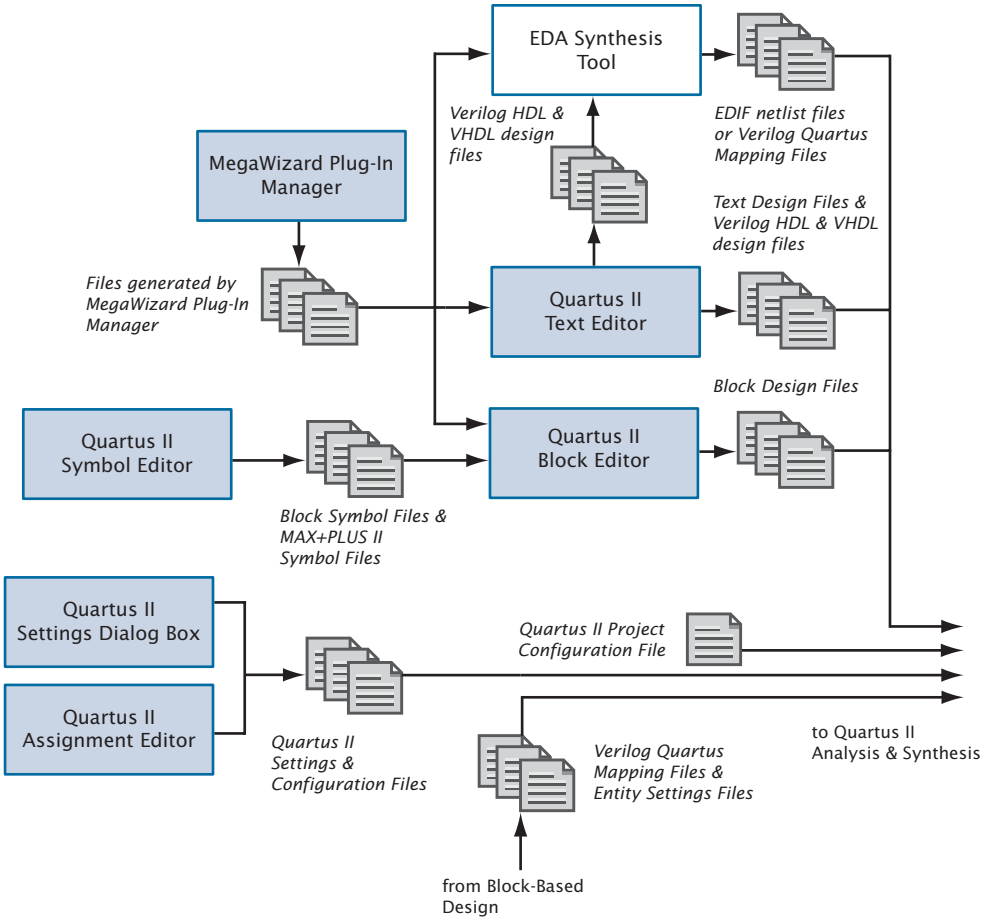
2

What's In Chapter 2:

Introduction

A project in the Quartus® II software is comprised of all the design files and settings associated with your design. You can use the Quartus II Block Editor, Text Editor, **MegaWizard® Plug-In Manager** (Tools menu), and EDA design entry tools to create designs that include Altera® megafunctions, library of parameterized modules (LPM) functions, and intellectual property (IP) functions. You can use the **Settings** dialog box (Assignments menu) and the Assignment Editor to make initial design constraints. Figure 1 shows the design entry flow.

Figure 1. Design Entry Flow



Creating a Project

The Quartus II software stores project information in the Quartus II Project Configuration File (**.quartus**). It contains all the information about your Quartus II project, including the design files; waveform files; SignalTap® II Files; memory initialization files; and Compiler, Simulator, and software build settings that comprise the project. You can create a new project by using the **New Project Wizard** (File menu) or the **quartus_map** executable.

With the **New Project Wizard**, you can 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 family and device (or allow the Quartus II software to automatically select a device).

Once you have created a project, you can add and remove design and other files from the project using the **Add/Remove** page of the **Settings** dialog box (Assignments menu). During Quartus II Analysis & Synthesis, the Quartus II software processes the files in the order they appear in the **Add/Remove** page.

Using the quartus_map executable

You can use Analysis & Synthesis separately at the command prompt to create a new project or add files to a project by using the **quartus_map** executable.

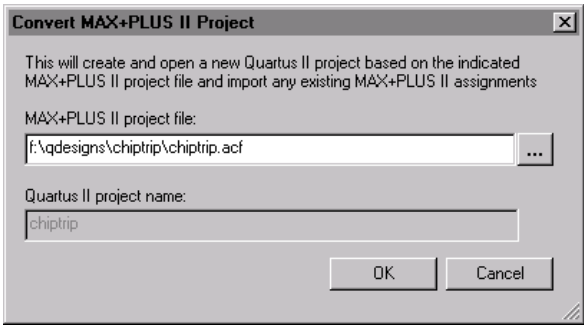
The **quartus_map** executable creates a separate text-based report file that can be viewed with any text editor.

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> ↵
```

If you have an existing MAX+PLUS® II project, you can also use the **Convert MAX+PLUS II Project** command (File menu) to convert the MAX+PLUS II Assignment & Configuration File (**.acf**) into a Quartus II project. The Quartus II software creates a new Quartus II Project Configuration File and associated settings and configurations files for the project. See Figure 2 on page 26.

Figure 2. Convert MAX+PLUS II Project Dialog Box



Creating a Design

You can use the Quartus II software to create a design in the Quartus II Block Editor or use the Quartus II Text Editor to create a design using the AHDL, Verilog HDL, or VHDL design languages.

The Quartus II software also supports designs created from EDIF Input Files (**.edf**) or Verilog Quartus Mapping Files (**.vqm**) generated by EDA design entry and synthesis tools. You can also create Verilog HDL or VHDL designs in EDA design entry tools, and either generate EDIF Input Files and VQM Files, or use the Verilog HDL or VHDL design files directly in Quartus II projects. For more information on using EDA synthesis tools to generate EDIF Input Files or VQM Files, see "Using Other EDA Synthesis Tools" on page 50 in "Chapter 3: Synthesis."

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

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

Type	Description	Extension
Block Design File	A schematic design file created with the Quartus II Block Editor.	**.bdf**
EDIF Input File	An EDIF version 2 0 0 netlist file, generated by any standard EDIF netlist writer.	**.edf** **.edif**

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

Type	Description	Extension
Graphic Design File	A schematic design file created with the MAX+PLUS II Graphic Editor.	**.gdf**
Text Design File	A design file written in the Altera Hardware Description Language (AHDL).	**.tdf**
Verilog Design File	A design file that contains design logic defined with Verilog HDL.	**.v** **.vlg** **.verilog**
VHDL Design File	A design file that contains design logic defined with VHDL.	**.vh** **.vhd** **.vhdl**
Verilog Quartus Mapping File	A Verilog HDL–format netlist file generated by the Synplicity Synplify software or the Quartus II software.	**.vqm**

Using the Quartus II Block Editor

The Block Editor allows you to enter and edit graphic design information in the form of schematic diagrams and block diagrams. The Quartus II Block Editor reads and edits Block Design Files and MAX+PLUS II Graphic Design Files. You can open Graphic Design Files in the Quartus II software and save them as a Block Design Files.

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

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

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

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

- **Instantiate Altera-provided megafunctions:** The **MegaWizard Plug-In Manager** (Tools menu) allows you to create or modify design files that contain custom variations of megafunctions. These custom megafunction variations are based on Altera-provided megafunctions, including LPM functions. Megafunctions are represented by blocks in Block Design Files. See "Using the MegaWizard Plug-In Manager" on page 33.

- **Insert block and primitive symbols:** Block diagrams use rectangular-shaped symbols, called blocks, to represent design entities and the corresponding assigned signals, and are useful in top-down design. Blocks are connected by conduits that represent the flow of the corresponding signals. You can use block diagrams exclusively to represent your design, or you can combine them with schematic elements.

  The Quartus II software provides symbols for a variety of logic functions—including primitives, library of parameterized modules (LPM) functions, and other megafunctions—that you can use in the Block Editor.

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

Using the Quartus II Text Editor

The Quartus II 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 for any 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, and statements, as well as on megafunctions and primitives.

Using the Quartus II Symbol Editor

The Symbol Editor allows you to view and edit predefined symbols that represent macrofunctions, megafunctions, primitives, or design files. Each Symbol Editor file represents one symbol. For each symbol file, you can choose from libraries containing Altera megafunctions and LPM functions. You can customize these Block Symbol Files, then add the symbols to schematics created with the Block Editor. The Symbol Editor reads and edits Block Symbol Files and MAX+PLUS II Symbol Files (**.sym**), and saves them as Block Symbol Files.

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 (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 more information on using the Verilog HDL and VHDL languages in the Quartus II software, see "Using Quartus II VHDL & Verilog HDL Integrated Synthesis" on page 47 in "Chapter 3: Synthesis."

AHDL is a high-level, modular language that is completely integrated into the Quartus II system. AHDL supports Boolean equation, state machine, conditional, and decode logic. AHDL also allows you to create and use

parameterized functions, and includes full support for LPM functions. AHDL is especially well suited for designing complex combinatorial logic, group operations, state machines, truth tables, and parameterized logic.

For Information About	Refer To
Using the Quartus II Block Editor and Symbol Editor	"Block Editor & Symbol Editor Introduction" in Quartus II Help
Using the Quartus II Text Editor	"Text Editor Introduction" in Quartus II Help
Creating designs in the Quartus II software	Design Entry module in the Quartus II Tutorial

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** (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 2. Altera-Provided Megafunctions & LPM Functions (Part 1 of 2)

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.

Table 2. Altera-Provided Megafunctions & LPM Functions (Part 2 of 2)

Type	Description
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. Additionally, 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 megafunctions and LPM functions.

Using Intellectual Property (IP) Functions

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 applications in communications, digital signal processing (DSP), PCI and other bus interfaces, and memory controllers.

With the OpenCore® and OpenCore Plus features, you can download and evaluate AMPP and MegaCore functions for free prior to licensing and purchasing.

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 web site at **www.altera.com/ipmegastore**.

- **MegaCore Functions:** MegaCore functions are pre-verified HDL design files for complex system-level functions, and are fully parameterizable using the **MegaWizard Plug-In Manager**. MegaCore functions consist of several different design files: a post-synthesis AHDL Include File for design implementation, as well as VHDL or Verilog HDL functional simulation models supplied for design and debugging with EDA simulation tools.

  MegaCore functions are available through the IP MegaStore on the Altera web site, or by using the MegaWizard Portal Extension to the **MegaWizard Plug-In Manager**. A license is not needed to evaluate MegaCore functions, and there is no time limit on evaluation.

- **OpenCore Evaluation Feature:** OpenCore megafunctions are MegaCore functions obtained through the OpenCore evaluation feature. The Altera OpenCore feature allows you to evaluate AMPP and MegaCore functions before purchase. You can use the OpenCore feature to compile, simulate a design and verify functionality and performance of a design, but it does not support programming file generation.

- **OpenCore Plus Hardware Evaluation Feature:** The OpenCore Plus evaluation feature enhances the OpenCore evaluation feature by supporting free RTL simulation and hardware evaluation. RTL simulation support allows you to simulate an RTL model of your MegaCore function in your design. Hardware evaluation support allows you to generate time-limited programming files for a design that includes Altera MegaCore functions. With these files, you can perform board-level design verification before deciding to purchase licenses for the MegaCore functions.

  MegaCore functions supported by the OpenCore Plus feature include a standard OpenCore version and an OpenCore Plus version. The OpenCore Plus license allows you to generate time-limited programming files but not output netlist files.

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** from the Tools menu or from within a Block Design File, or you can run it as a stand-alone utility. Table 3 lists the files generated by the **MegaWizard Plug-In Manager** for each custom megafunction variation you generate.

Table 3. Files Generated by the MegaWizard Plug-In Manager

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

Using the Stand-Alone MegaWizard Plug-In Manager

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

```
qmegawiz ↵
```

Instantiating Megafunctions in the Quartus II Software

You can instantiate Altera megafunctions and LPM functions in the Quartus II software through direct instantiation in the Block Editor, 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 graphical interface for customizing and parameterizing megafunctions, and ensures that you set all megafunction parameters correctly.

Instantiation in Verilog HDL and 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 and 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 need to 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 and 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. Use the **MegaWizard Plug-In Manager** to create and parameterize the megafunction or LPM function.
2. Use the black box file generated by the **MegaWizard Plug-In Manager** to instantiate the function in the EDA synthesis tool.
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. Use the **MegaWizard Plug-In Manager** to create and parameterize the megafunction or LPM function. Make sure you turn on **Generate a Clearbox body** in the **MegaWizard Plug-In Manager**.
2. Use the Verilog or VHDL design file generated by the **MegaWizard Plug-In Manager** to instantiate the function in the EDA synthesis tool.

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

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

For Information About	Refer To
List of ports and parameters for a megafunction	If you are using an IP function, refer to the IP documentation. For Altera megafunctions, refer to Quartus II Help.
Using Altera-provided megafunctions and LPM functions in EDA tools	"Overview: Creating & Instantiating Altera-Provided Functions in Other EDA Tools" in Quartus II Help *Application Note 226 (Synplify & Quartus II Design Methodology)* on the Altera web site *Application Note 225 (LeonardoSpectrum & Quartus II Design Methodology)* on the Altera web site *Application Note 222 (Using Precision RTL Synthesis in the Quartus II Design Methodology)* on the Altera web site
Using Altera-provided megafunctions and LPM functions in the Quartus II software	Design Entry module in the Quartus II Tutorial
Using the **MegaWizard Plug-In Manager** and Altera-provided megafunctions and LPM functions	"Overview: Using the MegaWizard Plug-In Manager" in Quartus II Help
The MegaCore, OpenCore, and OpenCore Plus features and functions	*Application Note 125 (Evaluating AMPP & MegaCore Functions)* on the Altera web site *Application Note 176 (OpenCore Plus Hardware Evaluation)* on the Altera web site

Specifying Initial Design Constraints

Once you have created a project and your design, you can use the **Settings** dialog box, the Assignment Editor, and the Floorplan Editor in the Quartus II software to specify your initial design constraints, such as pin assignments, device options, logic options, and timing constraints. The Quartus II software also provides the **Compiler Settings** wizard (Assignments menu) and **Timing** wizard (Assignments menu) to assist in specifying initial design constraints.

Using the Assignment Editor

The Assignment Editor is the interface for creating and editing assignments in the Quartus II software. Assignments allow you to specify various options and settings for the logic in your design, including location, I/O standard, timing, logic option, parameter, simulation, and pin assignments.

Using the Assignment Editor, you can select an assignment category; use the Quartus II Node Finder to select specific nodes and entities to assign; display information about specific assignments; and add, edit, or delete assignments for selected nodes. You can also add comments to an assignment, and you can view the settings and configuration file in which the assignment appears.

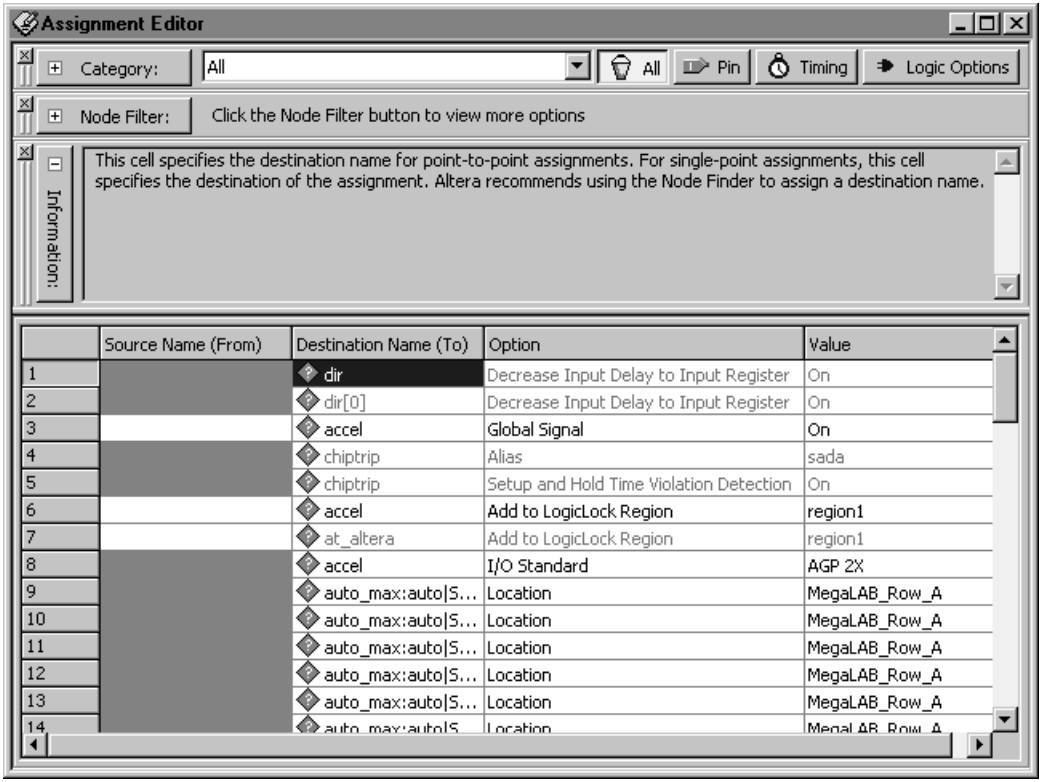
The following steps illustrate the basic flow for using the Assignment Editor to make assignments:

1. Open the Assignment Editor.
2. Select the appropriate category assignment in the **Category** bar.
3. Specify the appropriate node or entity in the **Node Filter** bar, or use the **Node Finder** dialog box to find a specific node or entity.
4. In the spreadsheet that displays the assignments for the current design, add the appropriate assignment information.

The spreadsheet in the Assignment Editor provides applicable drop-down lists or allows you to type assignment information. As you add, edit, and remove assignments, the corresponding Tcl command appears in the Messages window. You can also export the data from the Assignment Editor to a Tcl Script File (**.tcl**) or a spreadsheet-compatible file.

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, but when you view individual assignment categories, the Assignment Editor displays only the assignments that are related to the specific category selected.

Figure 3. The Quartus II Assignment Editor



Using the Settings Dialog Box

You can use the **Settings** dialog box (Assignments menu) to make Compiler, Simulator, and software build settings, timing settings, and modify other project-wide settings.

Using the **Settings** dialog box, you can perform the following types of tasks:

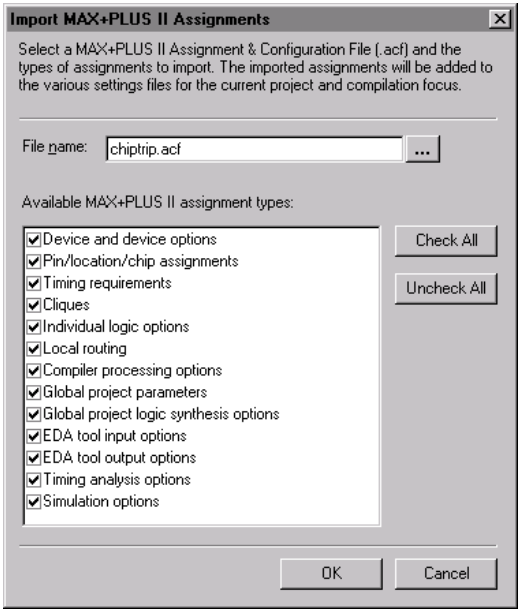
- **Modify project settings**: add and remove files from the project, specify custom user libraries, toolset directories, EDA tool settings, default logic option, and parameter settings.
- **Specify HDL settings**: Verilog HDL and VHDL language versions and Library Mapping Files (**.lmf**).
- **Specify timing settings**: default frequencies for the project or define individual clock settings, delay requirements and path-cutting options, and timing analysis reporting options.
- **Specify Compiler settings**: pin assignments (via the **Assign Pins** dialog box), device options (package, pin count, speed grade), migration devices, Compiler focus, mode, fitting and synthesis options, SignalTap® II settings, Design Assistant settings, and netlist optimization options.
- **Specify Simulator settings**: simulation focus, mode (functional or timing), and time and waveform file options.
- **Specify software build settings**: processor architecture and software toolset, compiler, assembler, and linker settings.
- **Specify HardCopy timing settings**: HardCopy™ timing options and generate HardCopy files.

Importing Assignments

The Quartus II software allows you to easily import the MAX+PLUS II Assignment & Configuration File (**.acf**), which contains MAX+PLUS II project assignments and settings, into your Quartus II project. You can use the **Import MAX+PLUS II Assignments** command (Assignments menu) to import specific types of assignments and add them to the settings and

configurations files in the Quartus II software. You can also import assignments from other EDA synthesis tools using Tcl commands or scripts. See Figure 4.

Figure 4. Import MAX+PLUS II Assignments Dialog Box



Verifying Pin Assignments

The Quartus II software allows you to verify pin assignments—location, I/O bank and I/O standard assignments—with the **Start > Start I/O Assignment Analysis** command (Processing menu). You can use this command at any step in the design process to verify the accuracy of the assignments, allowing you to create your final pin-out faster. You do not need design files to use this command, and can verify pin-outs before design compilation.

Design Methodologies & Design Planning

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

Top-Down versus Bottom-Up Design Methodologies

In the top-down design flow, there is only one output netlist for the entire design, which allows you to perform optimization across design boundaries and hierarchies for the entire design, and is often simpler to manage.

In the bottom-up design methodology, there are separate netlists for each design module. This functionality allows you to compile each module separately and apply different optimization techniques to each module. Modifications to individual modules do not affect the optimizations to other modules. The bottom-up design methodology also facilitates the reuse of design modules in other designs.

Block-Based Design Flow

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

You can use EDA design entry and synthesis tools in the block-based design flow to design and synthesize individual modules, and then incorporate the modules into a top-level design in the Quartus II software, or completely design and synthesize a block-based design in EDA design entry and synthesis tools.

Design Partitioning

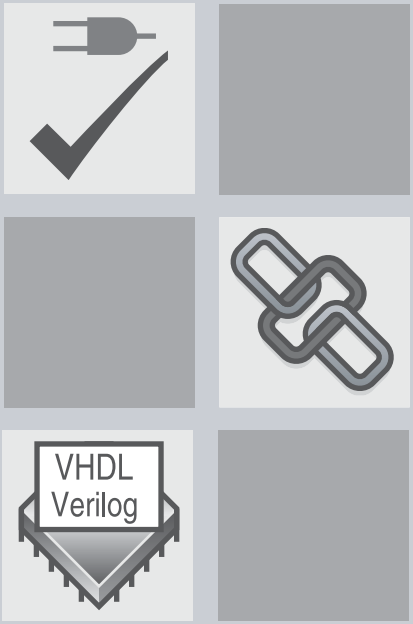
When creating a hierarchical design in the Quartus II software or in other EDA tools, the design is partitioned into separate modules. Considerations for the partitioning of a design during design planning include the following:

- Where to partition the design
- The number of clock and I/O connections between partitions
- Placement of state machines
- Separation of timing-critical functions from noncritical functions
- Limiting the critical path in hierarchical modules
- Registering the inputs and outputs of individual modules

For more information on using LogicLock features and block-based design, refer to "Chapter 6: Block-Based Design" on page 89.

Chapter Three

Synthesis



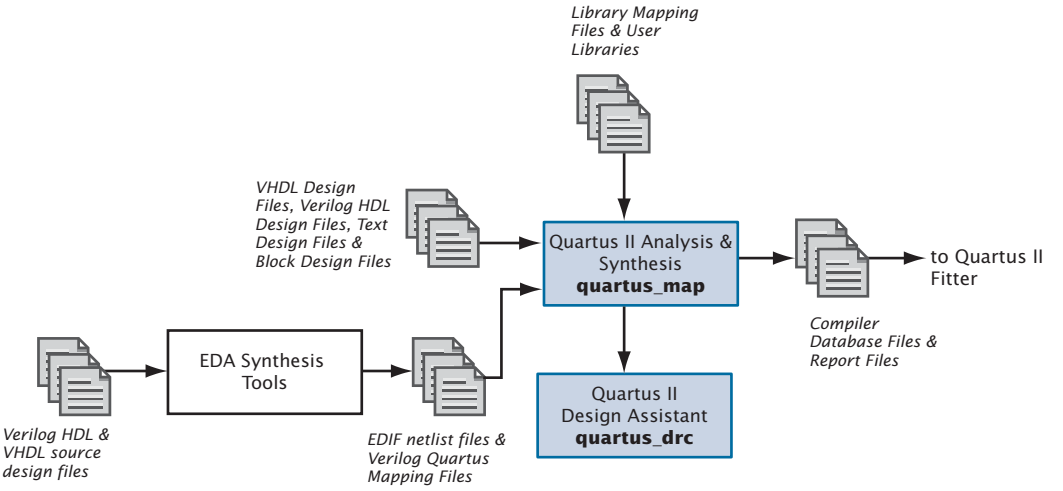
What's in Chapter 3:

3

Introduction

You can use the Quartus® II 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 VHDL Design Files (**.vhd**) or Verilog Design Files (**.v**). If you prefer, you can use other EDA synthesis tools to synthesize your VHDL or Verilog HDL 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. The Quartus II software also allows you to perform an Analysis & Elaboration without running Integrated Synthesis.

Using the quartus_map executable

You can also run Analysis & Synthesis separately at the command prompt or in a script by using the **quartus_map** executable. The **quartus_map** executable will create a new project if it 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:

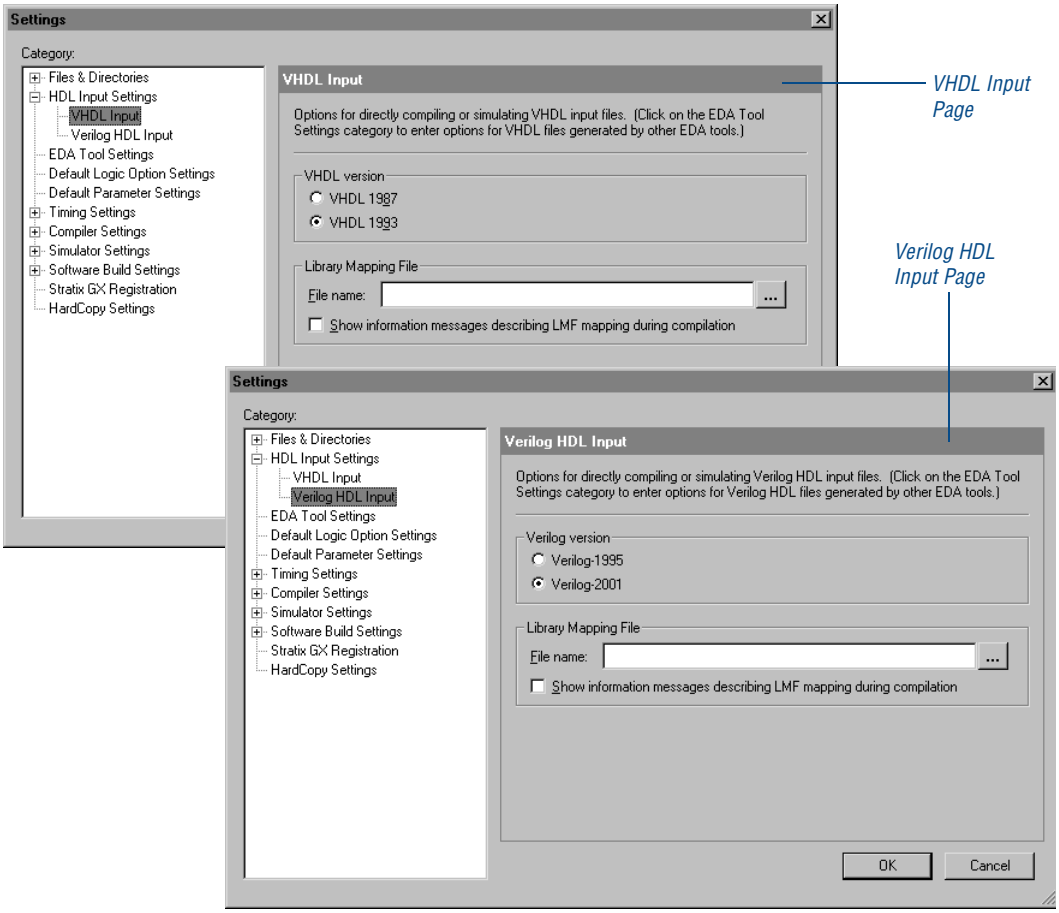
```
quartus_map -h ↵
quartus_map --help ↵
quartus_map --help=<topic name> ↵
```

Using Quartus II VHDL & Verilog HDL Integrated Synthesis

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

Analysis & Synthesis supports the Verilog-1995 standard (IEEE Std. 1364-1995) and most Verilog-2001 standard (IEEE Std. 1364-2001) constructs, 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. You can also specify a Library Mapping File (**.lmf**) that the Quartus II 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 of the **Settings** dialog box (Assignments menu), which are shown in Figure 2 on page 48.

Figure 2. VHDL Input & Verilog HDL Input Pages of Settings Dialog Box



Although most VHDL and Verilog HDL designs will compile successfully in both Quartus II Integrated Synthesis and in other EDA synthesis tools, if you use another EDA tool to instantiate these functions, you need to use a hollow-body or black box file for Altera megafunctions, library of parameterized modules (LPM) functions, and intellectual property (IP) megafunctions. When you are instantiating megafunctions for Quartus II Integrated 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 34 and "Instantiating Megafunctions in EDA Tools" on page 35 in "Chapter 2: Design Entry."

When you create your VHDL or Verilog HDL designs, you should add them to the project. You can add the design files when creating a project by using the **New Project Wizard** (File menu), or by using the **Add/Remove** page of the **Settings** dialog box, or, if you edit the files in the Quartus II Text Editor, you are prompted to add the file to the current project when you save it. When you add files to the project, you should make sure you add them in the order you want Integrated Synthesis to process them. For more information about adding files to a project, refer to "Creating a Project" on page 25 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, device programming, and so on.

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 will minimize the number of resources used. In addition, it replaces operators, such as + or - with modules from the Altera library of parameterized modules (LPM) functions, which are optimized for Altera devices.

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

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

For Information About	Refer To
Verilog HDL constructs supported in the Quartus II software	"Quartus II Verilog HDL Support" in Quartus II Help
VHDL constructs supported in the Quartus II software	"Quartus II VHDL Support" in Quartus II Help
Using Quartus II Integrated Synthesis	*Application Note 238 (Using Quartus II Verilog HDL & VHDL Integrated Synthesis)* on the Altera web site

Using Other EDA Synthesis Tools

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

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

If you have created assignments or constraints using other EDA tools, you can use Tcl commands or scripts to import those constraints into the Quartus II software with your design files. Many EDA tools generate an assignment Tcl script automatically. Table 1 lists the Quartus II–supported EDA synthesis software.

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

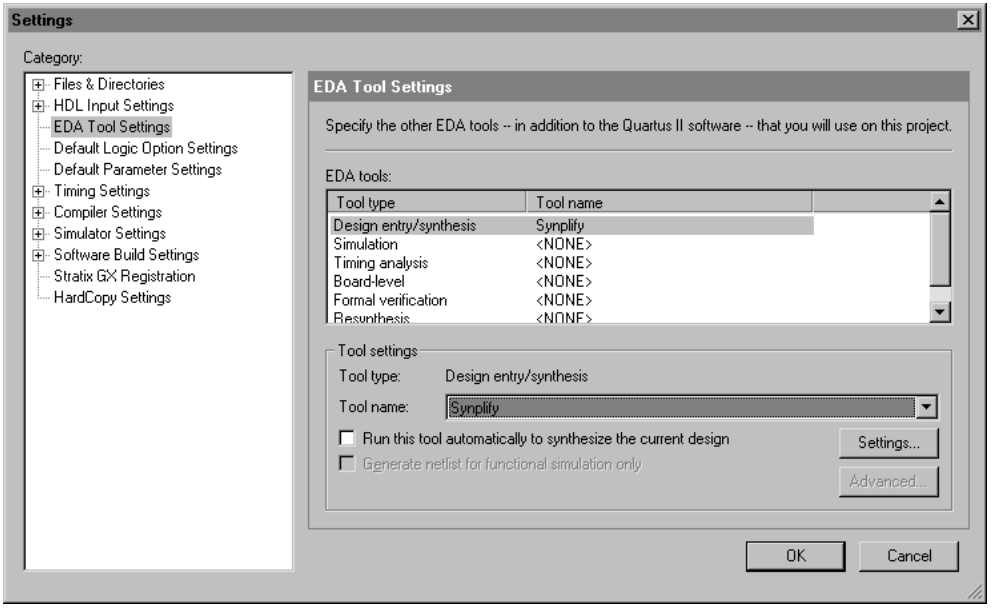
Synthesis Tool Name	EDIF Netlist File (.edf)	Verilog Quartus Mapping File (.vqm)	NativeLink Support
Mentor Graphics Design Architect	✓		
Mentor Graphics LeonardoSpectrum	✓		✓
Mentor Graphics ViewDraw	✓		

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

Synthesis Tool Name	EDIF Netlist File (.edf)	Verilog Quartus Mapping File (.vqm)	NativeLink Support
Mentor Graphics Precision RTL Synthesis	✓		✓
Synopsys Design Compiler	✓		
Synopsys FPGA Compiler II	✓		✓
Synopsys FPGA Express	✓		✓
Synplicity Synplify	✓	✓	✓
Synplify Pro	✓	✓	

In the **EDA Tool Settings** page of the **Settings** dialog box (Assignments menu), you can specify whether a EDA tool that has NativeLink support should be run automatically within the Quartus II software as part of full compilation to synthesize the design. The **EDA Tools Settings** page also allows you to specify other options for EDA tools. See Figure 3.

Figure 3. EDA Tool Settings Page of Settings Dialog Box



Many EDA tools also allow you to run the Quartus II software from within that EDA tool's graphical user interface. Refer to your EDA tool documentation for more information.

For Information About	Refer To
Using Mentor Graphics LeonardoSpectrum	*Application Note 225 (LeonardoSpectrum & Quartus II Design Methodology)* on the Altera web site *Application Note 168 (Getting Started with the LeonardoSpectrum Software)* on the Altera web site
Using Mentor Graphics Precision RTL Synthesis	*Application Note 222 (Using Precision RTL Synthesis in the Quartus II Design Methodology)* on the Altera web site
Using Synplicity Synplify	*Application Note 226 (Synplify & Quartus II Design Methodology)* on the Altera web site

Controlling Analysis & Synthesis

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

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

Using Compiler Directives and Attributes

The Quartus II software supports compiler directives, also called pragmas. You can include compiler directives, such as `translate_on` and `translate_off` directives, in Verilog HDL or VHDL code as comments. These directives are not Verilog HDL or VHDL commands; however, synthesis tools use them to drive the synthesis process in a particular manner. Other tools, such as simulators, ignore these directives and treat them as comments.

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

For Information About	Refer To
Using compiler directives and attributes	"VHDL Language Directives & Attributes" and "Verilog HDL Language Directives & Attributes" in Quartus II Help
Using compiler directives and attributes with Quartus II Integrated Synthesis	*Application Note 238 (Using Quartus II Verilog HDL & VHDL Integrated Synthesis)* on the Altera web site

Using Quartus II Logic Options

Quartus II logic options allow you to set attributes without editing the source code. You can specify Quartus II logic options in the Assignment Editor. Quartus II logic options allow you to preserve registers, specify the logic level for power-up, remove duplicate or redundant logic, optimize for either speed or area, control the fan-out, set the level of encoding for state machines, and control many other options.

For Information About	Refer To
Using Quartus II logic options to control synthesis	"Logic Options," "Creating, Editing, and Deleting Assignments," and "Specifying Settings for Default Logic Options" in Quartus II Help
Creating a logic option assignment	Compilation module in the Quartus II Tutorial
Using Quartus II synthesis options and logic options that affect synthesis	*Application Note 238 (Using Quartus II Verilog HDL & VHDL Integrated Synthesis)* on the Altera web site

Using Quartus II Synthesis Netlist Optimization Options

Quartus II synthesis optimization options allow you to set options for optimizing the netlist during synthesis for many of the Altera device families. These optimization options are in addition 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 **Netlist Optimizations** page of the **Settings** dialog box (Assignments menu) allows you to specify netlist optimization options, which include the following synthesis optimization options:

- **Perform WYSIWYG primitive resynthesis**
- **Perform gate-level register retiming**
- **Allow register retiming to trade off Tsu/Tco with Fmax**

The **Netlist Optimizations** page also includes Fitter netlist optimization and physical synthesis options. For more information about synthesis optimization options, Fitter netlist optimization options, and physical synthesis, refer to "Using Netlist Optimizations to Achieve Timing Closure" on page 118 in "Chapter 8: Timing Closure."

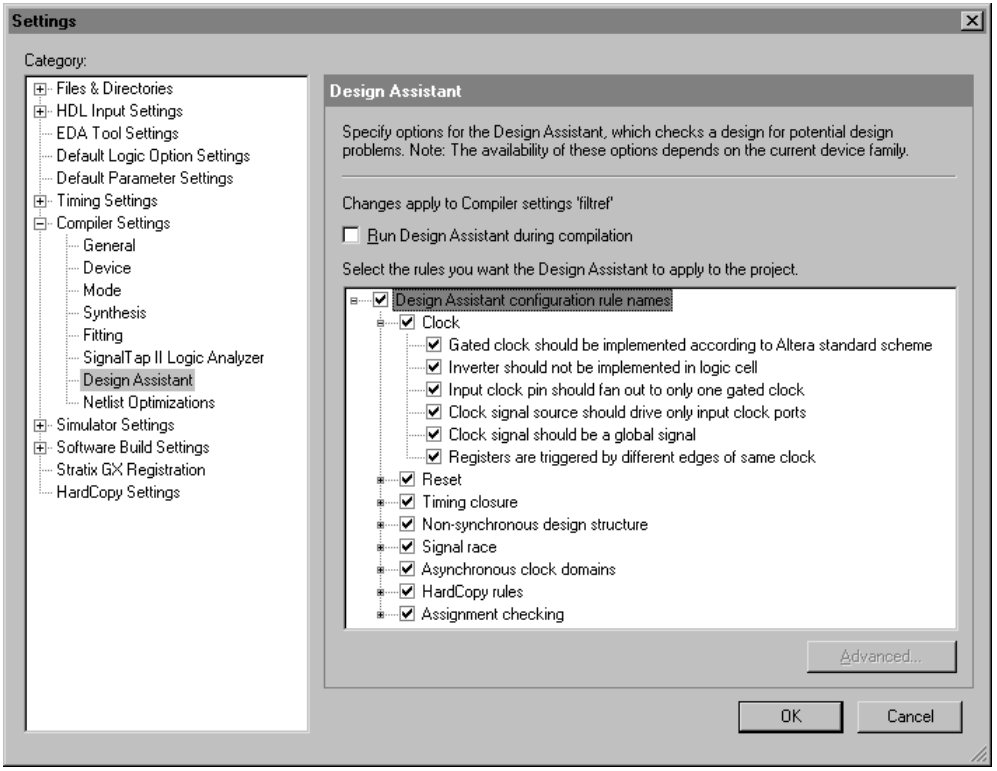
In the **Synthesis** page of the **Settings** dialog box, you can also specify whether you want Analysis & Synthesis to save synthesis results to a VQM File.

For Information About	Refer To
Using Quartus II synthesis and netlist optimization options	*Application Note 198 (Timing Closure with the Quartus II Software)* on the Altera web site *Application Note 297 (Optimizing FPGA Performance Using the Quartus II Software)* on the Altera web site

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 it for HardCopy™ devices. The **Design Assistant** page of the **Settings** dialog box (Assignments menu), allows you to specify which design reliability guidelines you want to use when checking your design. See Figure 4.

Figure 4. Design Assistant Page of Settings Dialog Box



Using the quartus_drc executable

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

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

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

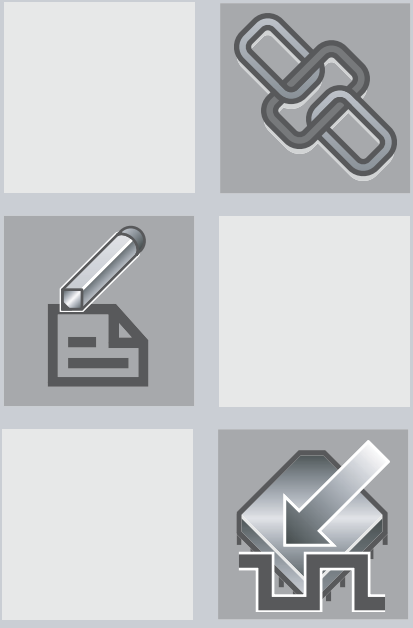
```
quartus_drc -h ↵
quartus_drc -help ↵
quartus_drc --help=<topic name> ↵
```

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

For Information About	Refer To
Using the Quartus II Design Assistant	"Analyzing Designs with the Design Assistant" and "Overview: Using the Design Assistant" in Quartus II Help
Using Quartus II synthesis options, following synchronous design practices, and following coding style guidelines	*Application Note 238 (Using Quartus II Verilog HDL & VHDL Integrated Synthesis)* on the Altera web site "AHDL, VHDL, and Verilog HDL Style Guide" in Quartus II Help

Chapter Four

Simulation



4

What's in Chapter 4:

Introduction

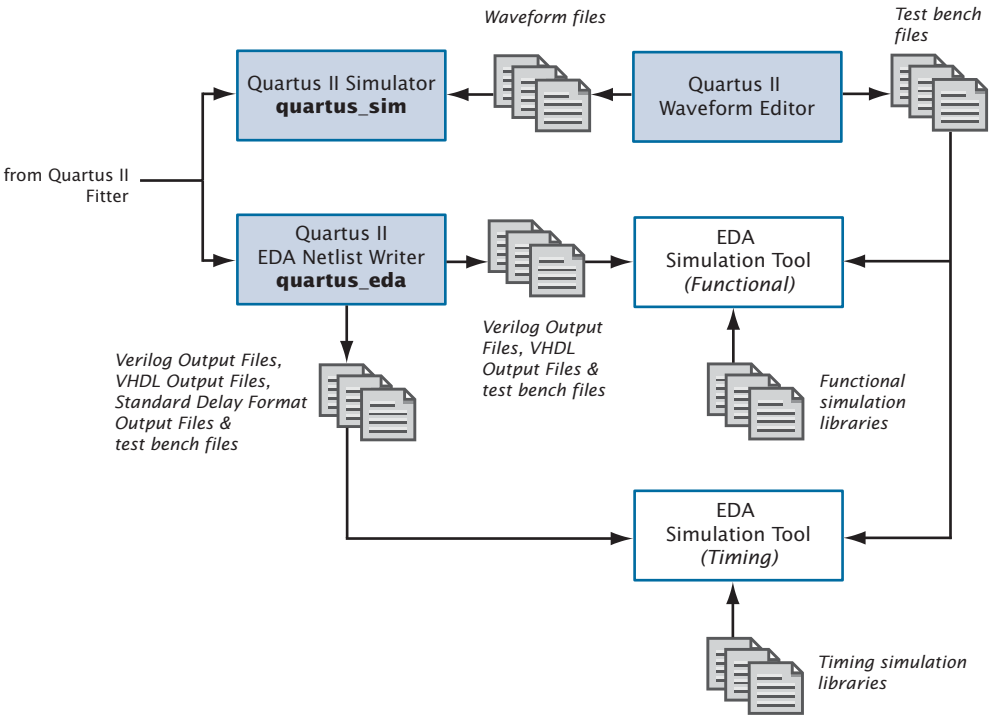
You can perform functional and timing simulation of your design using EDA simulation tools or by using the Quartus® II Simulator.

The Quartus II software provides the following features for performing simulation of designs in EDA simulation tools:

- NativeLink® integration with EDA simulation tools
- Generation of output netlist files
- Functional and timing simulation libraries
- PowerGauge™ power estimation
- Generation of test bench template and memory initialization files

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

Figure 1. Simulation Flow



Simulating Designs with EDA Tools

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

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

Table 1 lists which EDA simulation tools are supported by the NativeLink feature.

Table 1. Quartus II–Supported EDA Simulation Tools

Simulation Tool Name	NativeLink Support
Cadence Verilog-XL	
Cadence NC-Verilog	✓
Cadence NC-VHDL	✓
Model Technology™ ModelSim®	✓
Model Technology ModelSim-Altera	✓
Synopsys Scirocco	✓
Synopsys VCS	
Synopsys VSS	

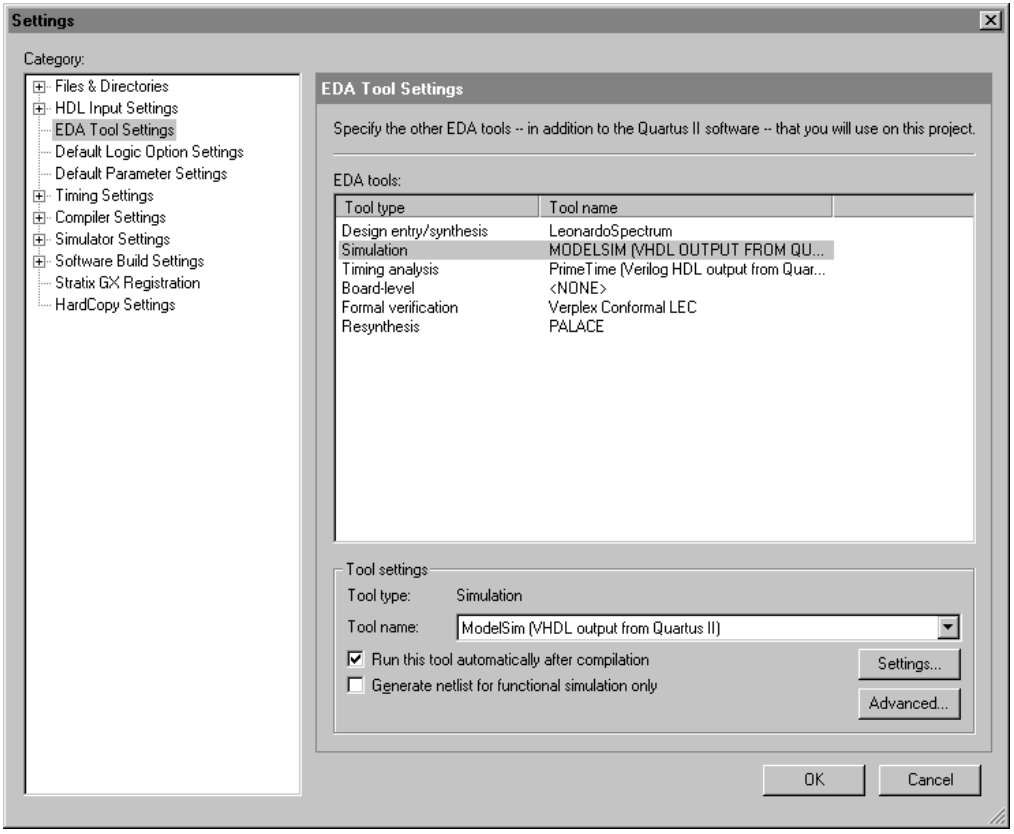
The ModelSim-Altera Software

The Model Technology ModelSim-Altera software is included in Altera® design software subscriptions for behavioral simulation and HDL test bench support.

Specifying EDA Simulation Tool Settings

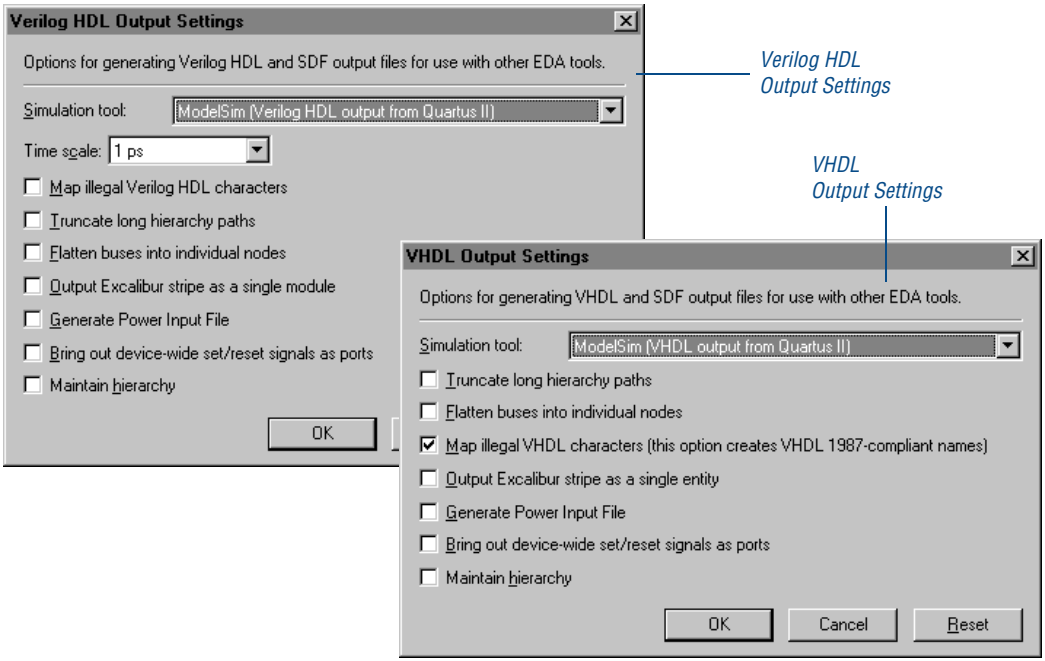
You can select an EDA simulation tool in the **New Project Wizard** (File menu) when you create a new project, or in the **EDA Tool Settings** page of the **Settings** dialog box (Assignments menu). Figure 2 shows the **EDA Tool Settings** page of the **Settings** dialog box.

Figure 2. EDA Tool Settings Page



You can specify other options for the generation of Verilog and VHDL Output Files and the corresponding SDF Output File in the **Verilog HDL/ VHDL Output Settings** dialog boxes, which are available from the **EDA Tool Settings** page. Figure 3 on page 61 shows the **Verilog/VHDL Output Settings** dialog boxes.

Figure 3. Verilog HDL/VHDL Output Settings Dialog Boxes



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 using the **Start > Start EDA Netlist Writer** command (Processing menu). If you are using the NativeLink feature, you can also run a simulation or timing analysis tool after an initial compilation by using the **Run EDA Simulation Tool** and **Run EDA Timing Analysis Tool** commands (Tools menu).

Using the quartus_eda executable

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

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

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

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

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

- **Power Estimation Data**: You can use the ModelSim or ModelSim-Altera software to perform a simulation that includes power estimation data. You can direct the Quartus II software to include power estimation data for the design in the Verilog HDL or VHDL output file. The ModelSim software generates a Power Input File (**.pwf**) that you can use in the Quartus II software to estimate the power consumption of a design.

- **Test Bench Files**: You can create Verilog Test Bench Files (**.vt**) and VHDL Test Bench Files (**.vht**) from a Vector Waveform File (**.vwf**) in the Quartus II Waveform Editor. 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**). You can then export the memory contents as a RAM Initialization File (**.rif**) for use in functional simulation with EDA simulation tools.

Simulation Flow

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

Functional Simulation Flow

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

1. Set up the project in the EDA simulation tool.
2. Create a working library.
3. Compile the appropriate functional simulation libraries with the EDA simulation tool.
4. Compile the design files and test bench files with the EDA simulation tool.
5. Perform the simulation with the EDA simulation tool.

NativeLink Simulation Flow

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

1. Specify EDA tool settings in the Quartus II software, either through the **Settings** dialog box (Assignments menu), or during project setup, using the **New Project Wizard** (File menu).
2. Turn on **Run this tool automatically after compilation** when specifying EDA tool settings.

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

Manual Timing Simulation Flow

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

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

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

3. Launch the EDA simulation tool.

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

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

6. Perform the simulation with the EDA simulation tool.

Simulation Libraries

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

components in EDA simulation tools that are supported by the Quartus II software. Additionally, Altera provides pre-compiled functional and timing simulation libraries for simulation in the ModelSim-Altera software.

Altera provides functional simulation libraries for designs that use Altera megafunctions and standard library of parameterized modules (LPM) functions. Altera also provides pre-compiled versions of the **altera_mf** and **220model** libraries for simulation in the ModelSim software. Table 2 shows the functional simulation libraries for use with EDA simulation tools.

Table 2. Functional Simulation Libraries

Library Name	Description
220model.v **220model.vhd** **220model_87.vhd**	Simulation models for LPM functions (version 2 2 0)
220pack.vhd	VHDL Component Declarations for **220model.vhd**
altera_mf.v **altera_mf.vhd** **altera_mf_87.vhd** **altera_mf_components.vhd**	Simulation models and VHDL Component Declarations for Altera-specific megafunctions

In the Quartus II software, the timing information for specific device architecture entities and Altera-specific megafunctions is located in postrouting 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
Timing Simulation libraries	"Altera Postrouting Libraries" in Quartus II Help
Functional Simulation libraries	"Altera Functional Simulation Libraries" in Quartus II Help
Performing simulation using the ModelSim or ModelSim-Altera software	*Application Note 204 (Using ModelSim-Altera in a Quartus II Design Flow)* on the Altera web site
Performing simulation with the VCS software	*Application Note 239 (Using VCS with the Quartus II Software)* on the Altera web site

Simulating Designs with the Quartus II Simulator

You can use the Quartus II Simulator to simulate any design in a project. Depending on the type of information you need, you can perform a functional simulation to test the logical operation of your design, or you can perform a timing simulation to test both the logical operation and the worst-case timing for the design in the target device.

The Quartus II software allows you to simulate an entire design, or to simulate any part of a design. You can designate any design entity in a project as the simulation focus. When you simulate the design, the Simulator simulates the focus entity and all of its subordinate design entities.

Specifying Simulator Settings

You specify the type of simulation that should be performed, the time period covered by the simulation, the source of vector stimuli, and other simulation options by creating Simulator settings. You can create your own customized groups of Simulator settings by using the **Settings** dialog box (Assignments menu) or **Simulator Settings Wizard** (Processing menu), or you can use the default Simulator settings that are generated automatically each time you create a new project.

Performing a Simulation

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

1. Specify Simulator settings.
2. Create and specify a vector source file.
3. Run the simulation using the **Start > Start Simulation** command (Processing menu) or using the **quartus_sim** executable.

The **Status** window shows the progress of a simulation and the processing time. The **Summary Section** section of the Report window shows the simulation results.

Using the quartus_sim executable

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

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

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

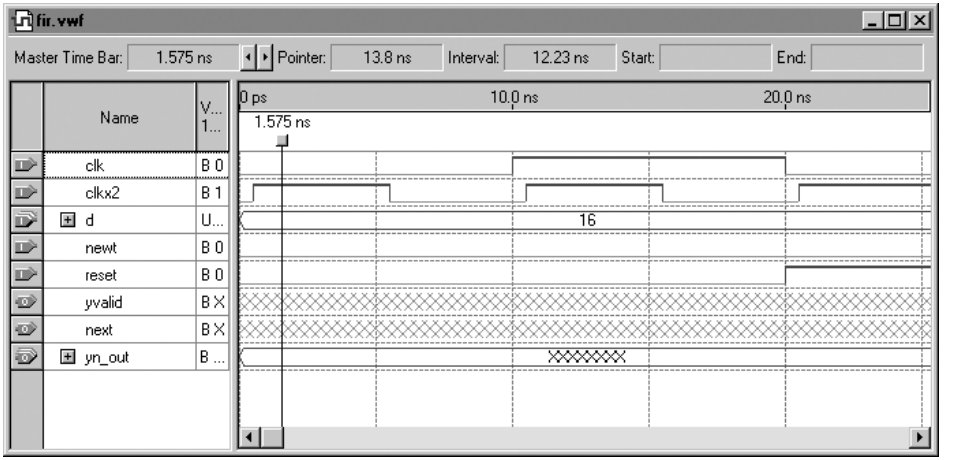
```
quartus_sim -h ↵
quartus_sim --help ↵
quartus_sim --help=<topic name> ↵
```

Creating Waveform Files

The Quartus II Waveform Editor allows you to create and edit input vectors for simulation in waveform format. Using the Waveform Editor, you can add input vectors to the waveform file that describe the behavior of the logic in your design. See Figure 4 on page 68.

The Quartus II software supports waveform files in the Vector Waveform File (**.vwf**), Table File (**.tbl**), Vector File (**.vec**), and Vector Table Output File (**.tbl**) formats. You can save MAX+PLUS® II Simulator Channel Files (**.scf**) as a Table File in the MAX+PLUS II software and then use the Waveform Editor to open and save them as Vector Waveform Files.

Figure 4. The Quartus II Waveform Editor



Performing PowerGauge Power Estimation

The Quartus II software allows you to estimate the power consumed by the current design during timing simulation. You can direct the Simulator to calculate and report in milliwatts (mW) the internal power, I/O pin power, and total power consumed by the design during the simulation period. You can view the results of the PowerGauge power estimation in the Report window.

Simulating Excalibur Designs

You can perform a functional simulation of an Excalibur™ device with the Quartus II Simulator using the bus functional model, or you can use EDA simulation tools to perform a functional or timing simulation of an Excalibur device using either the bus functional model or the full-stripe model. You can also use EDA simulation tools and software debuggers to perform functional, timing, and hardware co-simulation with the Excalibur Stripe Simulator (ESS) model.

Simulating Excalibur Designs in the Quartus II Software

The bus functional model emulates the behavior of the AMBA™ high-performance bus (AHB) in the Excalibur embedded processor stripe of an Excalibur device. It simulates the interactions between the Excalibur embedded processor stripe and the PLD over the Stripe-to-PLD Bridge via the Stripe Master-Port and over the PLD-to-Stripe Bridge via the Stripe Slave-Port.

You can perform a functional simulation of an Excalibur design before compilation and synthesis in the Quartus II software. The bus functional model verifies the functionality of the AHB slaves or masters connected to the Excalibur stripe bridges. You must first generate an uPCore Transaction Model Input File (**.mbus_in**). You can then use the Quartus II Simulator and the bus functional model to perform a functional simulation of the design and generate an uPCore Transaction Model Output File (**.mbus_out**) that contains the bus transactions.

The following steps describe the basic flow to perform a bus functional model functional simulation in the Quartus II software:

1. Create a Master Port High-Level Command File.
2. Use the **exc_bus_translate** utility to create an uPCore Transaction Model Input File.
3. Specify Simulator settings. You must specify the name of the uPCore Transaction Model Input File in the **Options** page of the **Settings** dialog box (Assignments menu).
4. Run the simulation. The Quartus II Simulator simulates the design and generates the uPCore Transaction Model Output File, which shows the results of the bus functional model simulation.

Using the Bus Functional Model with EDA Tools

You can use the bus functional model to perform functional or timing simulation with EDA simulation tools. To use the bus functional model for simulation using other EDA tools, you need to create bus functional model simulation files, which include the stripe-to-PLD and PLD-to-stripe bus transactions.

Once you have generated these files, you can then set up the simulation tool; compile the appropriate libraries, design files, and test bench files; and run the simulation. During simulation, the master commands, addresses, and data values for each transaction are written an output file.

Using the Full Stripe Model with EDA Tools

The Excalibur full-stripe model is a complete Register Transfer Level (RTL) model of the Excalibur embedded processor stripe. It includes the Excalibur embedded processor core and peripherals (for example, SDRAM Interface, Expansion Bus Interface, UART Interface). All stripe components are included in the full stripe model with the exception of the Configuration Logic Master.

You can use the full-stripe model to perform a functional or timing simulation to verify the functionality and timing of all elements in the stripe except the configuration logic. Software code can also be co-simulated with the full-stripe model. You can use the Software Builder to convert the software source files to memory initialization files for use with EDA simulation tools.

Using the ESS Model with EDA Tools

The ESS model is a fast stripe simulation model that emulates the function of the Excalibur embedded processor core, stripe registers, and Stripe-to-PLD and PLD-to-Stripe bus transactions for simulation of Excalibur designs.

The ESS model contains a functionally accurate model of the ARM 922T processor; watchdog timer, timer and interrupt controller; an embedded UART; and an interface to the PLD. It supports booting from flash memory

and configuration from serial files loaded into on-chip memory and has an interface to the ARM Development Suite (ADS) AXD and AWD software debuggers and the GDB GNU debugger.

You can use the ESS model with the ModelSim PE or SE software to perform both a functional hardware simulation to model the Stripe-to-PLD and PLD-to-Stripe bridges and PLD interface, and for software and hardware co-simulation, by connecting the AXD software debugger (provided as a part of the ARM Development Suite) to the Excalibur embedded processor core to control the execution of the software code while simulating the hardware design in the ModelSim software.

The ESS model also allows you to simulate Verilog HDL designs with the ModelSim PE/SE and ModelSim-Altera software, and simulate VHDL designs with the ModelSim SE software. The ESS model can be targeted from the AXD, ADW, Mentor Graphics XRAY, and GNUPro arm-elf-gdb software debuggers on PCs, and the GNUPro arm-elf-gdb debugger on Solaris workstations.

For Information About	Refer To
Bus functional model	*Excalibur Bus Functional Model User Guide* on the Altera web site Excalibur module in the Quartus II Tutorial
Using the bus functional model and full-stripe model in EDA simulation tools	"Overview: Using the ModelSim Software with the Quartus II Software" in Quartus II Help *Excalibur Hardware Design Tutorial* on the Altera web site *Application Note 240 (Simulating Excalibur Systems)* on the Altera web site
Performing simulation or co-simulation with the ESS Model	"Overview: Using the ModelSim Software with the ESS Model" in Quartus II Help *Excalibur Stripe Simulator User Guide* on the Altera web site

Chapter Five

Place & Route

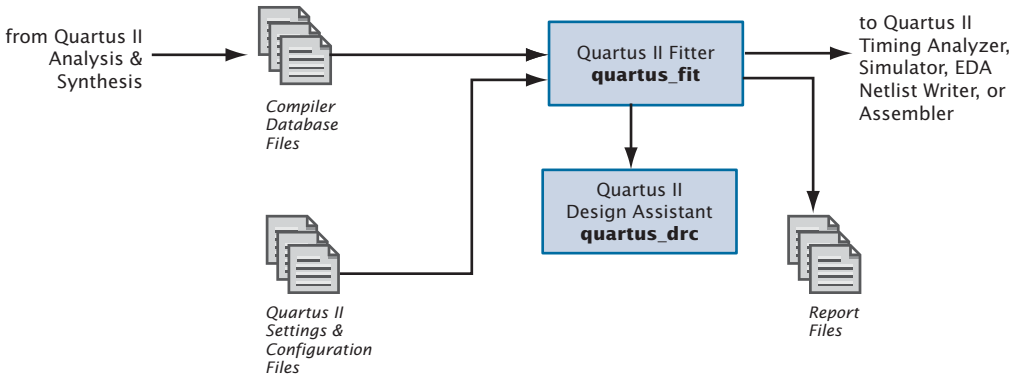
5

What's in Chapter 5:

Introduction

The Quartus® II Fitter, which is also known as the PowerFit™ Fitter, performs place and route, 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 a 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 may 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.

In the **Mode** page under **Compiler Settings** in the **Settings** dialog box (Assignments menu), you can specify whether you want to use a normal compilation or smart compilation. With a "smart" compilation, the Compiler creates a detailed database that can help future compilations run faster, but may consume extra disk space. During a recompilation after a smart compilation, 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. This option is similar to the MAX+PLUS® II **Smart Recompile** command (Processing menu).

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.

Using the quartus_fit executable

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

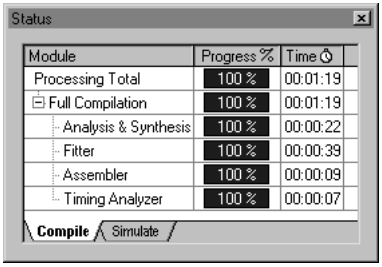
The **quartus_fit** executable creates a separate text-based report file that can be viewed with any text editor.

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

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

The Status window records the time spent processing in the Fitter during project compilation, as well as the processing time for any other modules you may have been running. See Figure 2.

Figure 2. Status Window



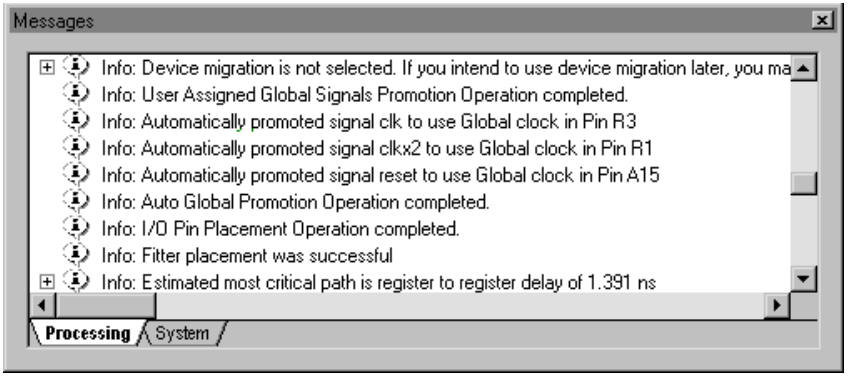
Analyzing Fitting Results

The Quartus II software offers several tools to help you analyze the results of compilation and fitting. The Message window and Report window provide fitting results information. The Floorplan Editor and Chip Editor allow 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 the Fitter generates. See Figure 3.

Figure 3. Messages Window



In the Messages window, you can choose **Help** from the right button pop-up menu to get Help on a particular message.

If you want to filter the messages that appear in the Messages window, you can set options in the **Processing** tab of the **Options** dialog box (Tools menu) that control the display of information and/or warning messages. The right button pop-up menu of the messages window also provides commands that let you control the display of warning messages, critical messages, information messages, and extra information messages.

If the message has a source in the design that you can locate, you can right-click the message and then choose **Locate** (right button pop-up menu). The **Utility Windows > Message Locations** command (View menu) also displays the source(s) of a message.

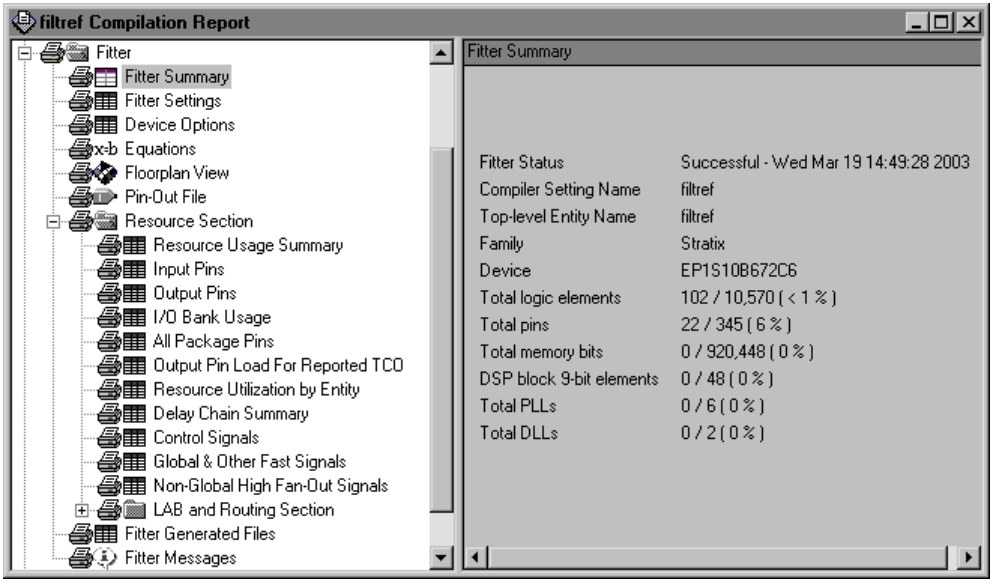
For Information About	Refer To
Viewing messages	"Viewing Messages" in Quartus II Help
Locating the source of a message	Compilation module of the Quartus II Tutorial

Using the Report Window or Report File to View Fitting Results

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

The Report window opens automatically when you run the Fitter or any other compilation or simulation module; however, if the Compiler Tool window is open, the Report window does not open automatically, but clicking on the report file icon for each module displays the report for that module. When the Fitter is processing the design, the Report window continuously updates with new information. If you stop the Fitter, the Report window contains only the information created prior to the point at which you stopped the Fitter. See Figure 4 on page 78.

Figure 4. Fitter Section of the Report Window



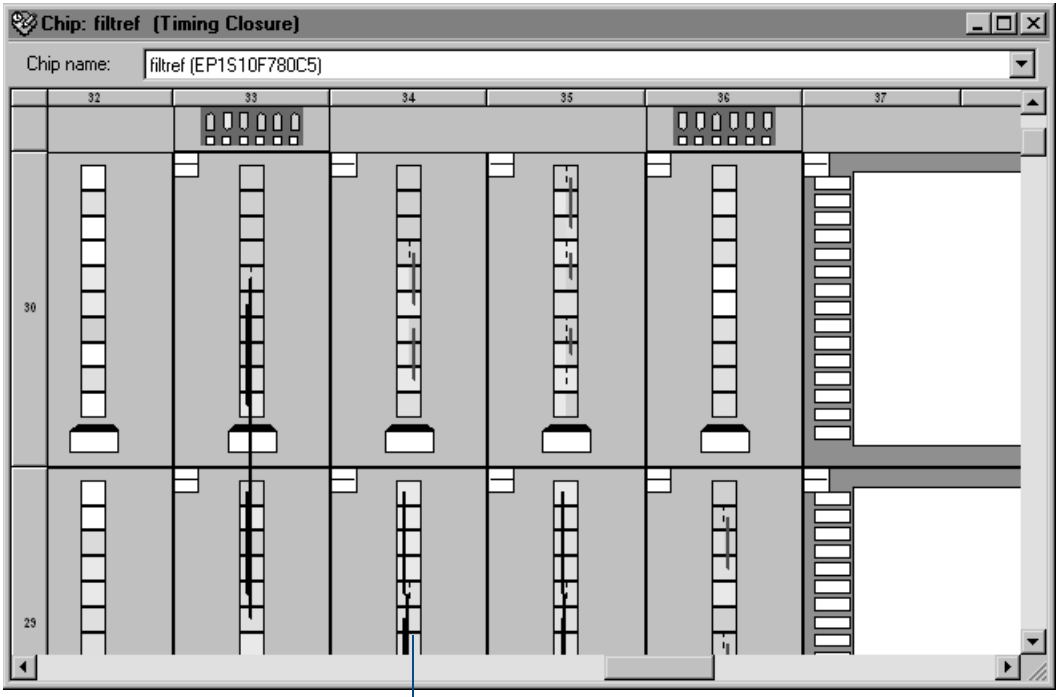
The Quartus II software automatically generates text and HTML versions of the Report window, depending on options you specify in the **Processing** tab of the **Options** dialog box (Tools menu).

For Information About	Refer To
Report Window sections	"Report Window & File Format" in Quartus II Help
Using the Report Window	"Overview: Viewing the Results of a Compilation or Simulation in the Report Window" in Quartus II Help
Viewing the compilation report	Compilation module of the Quartus II Tutorial

Using the Floorplan Editor to Analyze Results

After you run the Fitter, the Floorplan Editor displays the results of place and route. You can view the non-editable (read-only) Last Compilation floorplan, which shows the resource assignments and routing made during the last compilation. In addition, you can back-annotate the fitting results to preserve the resource assignments made during the last compilation. The editable Timing Closure floorplan allows you to view logic placement made by the Fitter and/or user assignments, make LogicLock™ region assignments, and view routing congestion. See Figure 5.

Figure 5. The Floorplan Editor



If you compile a design that is targeted for an Excalibur™ device, you can also view the Excalibur embedded processor stripe in the Floorplan Editor. The stripe is located between the logic cells and pins, and contains interfaces to the embedded logic for the microprocessor, as well as to the dual-port RAM.

Resource usage in the Floorplan Editor is color coded. Different colors represent different resources, such as unassigned and assigned pins and logic cells, unrouted items, MegaLAB™ structures, columns, and row FastTrack® fan-outs. The Floorplan Editor also provides different floorplan views that show the pins and interior structure of the device.

To edit assignments in the Floorplan Editor, you can click a resource assignment and drag it to a new location. While dragging a resource in the Floorplan Editor, 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 Floorplan Editor also allows you to view the node fan-out and node fan-in for specific structures, or view the paths between specific nodes. If necessary, you can change or delete resource assignments. For more information on using the Floorplan Editor, refer to "Using the Timing Closure Floorplan" on page 114 in "Chapter 8: Timing Closure."

If you want to view more fitting details and make additional fitting adjustments, the Chip Editor reveals additional details about design placement and routing that are not visible in the Floorplan Editor, and allows you to make changes by using the Resource Property Editor and Change Manager. For more information, refer to "Chapter 11: Engineering Change Management" on page 147.

For Information About	Refer To
Viewing assignment and routing information in the Floorplan Editor	"Overview: Working with Assignments in the Floorplan Editor" and "Overview: Viewing Routing Information" in Quartus II Help
Viewing the fit in the Last Compilation floorplan	Compiler module of the Quartus II Tutorial

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 (Assignments menu) allows you to specify which design reliability guidelines to use when checking your design. Refer to "Using the Design Assistant to Check Design Reliability" on page 55 of "Chapter 3: Synthesis." for more information.

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 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 Floorplan Editor or the Assignment Editor in order to control place and route. You may want to use the Floorplan Editor to edit assignments because it gives you a graphical view of the device and its features. If you want to create new location assignments, you may want to use the Assignment Editor, which allows you to create several node-specific assignments at once. In addition to using the Floorplan Editor 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 (Assignments menu). For more information about specifying initial design constraints, refer to "Specifying Initial Design Constraints" on page 38 in "Chapter 2: Design Entry."

Once you create an assignment, you can edit it in the Assignment Editor or the Floorplan Editor. After compilation, you can use the Floorplan Editor to edit existing resource assignments to pins, logic cells, rows, columns,

regions, MegaLAB structures, and LABs. You can use the Floorplan Editor, the LogicLock Regions window, or the **LogicLock Region Properties** dialog box (Assignments menu) to assign nodes or entities to LogicLock regions.

The Floorplan Editor provides different views of the device and helps you make precise assignments to specific locations. You can also view equations and routing information, and demote assignments by dragging and dropping assignments to various regions in the Regions window. If your design has too many constraints that prevent it from fitting in the device, you may also be able to optimize fitting by removing some of the location assignments and allowing the Fitter to place the logic.

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
- Logic options that affect fitting

Setting Fitter Options

The **Fitting** page of the **Settings** dialog box (Assignments menu) 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 specify whether you want the Fitter to use standard fitting, which works hardest to meet your $\mathbf{f_{MAX}}$ timing constraints, to use the fast fit feature, which improves the compilation speed but may reduce the $\mathbf{f_{MAX}}$, or to limit fitting to only one attempt, which may also reduce the $\mathbf{f_{MAX}}$.

Setting Fitting Optimization & Physical Synthesis Options

Quartus II fitting optimization options allow you set options for performing physical synthesis to optimize the netlist during fitting. You can specify Quartus II Fitter optimization options in the **Netlist Optimizations** page of the **Settings** dialog box (Assignments menu).

Fitter optimization options include the following options:

- **Perform physical synthesis for combinatorial logic**
- **Physical synthesis for registers:**
  - **Perform register duplication**
  - **Perform register retiming**

For more information about fitting optimization and physical synthesis options, refer to "Using Netlist Optimizations to Achieve Timing Closure" on page 118 in "Chapter 8: Timing Closure."

Setting 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 (Assignments menu) and can specify global default logic options in the **Settings** dialog box (Assignments menu). 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 combinatorial function in the same logic cell, also known as "register packing," or limit the length of carry chains, cascade chains, and parallel expander chains.

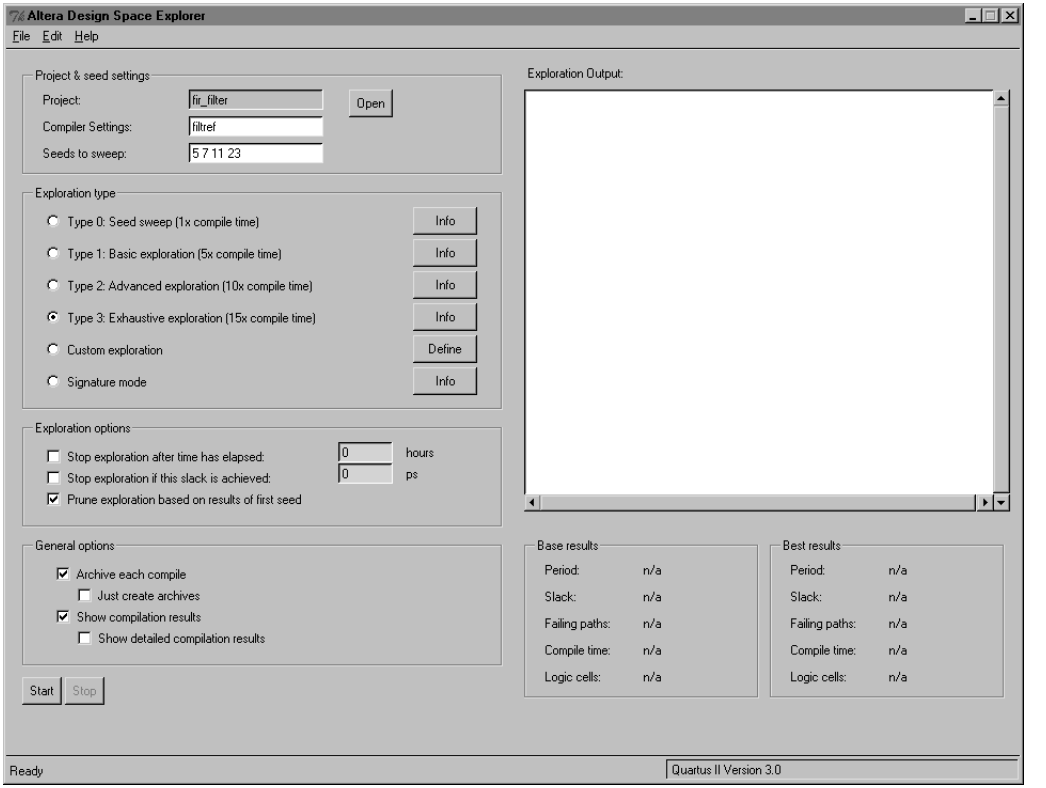
For Information About	Refer To
Using Quartus II logic options to control place and route	"Logic Options," "Creating, Editing & Deleting Assignments," and "Specifying Settings for Default Logic Options" in Quartus II Help
Using Quartus II Fitter optimization options	"Optimizing Netlists During Synthesis & Fitting" in Quartus II Help

Using the Design Space Explorer

Another way to control Quartus II fitting is to use the Design Space Explorer (DSE), which is a Tcl script, **dse.tcl**, that you can run from the command line with the **quartus_sh** executable to optimize your design. 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 you will allow DSE to make, your optimization goals, the target device, and the allowable compilation time. See Figure 6.

Figure 6. Design Space Explorer Interface



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:

```
quartus_sh -t dse ↵
```

For help on DSE options, type `quartus_sh --help=dse ↵` at command prompt, or choose **Show Documentation** (Help menu) in the DSE window.

You should not run DSE from within the Quartus II graphical user interface.

DSE provides several exploration modes, which are listed under **Exploration type** in the DSE window:

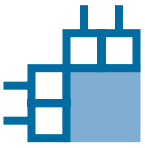
- **Type 0: Seed Sweep**
- **Type 1: Basic Exploration**
- **Type 2: Advanced Exploration**
- **Type 3: Exhaustive Exploration**
- **Custom**
- **Signature mode**

The Type 0, Type 1, Type 2, Type 3, and Custom modes are Parameter Sweep modes. These modes allow you to specify a target that must be met for the project, such as a set of timing requirements, so you can improve timing. The different modes allow you to specify the degree of effort you want DSE spend in fitting the design in a way that will achieve the target; 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. If you do not specify a target, DSE tries to find the best $\mathbf{f_{MAX}}$ result possible.

Signature mode allows you to explore the effect of a single parameter on your design, and their trade-offs for $\mathbf{f_{MAX}}$, slack, compile time, and area. In Signature mode, 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.

For Information About	Refer To
Using the Design Space Explorer	*Application Note 198 (Timing Closure with the Quartus II Software)* on the Altera web site
Parameters and settings for optimizing performance	*Application Note 297 (Optimizing FPGA Performance Using the Quartus II Software)* on the Altera web site

Performing Incremental Fitting



If you have made a change that affects only a few nodes, you can also avoid running a full compilation by using incremental fitting. Incremental fitting allows you to run the Fitter module of the Compiler in a mode that attempts to preserve the fitting results of the previous compilation. Incremental fitting attempts to reproduce the results of the previous compilation as closely as possible, which prevents unnecessary changes in the timing results, and because it reuses the results of the previous compilation, it usually requires less compilation time than standard fitting.

You can turn on incremental fitting by choosing the **Start > Start Incremental Fitting** command (Processing menu).

Running Incremental Fitting from the Command Line

You can also run incremental fitting with the **quartus_fit** executable by typing the following command at the command prompt:

```
quartus_fit <project name> --incremental_fitting ↵
```

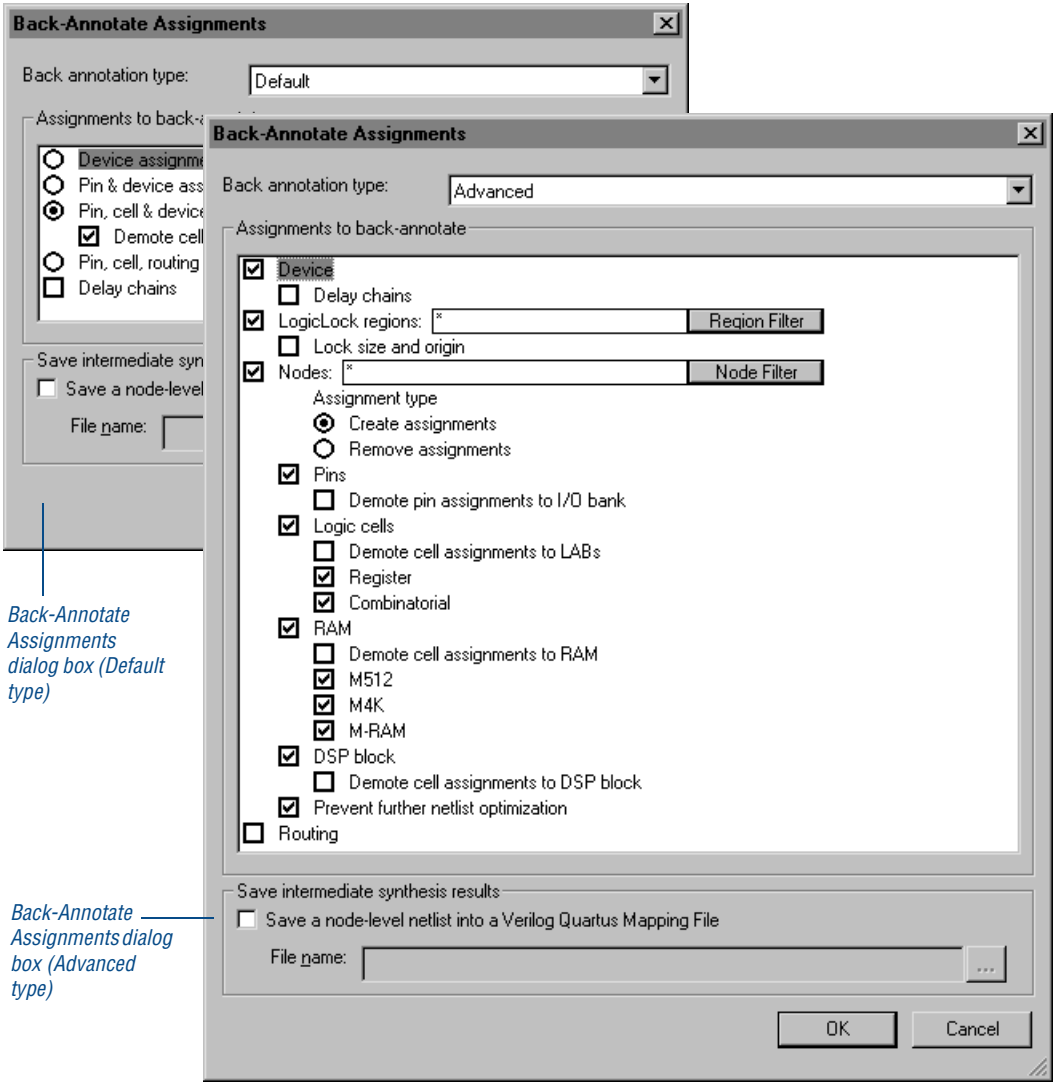
Preserving Assignments through Back-Annotation

You can preserve resource assignments from the last compilation by back-annotating assignments to any device resource. You can back-annotate all the resource assignments in a project; you can also back-annotate the size

and location of LogicLock regions. You can specify assignments to back-annotate in the **Back-Annotate Assignments** dialog box (Assignments menu).

The **Back-Annotate Assignments** dialog box allows you to select the type of back-annotation: Default type or Advanced type. See Figure 7.

Figure 7. Back-Annotate Assignments Dialog Box

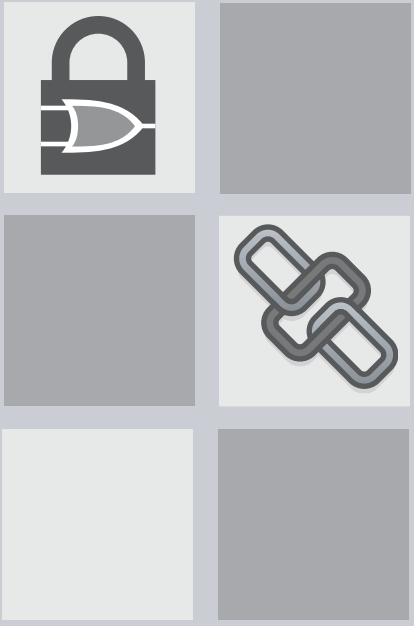


The **Back-Annotate Assignments** (Default type) dialog box allows you to "demote" pin and/or logic cell assignments to less restrictive location assignments so that you can allow the Fitter more freedom in rearranging assignments. The **Back-Annotate Assignments** (Advanced type) dialog box allows you to do everything that the Default back-annotation type allows you to do, as well as back-annotate LogicLock regions, and optionally the nodes and routing within them. The Advanced back-annotation type also provides many options for filtering based on region, path, resource type, and so on, and allows you to use wildcards. You should use only one type of back-annotation or the other, but not both. If you are not sure which type to use, Altera recommends that you use the Advanced back-annotation type for most situations because it offers more options, especially if you are using LogicLock regions. For more information about using back-annotation with LogicLock regions, refer to "Back-Annotating LogicLock Region Assignments" on page 96 in "Chapter 6: Block-Based Design."

For Information About	Refer To
Back-annotating location assignments	"Back-Annotating Assignments for a Project" in Quartus II Help
Back-annotating LogicLock region assignments	"Back-Annotating a LogicLock Region" in Quartus II Help
Back-annotating LogicLock placement	LogicLock module of the Quartus II Tutorial

Chapter Six

Block-Based Design



6

What's in Chapter 6:

Introduction

The Quartus® II LogicLock™ feature enables a block-based design flow by allowing you to create modular designs, designing and optimizing each module separately before incorporating it into the top-level design.

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

LogicLock regions support team-oriented, block-based design by enabling you to optimize logic blocks individually, and then import them and their placement constraints into a larger design. The LogicLock methodology also promotes module reuse, because modules can be developed separately and then constrained to LogicLock regions to be used in other designs with no loss in performance, allowing you to leverage resources and shorten design cycles.

Quartus II Block-Based Design Flow

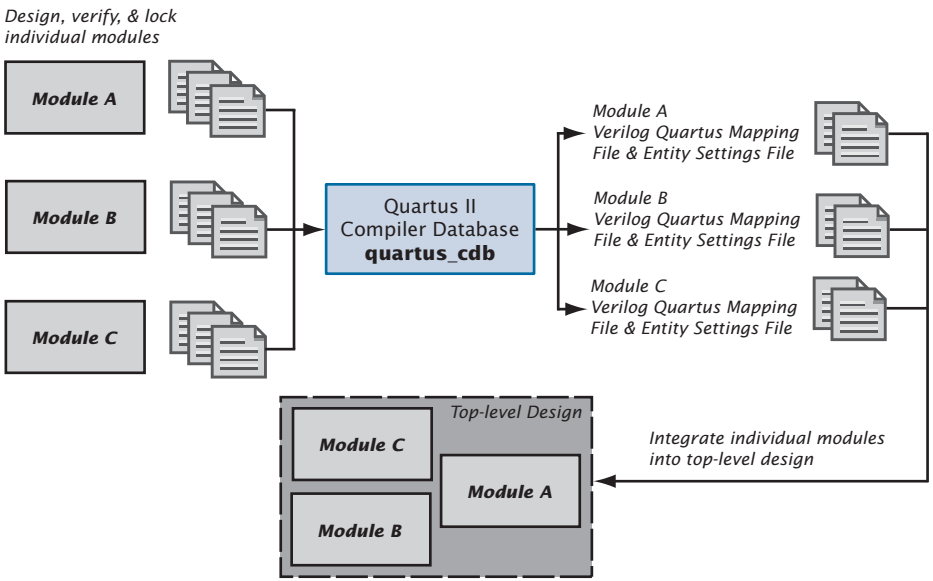
In traditional top-down design flows, there is only one netlist for the design. In a top-down design flow, individual modules of the design can have different performance in the overall design than when implemented on their own. In bottom-up block-based design flows, there are separate netlists for each module. This allows designers to create block-based designs, where each module is optimized independently and then incorporated into the top-level design. You can use block-based design in the following design flows:

- **Modular design flow**: In the modular design flow, you divide a design into a top-level design that instantiates separate submodules. You can develop each module separately and then incorporate each module into the top-level design. Placement is determined manually by you or by the Quartus II software.
- **Incremental design flow**: In the incremental design flow, you create and optimize a system, and then add future modules with little or no effect on the performance of the original system.

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

In all three design flows, you can preserve performance at all levels of development by partitioning designs into functional blocks, organized according to the physical structure of the circuit or by critical paths. Figure 1 illustrates the basic block-based design flow.

Figure 1. Block-Based Design Flow



Using LogicLock Regions

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

Table 1. LogicLock Region Properties

Property	Values	Behavior
State	Floating or Locked	Floating regions allow the Quartus II software to determine the region's location on the device. Locked regions have a user-defined location. Locked regions are shown in the floorplan with a solid boundary and floating regions shown by a dashed boundary in the floorplan. A locked region must have a fixed size.
Size	Auto or Fixed	Auto-sized regions allow the Quartus II software to determine the appropriate size of a region given its contents. Fixed regions have a user-defined shape and size.
Reserved	On or Off	The reserved property allows you to define whether the Quartus II software can use the resources within a region for entities that are not assigned to the region. If the reserved property is on, only items assigned to the region can be placed within its boundaries.
Enforcement	Hard or Soft	Soft regions give more deference to timing constraints, and allow some entities to leave a region if it improves the performance of the overall design. Hard regions do not allow the Quartus II software to place contents outside the boundaries of the region.
Origin	Any Floorplan Location	The origin defines the placement of the LogicLock region in the floorplan.

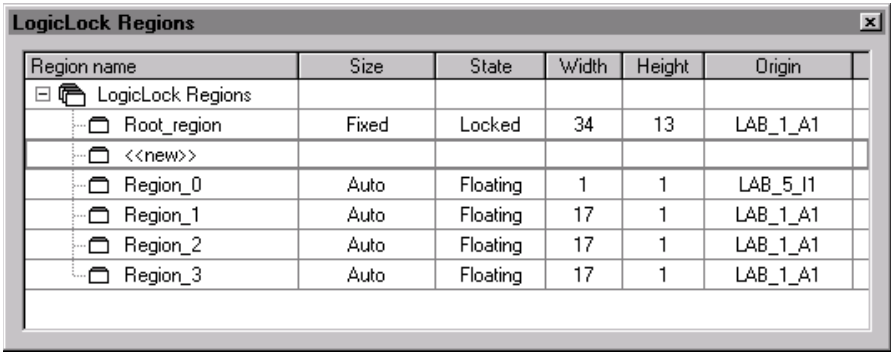
With the LogicLock design flow, you can define a hierarchy for a group of regions by declaring parent and child regions. The Quartus II software places child regions completely within the boundaries of a parent region. You can lock a child module relative to its parent region without constraining the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

You can create and modify LogicLock regions using the Floorplan Editor, the **LogicLock Region Properties** dialog box (right button pop-up menu), the **Hierarchy** tab of the Project Navigator, or by using Tcl scripts. Once you have created a LogicLock region, you can also place logic within that region with the Assignment Editor. All LogicLock attributes and constraint information (clock settings, pin assignments, and relative placement information) are stored in the Entity Settings File (**.esf**) for the specific project.

You can use the Floorplan Editor to create and edit LogicLock region assignments. You can draw LogicLock regions in the Timing Closure floorplan with the **Create New Region** button and then drag and drop nodes from the floorplan view, the Node Finder, or the **Hierarchy** tab of the Project Navigator.

After you have created a LogicLock region, you can use the LogicLock Regions window to view all of the LogicLock regions in your design, including size, state, width, height, and origin. You can also edit and add new LogicLock regions. See Figure 2.

Figure 2. LogicLock Regions Window

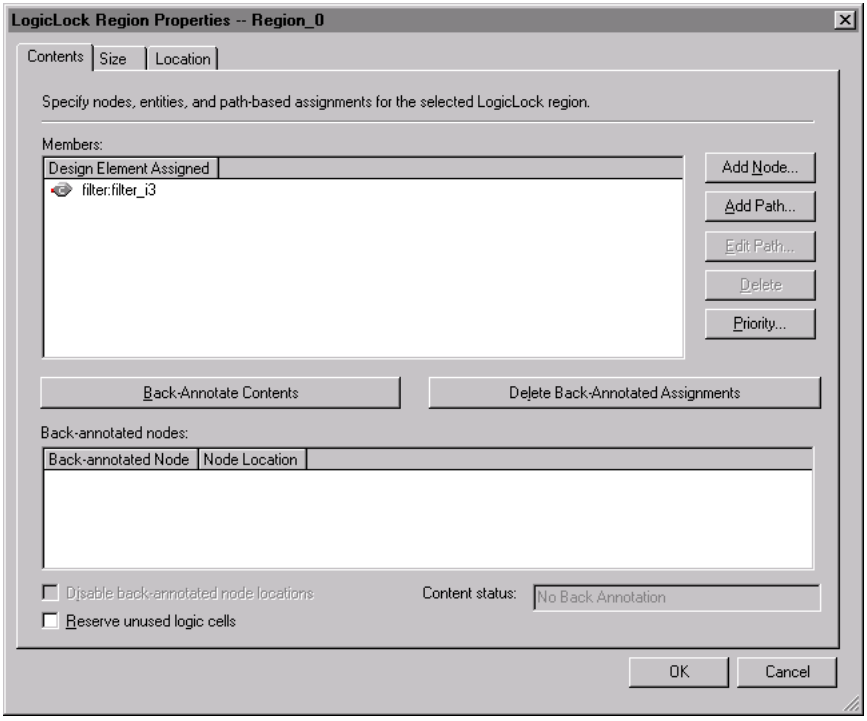


You can also use the **LogicLock Regions Properties** dialog box to edit existing LogicLock regions, open the **Back-Annotate Assignments** dialog box to back-annotate all nodes in a LogicLock region, view information on the LogicLock regions in the design, and determine which regions contain illegal assignments.

In addition, you can add path-based assignments (based on source and destination nodes), wildcard assignments, and Fitter priority for path-based and wildcard assignments to LogicLock regions. Setting the priority allows

you to specify the order in which the Quartus II software resolves conflicting path-based and wildcard assignments. You can open the **Priority** dialog box from the **LogicLock Region Properties** dialog box. See Figure 3.

Figure 3. LogicLock Region Properties Dialog Box



After you have performed an 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 Floorplan Editor.

Altera also provides LogicLock Tcl commands to assign LogicLock region content, at the command line or via 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	*Application Note 161* (*Using the LogicLock Methodology in the Quartus II Design Software*) on the Altera web site "Overview: Using LogicLock Regions" in Quartus II Help The LogicLock module in the Quartus II Tutorial

Saving Intermediate Synthesis Results

You can save synthesis results for individual entities in conjunction with the block-based LogicLock design flows by creating a Verilog Quartus Mapping File (**.vqm**) for an entity in a design, with a corresponding Entity Settings File that contains the LogicLock constraint information for the entity.

You can design a block of custom logic or instantiate a block of pre-verified Intellectual Property (IP), make assignments to that block, verify functionality and performance, lock the block to maintain this placement and performance, and then export the block to be imported into another design. In this way, blocks can be designed, tested, and optimized individually and can maintain their performance when integrated into a larger design.

In addition, by saving intermediate synthesis results into a VQM File and replacing the entity with the VQM File in the project where you import the assignments, you ensure that the node names synthesized in the new project correspond to the node names in the imported assignments.

The following steps describe the basic flow for saving intermediate synthesis results, back-annotating assignments, and exporting and importing Entity Settings Files for designs that contain LogicLock regions:

1. Specify that the Quartus II software should save the intermediate synthesis results for the current compilation focus as a VQM File by specifying the name of the VQM File in the **Synthesis** page of the **Settings** dialog box (Assignments menu) or in the **Back-Annotate Assignments** (Advanced type) dialog box (Assignments menu).

2. Create LogicLock regions.

3. Compile the design to generate the VQM File or use the **Start > Start VQM Writer** command (Processing menu) to generate the VQM File after an initial compilation.

4. Use the Advanced **Back-Annotate Assignments** (Advanced type) dialog box to lock the logic placement in the LogicLock region(s).

5. Export the LogicLock region assignments to an Entity Settings File by using the **Export LogicLock Regions** command (Assignments menu).

6. You can then instantiate the module in the VQM File into a top-level design and import the LogicLock region assignments by using the **Import LogicLock Regions** command (Assignments menu).

Using the quartus_cdb executable

You can also save intermediate synthesis results as a VQM File, back-annotate assignments, and export and import LogicLock regions separately at the command prompt or in a script by using the **quartus_cdb** executable.

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

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

Back-Annotating LogicLock Region Assignments

You can use the **Back-Annotate Assignments** (Advanced Type) command to lock the logic placement into LogicLock regions in a design before exporting the assignments for use in a top-level design. Back-annotation allows you to maintain the performance of a LogicLock region when importing the region and its assignments into a top-level design.

You must use the **Back-Annotate Assignments** (Advanced Type) command to back-annotate LogicLock region assignments, and you can also use it to back-annotate designs that do not include LogicLock region assignments.

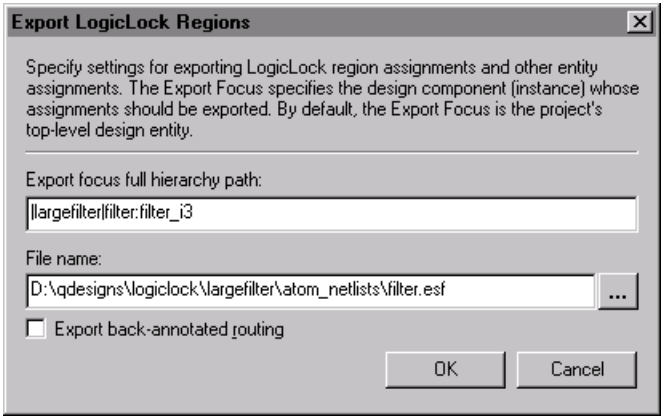
For more information on back-annotating assignments, refer to "Preserving Assignments through Back-Annotation" on page 86 in "Chapter 5: Place & Route."

Exporting & Importing LogicLock Assignments

The **Export LogicLock Regions** and **Import LogicLock Regions** dialog boxes (Assignments menu) allow you to optimize entities individually using LogicLock region assignments and preserve your optimization when you instantiate those entities in a top-level design.

When you export LogicLock region assignments, the Quartus II software writes all LogicLock region assignments, other Entity Settings File assignments, and I/O standard assignments that apply to the specific entity instance to an Entity Settings File that you specify in the **Export LogicLock Regions** dialog box. By default, the Quartus II software exports the LogicLock region assignments for the entire design. You can specify sub-design entities to export in the **Export focus full hierarchy path** box. See Figure 4.

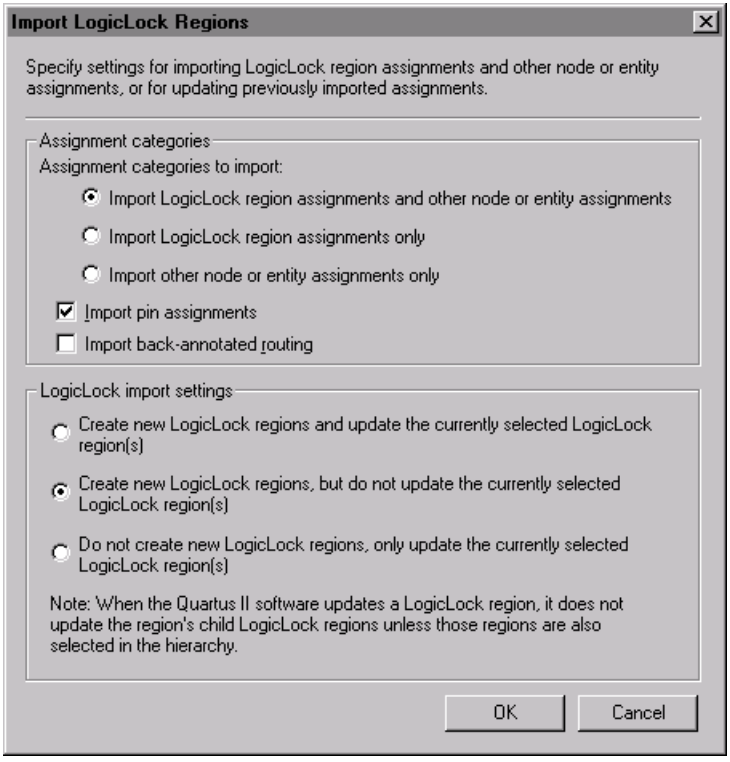
Figure 4. Export LogicLock Regions Dialog Box



When you import LogicLock region assignments, the Quartus II software traverses the compilation hierarchy, starting at the current compilation focus. If the current project contains multiple instances of a lower-level entity, the Quartus II software instantiates the assignments imported for that lower-level entity once for each instance.

To prevent placement conflicts, the Quartus II software assigns imported top-level LogicLock regions to floating locations. However, it preserves the location of imported child regions relative to their parents. When importing LogicLock regions, you can specify the assignment categories to import and specify whether to create new LogicLock regions, update the currently selected LogicLock region(s), or both. See Figure 5.

Figure 5. Import LogicLock Regions Dialog Box



For Information About	Refer To
Saving intermediate synthesis results as a VQM File, back-annotating assignments, and exporting and importing LogicLock region assignments	*Application Note 161 (Using the LogicLock Methodology in the Quartus II Design Software)* on the Altera web site "Overview: Saving Intermediate Synthesis Results" in Quartus II Help "Overview: Using LogicLock Regions" in Quartus II Help The LogicLock Module in the Quartus II Tutorial

Using LogicLock with EDA Tools

The block-based LogicLock design flow supports modules that are created and optimized in EDA design entry and synthesis tools and then imported as separate modules in the Quartus II software. You use the EDA design entry and synthesis tool to create separate netlist files (EDIF Input Files (**.edf**) or VQM Files) for modules in a design hierarchy. You can then use the Quartus II software to place each netlist file into a separate LogicLock region in a top-level design. Once in the Quartus II software, you can make changes, optimize, and resynthesize specific modules in the design by using the EDA tool to update the corresponding netlist file, without affecting the other modules in the design.

The Mentor Graphics LeonardoSpectrum, Synplicity Synplify, and Synopsys FPGA Compiler II, and Mentor Graphics Precision RTL Synthesis software provide customized features for using these tools in the block-based LogicLock design flow.

For Information About	Refer To
Using LogicLock with EDA synthesis tools	*Application Note 165 (Synplify and Quartus II LogicLock Design Flow)* on the Altera web site *Application Note 164 (LeonardoSpectrum and Quartus II LogicLock Design Flow)* on the Altera web site *Application Note 171 (FPGA Compiler II BLIS & the Quartus II LogicLock Design Flow)* on the Altera web site *Application Note 222 (Using Precision RTL Synthesis in the Quartus II Design Methodology)* on the Altera web site

Chapter Seven

Timing Analysis

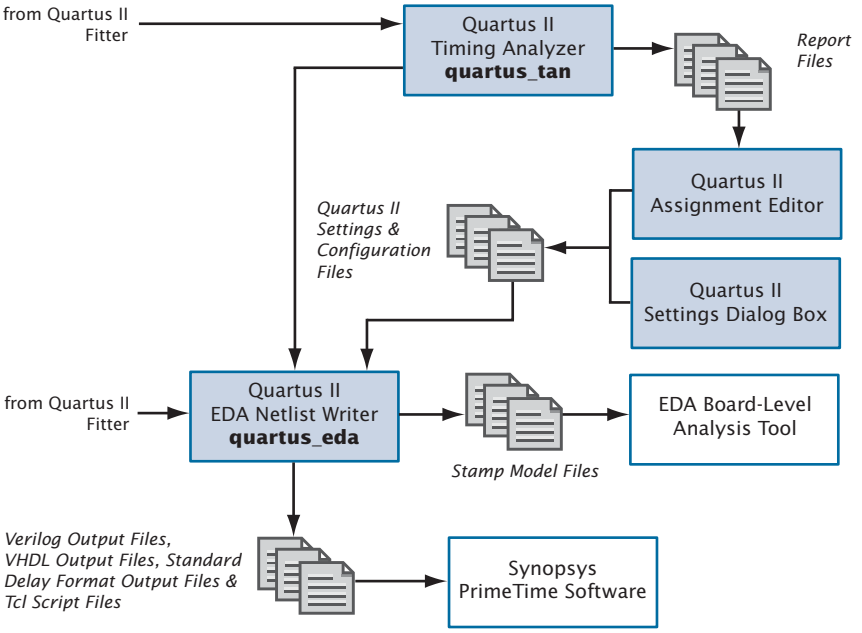
7

What's in Chapter 7:

Introduction

The Quartus® II Timing Analyzer allows you to analyze the performance of all logic in your design and helps to guide the Fitter to meet the timing requirements in your design. By default, the Timing Analyzer runs automatically as part of a full compilation to observe and report timing information such as the setup times (t_{SU}), hold times (t_H), clock-to-output delays (t_{CO}), pin-to-pin delay (t_{PD}), maximum clock frequencies (f_{MAX}), slack times, and other timing characteristics for the design. You can use the information generated by the Timing Analyzer to analyze, debug, and validate the timing performance of your design. You can also use the Quartus II Timing Analyzer to perform a minimum timing analysis, which reports the best-case timing results to verify clock-to-pad delays for signals driving off-chip. Figure 1 shows the timing analysis flow.

Figure 1. Timing Analysis Flow



Performing Timing Analysis in the Quartus II Software

The Timing Analyzer automatically performs timing analysis on your design during a full compilation. The following guidelines describe some of the tasks that you can accomplish with the Quartus II Timing Analyzer:

- Specify initial project-wide and individual timing requirements, using the **Timing** wizard (Assignments menu), the **Settings** dialog box (Assignments menu), and the Assignment Editor.
- Perform the timing analysis during a full compilation or separately after an initial compilation.
- View the timing results using the Report Window, Timing Closure floorplan, and `list_paths` Tcl command.

Specifying Timing Requirements

Timing requirements allow you to specify the desired speed performance for the entire project, for specific design entities, or for individual entities, nodes, and pins.

You can use the **Timing** wizard (Assignments menu) to help you to create initial project-wide timing settings. Once you have specified initial timing settings, you can modify settings either by using the **Timing** wizard again, or by using the **Settings** dialog box (Assignments menu).

You can make individual timing settings with the Assignment Editor (Assignments menu). After specifying project-wide timing assignments and/or individual timing assignments, you can run a timing analysis by compiling the design, or by running the Timing Analyzer separately after an initial compilation.

If you do not specify timing requirement settings or options, the Quartus II Timing Analyzer will run the analyses using default settings. By default, the Timing Analyzer calculates and reports the f_{MAX} of every register, the t_{SU} and t_H of every input register, the t_{CO} of every output register, the t_{PD} between all pin-to-pin paths, slack times, hold times, minimum t_{CO}, and minimum t_{PD} of the current design entity.

Using the **Settings** dialog box or the **Timing** wizard, you can specify the following timing requirements and other options:

- Overall frequency requirement for the project, or settings for individual clock signals
- Delay requirements, minimum delay requirements, and path-cutting options
- Reporting options, including the number or source and destination registers and exclude paths
- Timing-driven compilation options

Specifying Project-Wide Timing Settings

Project-wide timing settings include maximum frequency, setup time, hold time, clock-to-output delay and pin-to-pin delay, and minimum timing requirements. You can also set project-wide clock settings and multiple clock domains, path-cutting options, and default external delays.

Table 1. Project-Wide Timing Settings (Part 1 of 2)

Requirement	Description
f_{MAX} (maximum frequency)	The maximum clock frequency that can be achieved without violating internal setup (t_{SU}) and hold (t_H) time requirements.
t_{SU} (clock setup time)	The length of time for which data that feeds a register via its data or enable input(s) must be present at an input pin before the clock signal that clocks the register is asserted at the clock pin.
t_H (clock hold time)	The length of time for which data that feeds a register via its data or enable input(s) must be retained at an input pin after the clock signal that clocks the register is asserted at the clock pin.
t_{CO} (clock-to-output delay)	The time required to obtain a valid output at an output pin that is fed by a register after a clock signal transition on an input pin that clocks the register.
t_{PD} (pin to pin delay)	The time required for a signal from an input pin to propagate through combinatorial logic and appear at an external output pin.

Table 1. Project-Wide Timing Settings (Part 2 of 2)

Requirement	Description
minimum t_{CO} (clock-to-output delay)	The minimum time required to obtain a valid output at an output pin that is fed by a register after a clock signal transition on an input pin that clocks the register. This time always represents an external pin-to-pin delay.
minimum t_{PD} (clock-to-output delay)	Specifies the minimum acceptable pin-to-pin delay, that is, the time required for a signal from an input pin to propagate through combinatorial logic and appear at an external output pin.

Specifying Individual Timing Assignments

You can make individual timing assignments to individual entities, nodes, and pins with the Assignment Editor. Individual timing assignments override project-wide requirements (if they are more stringent). The Assignment Editor supports point-to-point timing assignments and wildcards to identify specific nodes when making assignments.

The timing requirements that you enter for pins and nodes are saved in the Entity Settings File (**.esf**) for the top-level entity in the current hierarchy.

You can make the following types of individual timing assignments in the Timing Analyzer:

- **Individual 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
- **Multicycle paths**: paths between registers that require more than one clock cycle to become stable. You can set multicycle paths to instruct the Timing Analyzer to adjust its measurements and avoid incorrect setup or hold time violations.
- **Cut paths**: by default, the Quartus II software will cut 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 will also cut paths between unrelated clock domains if individual clock assignments are set but there is no defined relationship between the clock assignments. You can also define cut paths for specific paths in the design.

- **Minimum delay requirements**: individual t_H, minimum t_{CO}, and minimum t_{PD} timing requirements for specific nodes or groups. You can make these assignments to specific nodes or groups to override project-wide minimum timing requirements.
- **External delay**: specifies the delay of a signal from an external register (outside the device) to the input pin.
- Individual t_{SU}, t_{PD}, and t_{CO} requirements on specific nodes in the design.

Performing a Timing Analysis

Once you have specified timing settings and assignments, you can run the Timing Analyzer by performing a full compilation.

After compilation is complete, you can rerun timing analysis separately by using the **Start > Start Timing Analyzer** command (Processing menu) or run a minimum timing analysis by choosing **Start > Start Minimum Timing Analysis** (Processing menu).

Using the quartus_tan executable

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

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

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

```
quartus_tan -s ↵
```

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

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

For Information About	Refer To
Specific timing settings and performing a timing analysis in the Quartus II Software	"Overview: Using the Timing Analyzer" in Quartus II Help *Application Note 123 (Using Timing Analysis in the Quartus II Software)* on the Altera web site Timing Analysis module of the Quartus II Tutorial

Viewing Timing Analysis Results

After you run a timing analysis, you can view the timing analysis results in the **Timing Analyzer** folder of the Compilation Report. You can then list the timing paths to validate circuit performance, determine critical speed paths and paths that limit the design's performance, and make additional timing assignments. Additionally, you can use the `list_paths` Tcl command to locate and view information on any delay path in the design.

Users familiar with MAX+PLUS® II timing reporting can find the timing information, such as the delay information from the Delay Matrix, in the Timing Analyzer sections of the Compilation report.

Using the Report Window

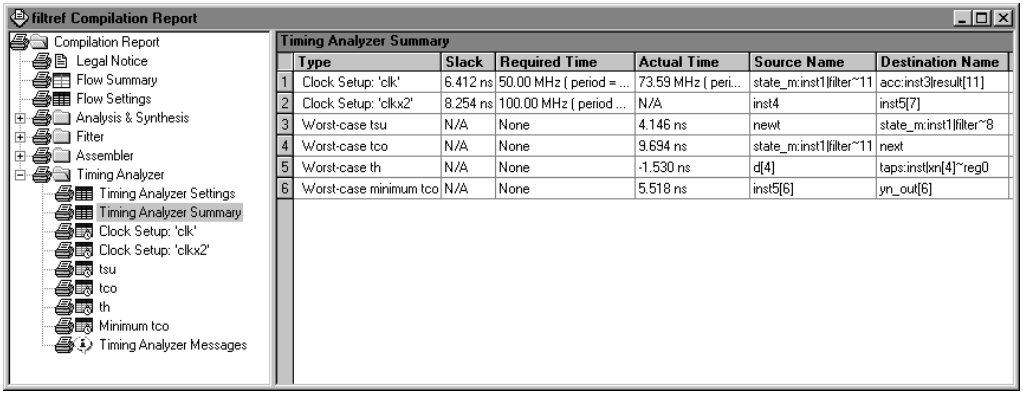
The Timing Analysis sections in the Report Window list the reported timing information for clock setup and clock hold; t_{SU}, t_H, t_{PD}, t_{CO}; minimum pulse width requirements; any timing assignments that were ignored during the timing analysis; and any messages generated by the Timing Analyzer. By default, the Timing Analyzer also reports the best-case minimum clock-to-output times and best-case minimum point-to-point delays.

The Report Window reports the following types of information for timing analysis:

- Settings for timing requirements
- Slack and minimum slack
- Source and destination clock names
- Source and destination node names

- Required and actual point-to-point times
- Required hold relationships
- Actual f_{MAX}

Figure 2. Timing Analysis Results in the Report Window



Making Assignments & Viewing Delay Paths

You can access the Assignment Editor, **List Paths** and **Locate in Timing Closure Floorplan** commands directly from the Timing Analyzer sections in the Report Window to make individual timing assignments and view delay path information. In addition, you can use the `list_paths` Tcl command to list delay path information.

You can use the Assignment Editor to make an individual timing assignment on any path in a Timing Analyzer report. This feature allows you to easily make point-to-point assignments on paths.

The following steps describe the basic flow for making individual timing assignments in the Assignment Editor:

1. In the **Category** bar, click **Timing** to indicate the category of assignment you wish to make.

2. Click the **Destination Name (To)** cell in the spreadsheet and use the Node Finder to find a node, or type a node name and/or wildcard character that identifies the destination node you want to assign.

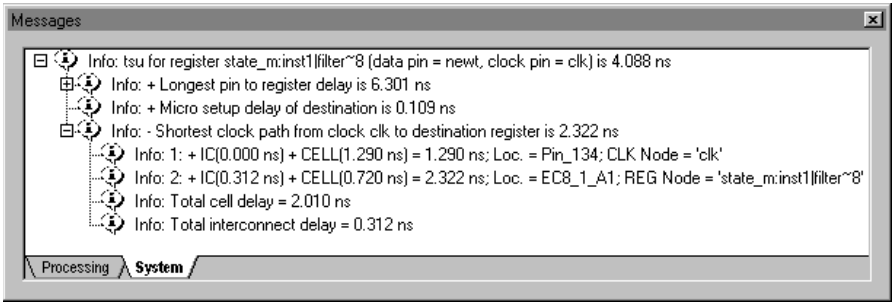
3. Click the **Source Name (From)** cell in the spreadsheet and use the Node Finder to find a node, or type a node name and/or wildcard character that identifies the source node you want to assign.

4. In the spreadsheet, double-click the **Option** cell and select the timing assignment you wish to make. For assignments that require a value, double-click the **Value** cell and type or select the appropriate assignment value.

You can also use the **Locate in Timing Closure Floorplan** command (Project menu) to locate a path in the Timing Closure floorplan, which allows you to take advantage of the Timing Closure floorplan features to make assignments to a specific path. For more information on using the Timing Closure floorplan, refer to "Using the Timing Closure Floorplan" on page 114 in "Chapter 8: Timing Closure."

You can use the **List Paths** command (right button pop-up menu) to display the intermediate delays of any path in a Timing Analyzer report panel in the Messages window. This allows you to find pin-to-pin, register-to-register, and clock-to-output-pin delay paths, and display information about any delay path in the design that appears in the Report Window. See Figure 3.

Figure 3. Output from List Paths Command



The `list_paths` Tcl command, which you can use in the **quartus_tan** API and the Quartus II Tcl Console, allows you to specify any point-to-point path and view the delay information. You can specify the number of paths to

report, the type of path (including minimum timing paths), and use wildcards to identify source and destination nodes. This option reports information in the same manner as the **List Path** command. See Figure 4.

Figure 4. Sample Output from list_paths Command

```
---------------------------------------------------------------------------------------------
Path Number: 1
tco from clock clock to destination pin gt1 through register auto_max:auto|street_map[0] is 8.869 ns
---------------------------------------------------------------------------------------------
    + Longest clock path from clock clock to source register is 2.799 ns^M
        1: + IC(0.000 ns) + CELL(0.619 ns) = 0.619 ns; Loc. = Pin_M2; CLK Node = 'clock'
        2: + IC(1.638 ns) + CELL(0.542 ns) = 2.799 ns; Loc. = LC_X31_Y1_N9; REG Node = 'auto_max:auto|street_map[0]'
        Total cell delay = 1.161 ns
        Total interconnect delay = 1.638 ns
    + Micro clock to output delay of source is 0.156 ns
    + Longest register to pin delay is 5.914 ns
        1: + IC(0.000 ns) + CELL(0.000 ns) = 0.000 ns; Loc. = LC_X31_Y1_N9; REG Node = 'auto_max:auto|street_map[0]'
        2: + IC(0.716 ns) + CELL(0.075 ns) = 0.791 ns; Loc. = LC_X30_Y1_N9; COMB Node = 'rtl~261'
        3: + IC(0.518 ns) + CELL(0.366 ns) = 1.675 ns; Loc. = LC_X30_Y1_N8; COMB Node = 'rtl~17'
        4: + IC(1.350 ns) + CELL(2.889 ns) = 5.914 ns; Loc. = Pin_AA13; PIN Node = 'gt1'
        Total cell delay = 3.330 ns
        Total interconnect delay = 2.584 ns
---------------------------------------------------------------------------------------------
```

Performing Timing Analysis with EDA Tools

The Quartus II software supports timing analysis and minimum timing analysis using the Synopsys PrimeTime software on UNIX workstations and board-level timing analysis using the Mentor Graphics® BLAST or Tau board-level verification tools.

You can generate the necessary output files for performing timing analysis in EDA timing analysis tools by specifying the appropriate timing analysis tool in the **Settings** dialog box (Assignments menu) or in the **New Project Wizard** (File menu) when creating a project, and then performing a full compilation. You can also generate the files by using the **Start > Start EDA Netlist Writer** command (Processing menu) after an initial compilation.

Using the quartus_eda executable

You can also run the EDA Netlist Writer to generate the necessary output files separately at the command prompt or in a script by using the **quartus_eda** executable. You must run the Quartus II Fitter executable **quartus_fit** before running the EDA Netlist Writer.

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

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

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

Using the PrimeTime Software

The Quartus II software generates a Verilog Output File or VHDL Output File, a Standard Delay Format Output File (**.sdo**) that contains timing delay information, and Tcl Script File that sets up the PrimeTime environment. If you are performing a minimum timing analysis, the Quartus II software uses the minimum delay information generated by the Timing Analyzer in the SDF Output File for the design.

Using 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 (SDC) 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 through the **Settings** dialog box (Assignments menu), or during project setup, using the **New Project Wizard** (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 (**.tcl**) to set up the PrimeTime environment.

4. Perform timing analysis in the PrimeTime software.

Using the BLAST and Tau Software

The Quartus II software generates Stamp model files which can be imported into the BLAST or Tau software to perform board-level timing verification.

The following steps describe the basic flow generate Stamp model files:

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

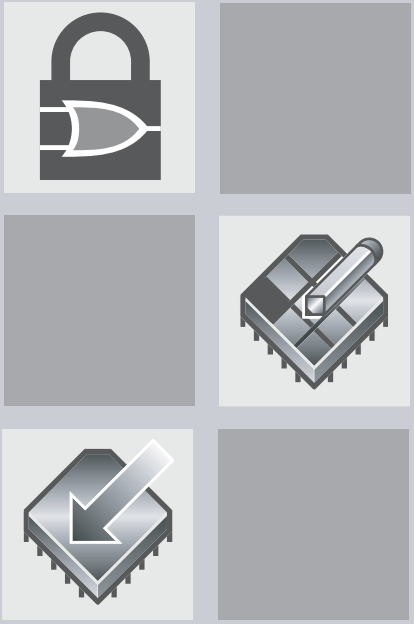
2. Compile the design in the Quartus II software to generate the Stamp model files. The Quartus II software places the files in a tool specific directory.

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

For Information About	Refer To
Using the Synopsys PrimeTime software with the Quartus II software	"Overview: Using the PrimeTime Software with the Quartus II Software" in Quartus II Help
Using the Innoveda BLAST and Mentor Graphics Tau software with the Quartus II software	"Overview: Using the BLAST Software with the Quartus II Software" in Quartus II Help "Overview: Using the Tau Software with the Quartus II Software" in Quartus II Help

Chapter Eight

Timing Closure



8

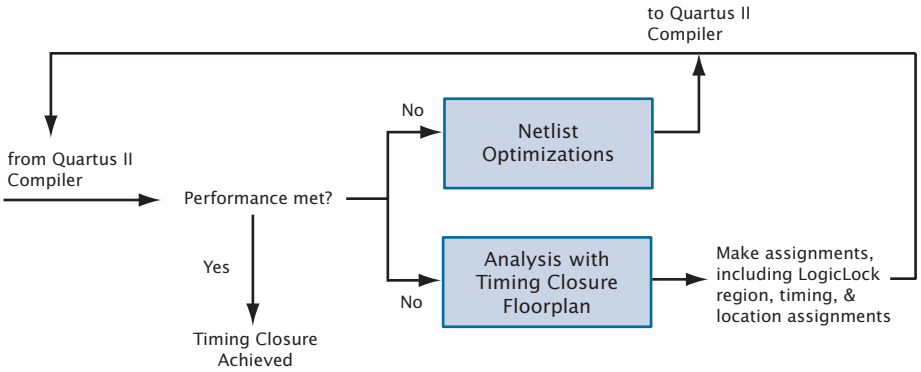
What's in Chapter 8:

Introduction

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

The timing closure flow allows you to perform an initial compilation, view design results, and perform further design optimization efficiently. You can use netlist optimizations on the design after synthesis and during place and route, use the Timing Closure floorplan to analyze the design and make assignments, and use LogicLock™ region assignments to further optimize the design. Figure 1 shows the timing closure flow.

Figure 1. Timing Closure Flow



Using the Timing Closure Floorplan

You can use the Timing Closure floorplan 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.

You can customize the way the Timing Closure floorplan displays information using options available from the View menu. You can show the device according to package pins and their function; by interior MegaLAB™ structures, LABs, and cells; by regions of the chip; by the name and location of selected signals; and by using the **Field View** command (View menu).

The **Field View** command displays the major classifications of device resources in a high-level outline view in the Floorplan Editor. Assignments are represented in Field view by colored areas that indicate the amount of user assigned, Fitter placed, and unassigned logic per structure in the device. You can use the information in the Field view to make assignments to achieve timing closure on a design.

Viewing Assignments & Routing

The Timing Closure floorplan can simultaneously show user assignments and Fitter location assignments. User assignments are all location and LogicLock region assignments you have made in the design. Fitter assignments are the locations where the Quartus II software placed all nodes after the last compilation. You can show user assignments and Fitter assignments with the **Assignments** command (View menu).

The Timing Closure Floorplan allows you to show the device resources and the corresponding routing information for all design logic. Using the **Routing** command (View menu), you can select device resources and view the following types of routing information:

- **Paths between nodes**: display the path between selected logic cells, I/O cells, embedded cells, and pins that feed one another.
- **Node fan-in and fan-out**: display node fan-in and fan-out routing information for selected embedded cells, logic cells, I/O cells and pins.
- **Routing delays**: display routing delays between selected, or to and from, specific logic cells, I/O cells, embedded cells, or pins; or routing delays along one or more critical paths.
- **Connection counts**: show or hide the number of connections to a selected object, from a selected object, or between selected objects.
- **Routing statistics**: display routing statistics for one or more selected pins, logic cells, embedded cells, I/O cells, LABs, or MegaLAB structures.

- **Physical timing estimates**: displays the approximate delay to any other node or entity on the device. Once you select a node or entity, the delay is represented by the shade of potential destination resources (the darker the resource, the longer the delay) and the delay to a destination node is shown by placing the mouse over possible destination nodes.

- **Routing congestion**: displays a graphical representation of the routing congestion in a design. The darker the shading, the greater the routing resource utilization. You can select a routing resource and then specify the congestion threshold (displayed as red areas in the device) for the resource.

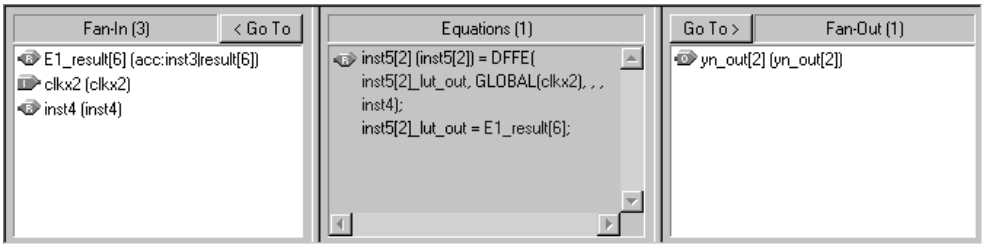
- **Critical paths**: displays the critical paths in a design, including path edges and routing delays. The default critical path view shows the register-to-register paths. You can also view all the combinatorial nodes for the worst-case path between the source and destination nodes. You can specify whether you want to view critical paths by delay or slack criteria and can specify a clock domain, source and destination node names, and the number of critical paths to display.

You can also view the routing information for LogicLock regions in the design, including connectivity and intra-region delay. LogicLock region connectivity displays the connectivity between entities assigned to LogicLock regions in the design and intra-region delay displays the maximum time delay between source and destination paths in a LogicLock region, including its child regions.

The Equations window displays routing and equation information for pin, I/O cell, logic cell, and embedded cell assignments. When you turn on **Equations** (View menu), the Equations window is displayed at the bottom of the Floorplan Editor window. See Figure 2 on page 117.

By selecting one or more logic cell, embedded cell, and/or pin assignments in the floorplan, you can display their equations, fan-in, and fan-out in the **Equations** list and expand or collapse the terms. The **Fan-In** list displays all nodes that feed the selected logic cell, embedded cell, and/or pin assignments. The **Fan-Out** list displays all nodes that are fed by the selected logic cell, embedded cell, and/or pin assignments.

Figure 2. Equations Window



Making Assignments

To facilitate achieving timing closure, the Timing Closure floorplan allows you to make location and timing assignments directly from the floorplan. You can create and assign nodes or entities to custom regions and to LogicLock regions in the Timing Closure floorplan and you can also edit existing assignments to pins, logic cells, rows, columns, regions, MegaLAB structures, and LABs.

You can edit assignments in the Timing Closure floorplan in the following ways:

- Cut, copy, and paste node and pin assignments.
- Launch the Assignment Editor to make assignments.
- Use the Node Finder to help make assignments.
- Create and assign logic to LogicLock regions.
- Drag and drop nodes and entities from the **Hierarchy** tab of the Project Navigator, LogicLock regions, and the Timing Closure floorplan to other areas of the floorplan.

Before making assignments, you can preserve resource assignments from the current compilation by back-annotating assignments to pins, logic cells, rows, columns, regions, LABs, MegaLAB structures, and LogicLock regions by using the **Back-Annotate Assignments** command (Assignments menu). For more information on using the **Back-Annotate Assignments** command, see "Preserving Assignments through Back-Annotation" on page 86 in "Chapter 5: Place & Route."

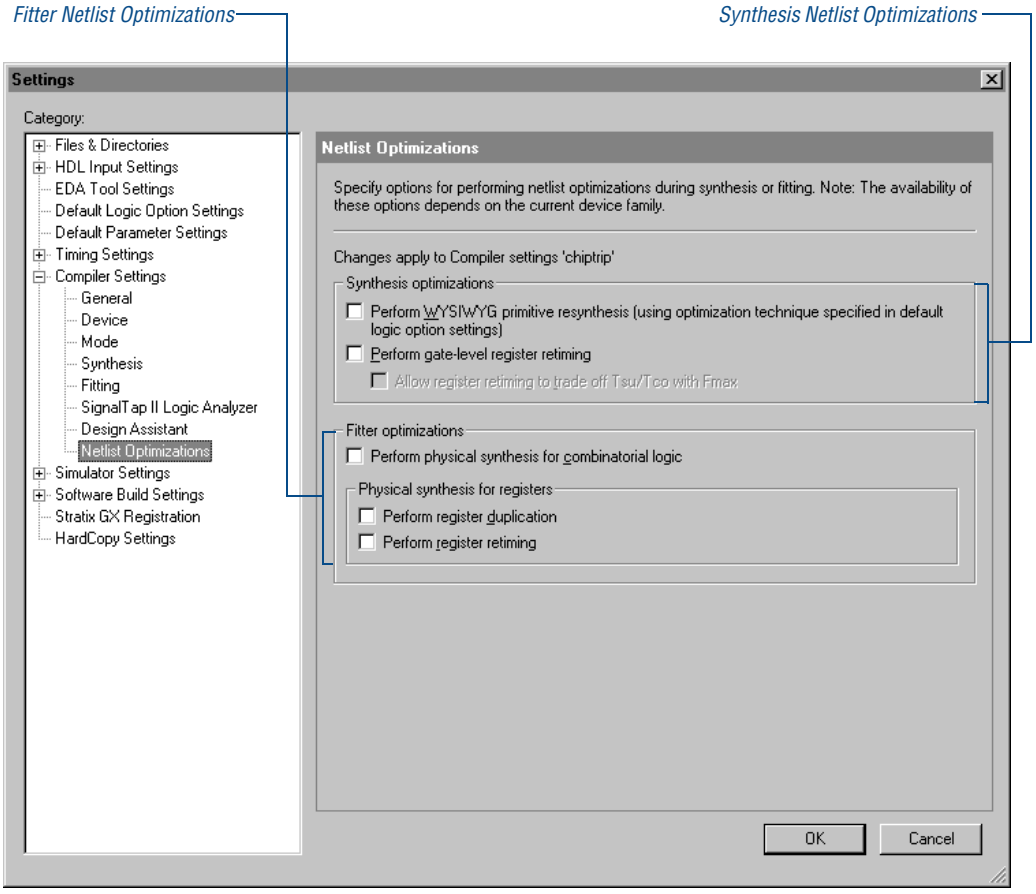
For Information About	Refer To
Viewing and making assignments and viewing routing in the Timing Closure floorplan	*Application Note 198 (Timing Closure with the Quartus II Software)* on the Altera web site "Overview: Viewing Routing Information" in Quartus II Help "Overview: Working with Assignments in the Floorplan Editor" in Quartus II Help LogicLock module in the Quartus II tutorial

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 Analysis & Synthesis and Fitter netlist optimizations in the **Netlist Optimizations** page of the **Settings** dialog box (Assignments menu). See Figure 3 on page 119.

Figure 3. Netlist Optimizations



Netlist optimizations for synthesis include the following options:

- **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 back to Altera® `LCELL` primitives. This option allows the Quartus II software to use different techniques specific to a device architecture during the re-mapping process.

- **Perform gate-level register retiming**: Allows registers to be moved across combinatorial logic to balance timing, but does not change the functionality of the current design. This option moves registers across

combinatorial 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 combinatorial logic block to the block's output, potentially combining the registers. It can also create multiple registers at the input of a combinatorial logic block from a register at the output of a combinatorial logic block.

- **Allow register retiming to trade off Tsu/Tco with Fmax**: Directs the Quartus II software to move logic across registers that are associated with I/O pins during register retiming to trade off t_{CO} and t_{SU} with f_{MAX}. When you turn on this option, register retiming can affect registers that feed and are fed by I/O pins. If you do not turn on this option, register retiming does not touch any registers that are connected to I/O pins.

Netlist optimizations for fitting and physical synthesis include the following options:

- **Perform physical synthesis for combinatorial logic**: Directs the Quartus II software to try to increase performance by performing physical synthesis optimizations on combinatorial logic during fitting.
- **Perform register duplication**: Directs the Quartus II software to increase performance by using register duplication to perform physical synthesis optimizations on registers during fitting.
- **Perform register retiming**: Directs the Quartus II software to increase performance by using register retiming to perform physical synthesis optimizations on registers during fitting.

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 VQM 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	*Application Note 198 (Timing Closure with the Quartus II Software)* on the Altera web site

Using LogicLock Regions to Achieve Timing Closure

You can use LogicLock regions to achieve timing closure by analyzing the design in the Timing Closure floorplan, and then constraining critical logic in LogicLock regions. LogicLock regions are generally hierarchical, giving 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 in a design requires a detailed understanding of the design's critical paths. Once you have implemented LogicLock regions and attained the desired performance, back-annotate the contents of the region to lock the logic placement.

Soft LogicLock Regions

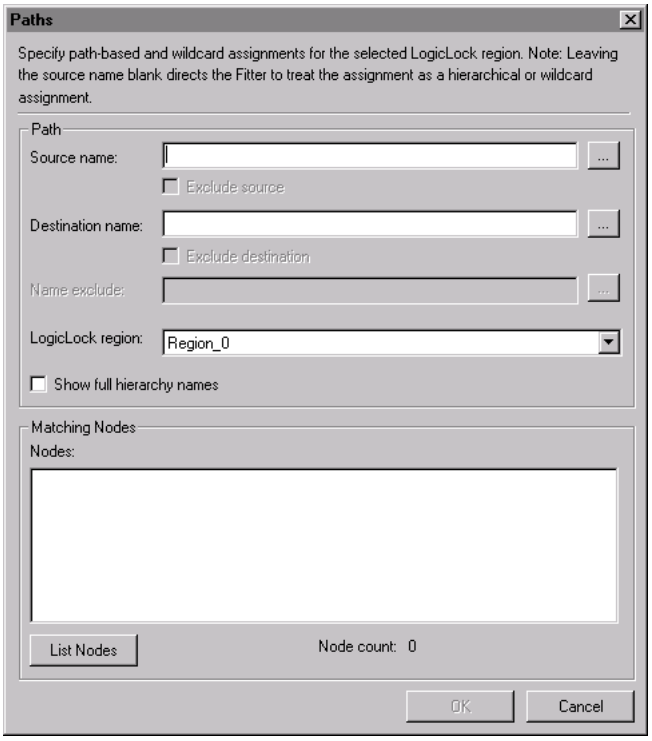
LogicLock regions have predefined boundaries and nodes assigned to a particular region always reside within the boundary or LogicLock region size. Soft LogicLock regions can enhance design performance by removing the fixed rectangular boundaries of LogicLock regions. With the soft region property enabled,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

Path-Based Assignments

The Quartus II software enables you to assign specific source and destination paths to LogicLock regions, allowing for easy grouping of critical design nodes into a LogicLock region. You can create path-based assignments with the **Path** dialog box (available from the **LogicLock Region Properties** dialog box), by dragging and dropping paths from the Timing Analyzer section of the Report window, and by dragging and dropping from critical paths in the Timing Closure floorplan (available by right-clicking on a critical path in the Timing Closure floorplan and selecting **Properties**).

The **Paths** dialog box allows you to specify a path by identifying a source and destination node, use wildcards when identifying nodes, and click **List Nodes** to determine how many nodes will be assigned to the LogicLock region. See Figure 4.

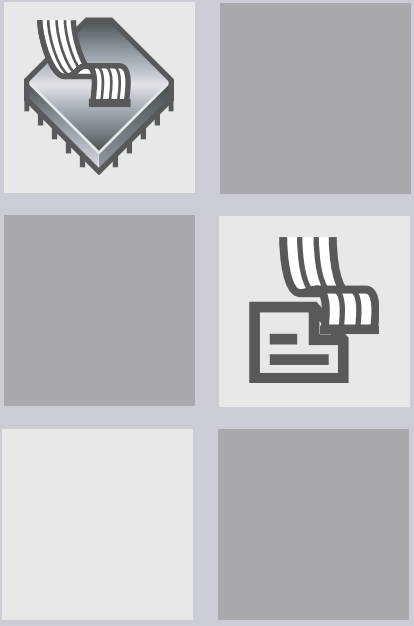
Figure 4. Paths Dialog Box



For Information About	Refer To
Achieving timing closure using the LogicLock methodology	*Application Note 161 (Using the LogicLock Methodology in the Quartus II Design Software)* on the Altera web site *Application Note 198 (Timing Closure with the Quartus II Software)* on the Altera web site
Creating LogicLock regions	LogicLock module of the Quartus II Tutorial

Chapter Nine

Programming & Configuration



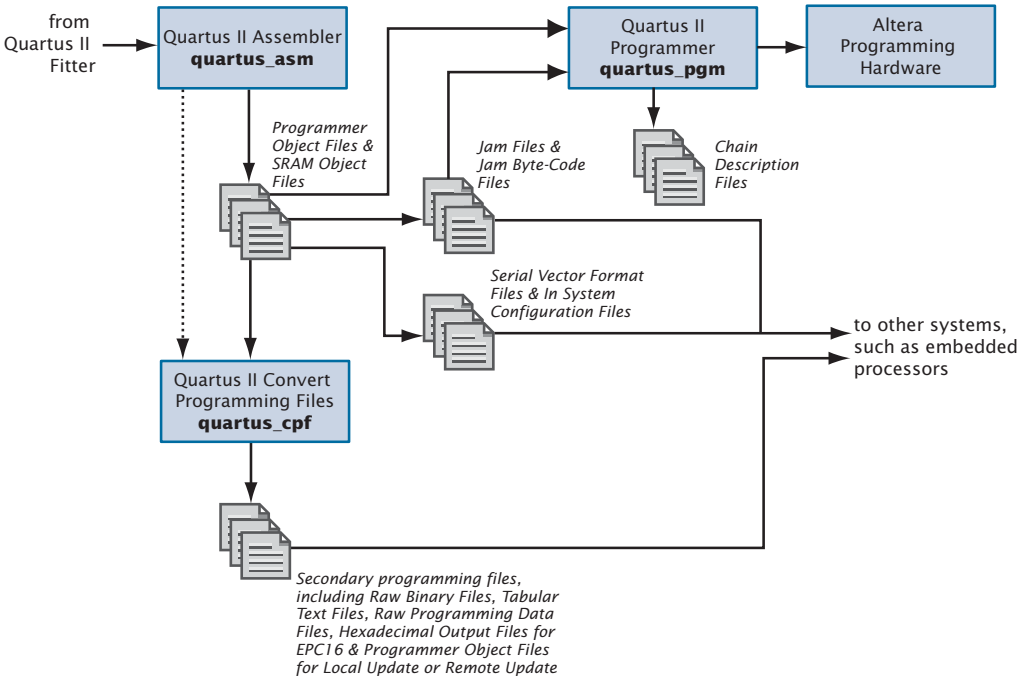
9

What's in Chapter 9:

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



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

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

Using the quartus_asm executable

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

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

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

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

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

- The **Device & Pin Options** dialog box, which is available from the **Device** page of the **Settings** dialog box (Assignments menu), allows you to specify optional programming file formats, such as Hexadecimal (Intel-Format) Output Files (**.hexout**), Tabular Text Files (**.ttf**), Raw Binary Files (**.rbf**), Jam™ Files (**.jam**), Jam Byte-Code Files (**.jbc**), Serial Vector Format Files (**.svf**), and In System Configuration Files (**.isc**).
- The **Create/Update > Create JAM, SVF, or ISC File** command (File menu) generates Jam Files, Jam Byte-Code Files, Serial Vector Format Files, or In System Configuration Files.
- The **Convert Programming Files** command (File menu) combines and converts SOFs and POFs for one or more designs into other secondary programming file formats, such as Raw Programming Data Files (**.rpd**), HEXOUT Files for EPC16 or SRAM, POFs, POFs for Local Update or Remote Update, Raw Binary Files, and Tabular Text Files.

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

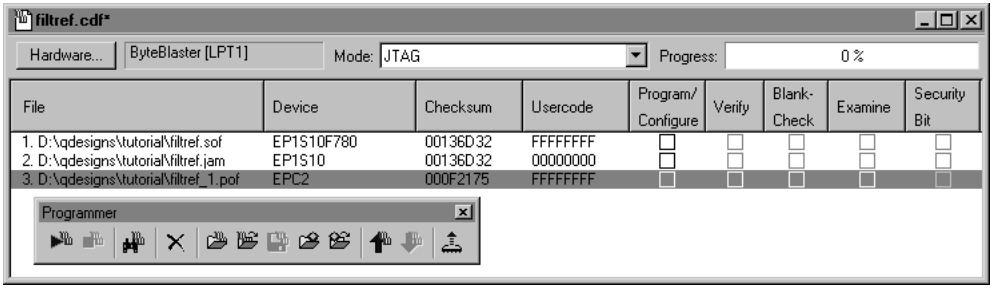
The Programmer uses the POFs and SOFs generated by the Assembler to program or configure all Altera devices supported by the Quartus II software. You use the Programmer with Altera programming hardware, such as the MasterBlaster™, ByteBlasterMV™, ByteBlaster™ II, or USB-Blaster download cable; or the 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. For some programming modes that allow programming or configuring multiple devices, the CDF also specifies top-to-bottom order of the SOFs, POFs, 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 mode
- JTAG mode
- Active Serial Programming mode
- In-Socket Programming mode

The Passive Serial and JTAG programming modes allow you to program single or multiple devices using a CDF 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 CDF 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, "Overview: Working with Chain Description Files," and "Overview: Converting Programming Files" in Quartus II Help
Using the Programmer	The Programming module of the Quartus II Tutorial
Altera programming hardware	*Altera Programming Hardware Data Sheet*, *ByteBlaster II Parallel Port Download Cable Data Sheet*, *ByteBlasterMV Parallel Port Download Cable Data Sheet*, *MasterBlaster Serial/USB Communications Cable Data Sheet*, or *USB-Blaster Download Cable Data Sheet* on the Altera web site
Programming hardware installation	*Quartus II Installation & Licensing for PCs* and *Quartus II Installation & Licensing for UNIX and Linux Workstations* manuals
Device-specific programming information	*Configuring Stratix & Stratix GX Devices* and *Using Altera Enhanced Configuration Devices* chapters of the *Stratix Device Handbook* on the Altera web site *Application Note 250 (Configuring Cyclone FPGAs)* on the Altera web site

Programming One or More Devices by Using the Programmer

The Quartus II Programmer allows you to edit a CDF, which stores device name, device order, and optional programming file name information for a design. You can use a CDF to program or configure a device with one or more SOFs, POFs, or with a single Jam File or Jam Byte-Code File.

The following steps describe the basic flow for programming one or more devices by using the Programmer:

1. Connect Altera programming hardware to your system and install any necessary drivers.

2. Perform a full compilation of the design, or at least run the Analysis & Synthesis, Fitter, and Assembler modules of the Compiler. The Assembler automatically creates SOFs and POFs for the design.

3. Open the Programmer to create a new CDF. Each open Programmer window represents one CDF; you can have multiple CDFs open, but you can program using only one CDF at a time.

4. Select a programming hardware setup. The programming hardware setup you select affects the types of programming modes available in the Programmer.

5. Select an appropriate programming mode, such as Passive Serial mode, JTAG mode, Active Serial Programming mode, or In-Socket Programming mode.

6. Depending on the programming mode, you can add, delete, or change the order of programming files and devices in the CDF. You can direct the Programmer to detect Altera-supported devices in a JTAG Chain automatically and add them to the device list of the CDF. You can also add user-defined devices.

7. For non-SRAM, non-volatile devices, such as configuration devices, MAX 3000, and MAX 7000 devices, you can specify additional programming options to query the device, such as **Verify**, **Blank-Check**, **Examine**, and **Security Bit**.

8. Start the Programmer.

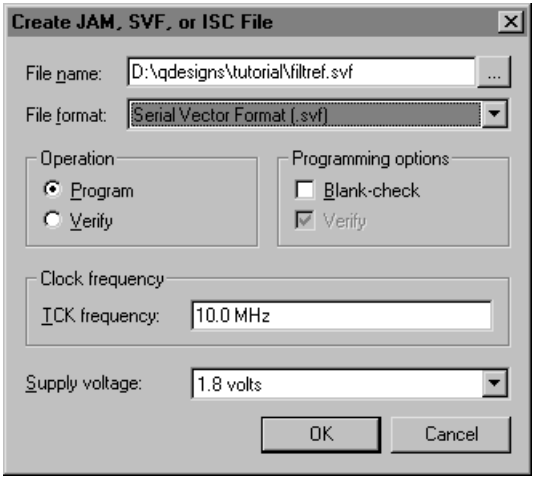
Creating Secondary Programming Files

You can also create secondary programming files in other formats, such as Jam Files, Jam Byte-Code Files, Serial Vector Format Files, In System Configuration Files, Raw Binary Files, or Tabular Text Files, for use by other systems, such as embedded processors. Additionally, you can convert SOFs or POFs into other programming file formats, such as a POF for Remote Update, a POF for Local Update, a HEXOUT File for EPC16, a HEXOUT File for SRAM, or a Raw Programming Data File. You can create these secondary programming files by using the **Create/Update > Create JAM, SVF, or ISC File** command (File menu) and the **Convert Programming Files** command (File menu). You can also use the **Programming Files** tab of the **Device & Pin Options** dialog box, which is available from the **Device** page in the **Settings** dialog box (Assignments menu), to specify optional programming file formats for the Assembler to generate during compilation.

Creating Other Programming File Formats

You can use the **Create/Update > Create JAM, SVF, or ISC File** command (File menu) to create Jam Files, Jam Byte-Code Files, Serial Vector Format Files, or In System Configuration Files. These files can then be used in conjunction with Altera programming hardware or an intelligent host to configure any Altera device supported by the Quartus II software. You can also add Jam Files and Jam Byte-Code Files to CDFs. See Figure 3 on page 130.

Figure 3. Create JAM, SVF, or ISC File Dialog Box



The following steps describe the basic flow for creating Jam Files, Jam Byte-Code Files, Serial Vector Format Files, or In System Configuration Files:

1. Perform a full compilation of the design, or at least run the Analysis & Synthesis, Fitter, and Assembler modules of the Compiler. The Assembler automatically creates SOFs and POFs for the design.

2. Open the Programmer window to create a new CDF.

3. Specify JTAG mode.

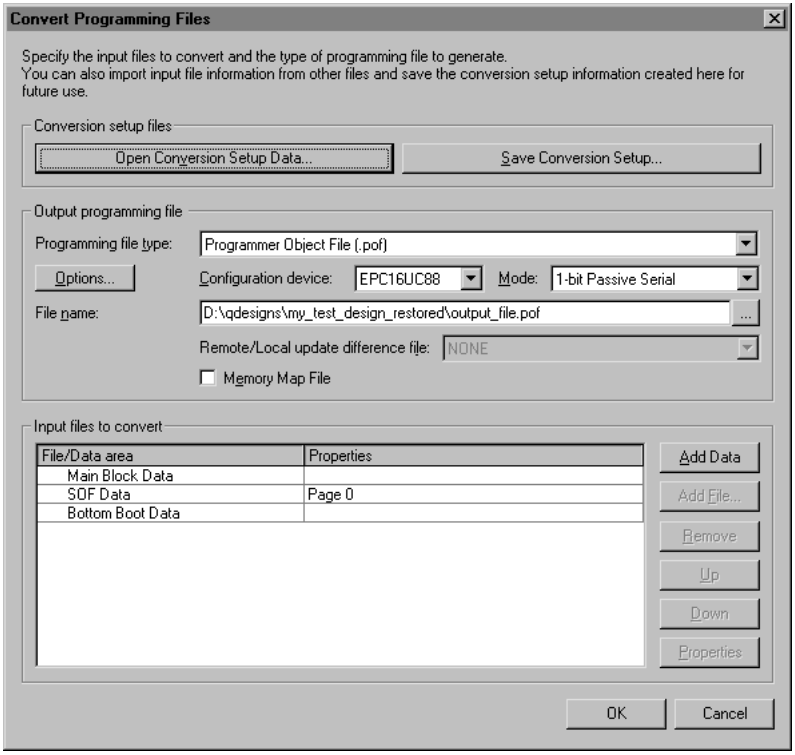
4. Add, delete, or change the order of programming files and devices in the CDF. You can direct the Programmer to detect Altera-supported devices in a JTAG Chain automatically and add them to the device list of the CDF. You can also add user-defined devices.

5. Choose **Create/Update > Create Jam, SVF, or ISC File** (File menu) and specify the name and file format of the file you want to create.

Converting Programming Files

You can use the **Convert Programming Files** dialog box (File menu) to combine and convert SOFs or POFs for one or more designs into other programming file formats for use with different configuration schemes. For example, you can add a remote update-enabled SOF to a POF for Remote Update, which is used to program a configuration device in remote update configuration mode, or you can convert a Programmer Object File into a HEXOUT File for EPC16 for use by an external host. Or you can convert a POF into a Raw Programming Data File for use with some configuration devices. See Figure 4.

Figure 4. Convert Programming Files Dialog Box



You can use the **Convert Programming Files** dialog box to set up output programming files by arranging the chain of SOFs stored in a HEXOUT File for SRAM, POFs, Raw Binary Files, or Tabular Text Files, or by specifying a POF to be stored in a HEXOUT File for EPC16. The settings you specify in

the **Convert Programming Files** dialog box are saved to a Conversion Setup File (**.cof**) that contains information such as device and file names, device order, device properties, and file options.

For a Programmer Object File for an EPC4, EPC8, or EPC16 configuration device, you can also specify the following information:

- Establish different configuration bitstreams, which are stored in pages in the configuration memory space.
- Create parallel chains of SOFs within each page.
- Arrange the order of SOFs and Hexadecimal (Intel-Format) Files (**.hex**) stored in flash memory.
- Specify the properties of **SOF Data** items and HEX Files.
- Add or remove **SOF Data** items from the configuration memory space.
- If you wish, create Memory Map Files (**.map**).

For POFs for Local Update and POFs for Remote Update, you can specify the following information:

- Add or remove remote update enabled POFs and remote update enabled SOFs from the configuration memory space.
- Specify the properties of **SOF Data** items.
- Add or remove **SOF Data** items.
- If you wish, create Memory Map Files, and generate remote update difference files and local update difference files.

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

Using the quartus_cpf executable

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

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

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

The following steps describe the basic flow for converting programming files:

1. Run the Assembler module of the Compiler. The Assembler automatically creates SOFs and POFs for the design.
2. Use the **Convert Programming Files** dialog box (File menu) to create a Conversion Setup File and specify the format and name of the programming file you want to create.
3. Specify a configuration mode that is compatible with the configuration memory space of the programming file.
4. Specify appropriate programming options for the programming file type and target device.
5. (Optional) Generate a remote update difference file or a local update difference file for a Programmer Object File for Remote Update or a Programmer Object File for Local Update, by selecting the type of difference file.
6. Add or remove **SOF Data** items and assign them to pages.
7. (Optional) Add, remove, or change the order of SOFs and POFs to be converted for one or more **SOF Data** item(s) or **POF Data** item(s).
8. (Optional) Add a HEX File to a **Bottom Boot Data** or **Main Block Data** item for a POF for an EPC4, EPC8, or EPC16 configuration device, and specify additional properties of SOF Data, POF Data, and HEX Files.
9. Convert the file. If you want, you can also specify a Memory Map File to be created.

For Information About	Refer To
In-system programmability and In-circuit reconfigurability	*Application Note 100 (In-System Programmability Guidelines)* on the Altera web site *Application Note 95 (In-System Programmability in MAX Devices)* on the Altera web site *Application Note 88 (Using the Jam Language for ISP & ICR via an Embedded Processor)* on the Altera web site *Application Note 122 (Using Jam STAPL for ISP & ICR via an Embedded Processor)* on the Altera web site *Application Note 85 (In-System Programming Times for MAX Devices)* on the Altera web site *Application Note 298 (Reconfiguring Excalibur Devices under Processor Control)* on the Altera web site
In-system programming	The Programming module of the Quartus II Tutorial
Remote system configuration	*Using Remote System Configuration with Stratix & Stratix GX Devices* chapter of the *Stratix Device Handbook* on the Altera web site

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

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

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

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

For Information About	Refer To
Using a Local JTAG Server	"Configuring Local JTAG Server Settings," and "Adding a JTAG Server" in Quartus II Help

Chapter Ten

Debugging

10

What's in Chapter 10:

Introduction

The Quartus® II SignalTap® II Logic Analyzer and the SignalProbe™ feature analyze internal device nodes and I/O pins while operating in-system and at system speeds. The SignalTap II Logic Analyzer uses an embedded logic analyzer to route the signal data through the JTAG port to either the SignalTap II Logic Analyzer or an external logic analyzer or oscilloscope. The SignalProbe feature uses incremental routing on unused device routing resources to route selected signals to an external logic analyzer or oscilloscope. Figure 1 and Figure 2 show the SignalTap II and SignalProbe debugging flows.

Figure 1. SignalTap II Debugging Flow

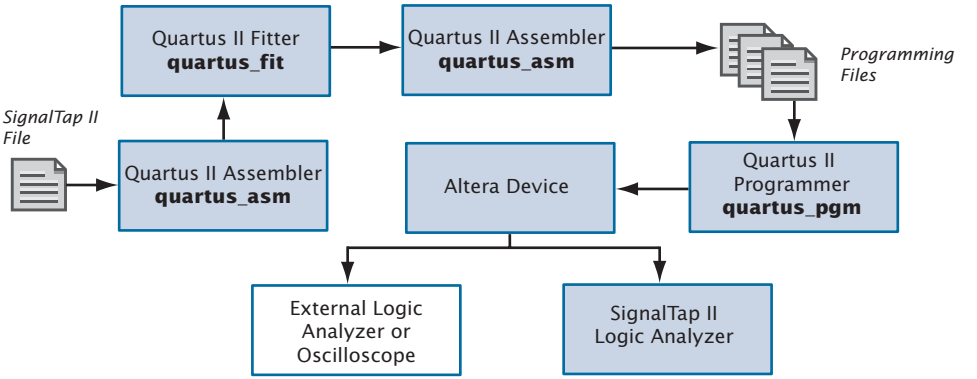
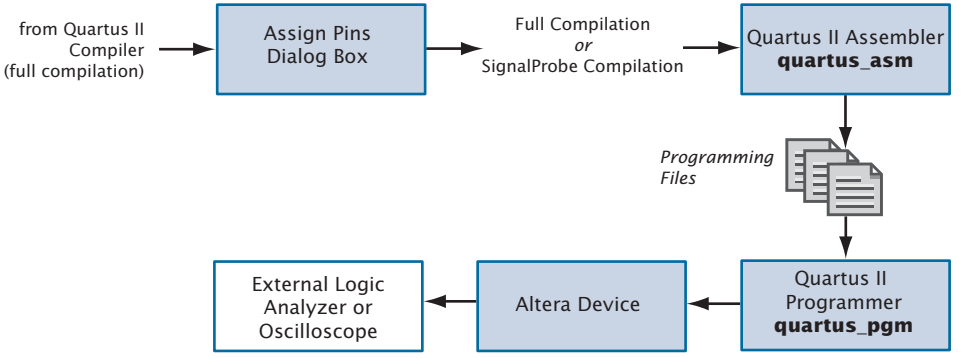


Figure 2. SignalProbe Debugging Flow



Using the SignalTap II Logic Analyzer

The SignalTap II Logic Analyzer is a second-generation 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 the 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, or USB-Blaster communications cable to download configuration data to the device. These cables are also used to upload captured signal data from the device's RAM resources to the Quartus II software. The Quartus II software then displays data acquired by the SignalTap II Logic Analyzer as waveforms.

Setting Up & Running the SignalTap II Logic Analyzer

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

Each logic analyzer instance is embedded in the logic on the device. The SignalTap II Logic Analyzer supports up to 1,024 channels and 128K samples on a single device.

After compilation, you can run the SignalTap II Logic Analyzer by using the **Run Analysis** command (Processing menu). See Figure 3 on page 140.

The following steps describe the basic flow to set up an SignalTap II File and acquire signal data:

1. Create a new SignalTap II File.

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

3. Assign a clock to each instance.

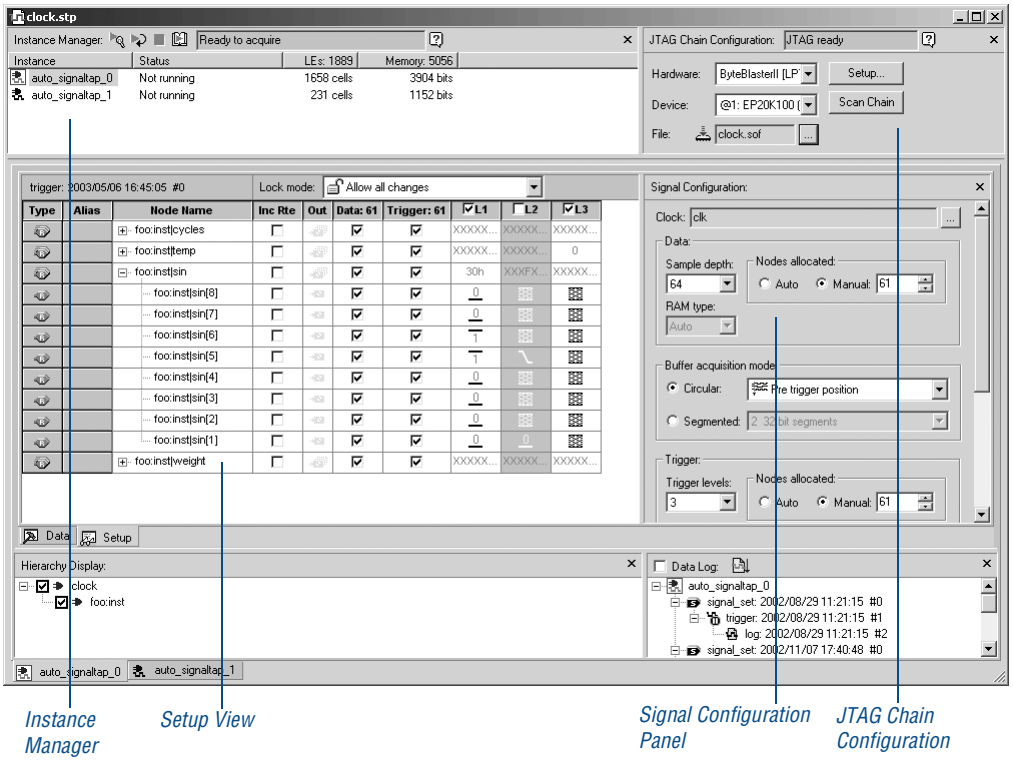
4. Set other options, such as sample depth and trigger level, and assign signals to the data/trigger input and debug port.

5. Compile the design.

6. Program the device.

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

Figure 3. The SignalTap II Logic Analyzer



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

- **Multiple Logic Analyzers**: The SignalTap II Logic Analyzer supports multiple embedded instances of the logic analyzer in each device. You can use this feature to create a separate and unique logic analyzer for each clock domain in the device, and apply different settings to multiple embedded logic analyzers.

- **Instance Manager**: The Instance Manager allows you create and perform SignalTap II logic analysis on multiple instances. You can use it to create, delete, and rename instances in the SignalTap II File. The Instance Manager displays all instances in the current SignalTap II File, the current status of each associated instance, and the number of logic elements and memory bits used in the associated instance. The Instance Manager helps you to check the amount of resource usage that each logic analyzer requires on the device. You can start multiple logic analyzers at the same time by selecting them and selecting **Run Analysis** (Processing menu).

- **Triggers**: A trigger is a pattern of logic events in terms of logic levels and/or edges. The SignalTap II Logic Analyzer supports multi-level triggering, multiple trigger positions, multiple segments, and external trigger events. You can set trigger options using the **Signal Configuration** panel in the SignalTap II Logic Analyzer window.

  You can configure the logic analyzer with up to ten trigger levels, helping you to view only the most significant data. You can specify four separate trigger positions: pre, center, post, and continuous. The trigger position allows you to specify the amount of data that should be acquired before the trigger and the amount that should be acquired after the trigger in the selected instance. Segmented mode allows you to capture data for periodic events without allocating a large sample depth by segmenting the memory into discreet time periods.

- **Incremental Routing**: The incremental routing feature helps to shorten the debugging process by allowing you to analyze post-fitting nodes without performing a full recompilation.

  Before using the SignalTap II incremental routing feature, you must perform a smart compilation by turning on **Automatically turn on smart compilation if conditions exist in which SignalTap II with incremental routing is used**, in the **SignalTap II Logic Analyzer** page of the **Settings** dialog box (Assignments menu). Also, you must reserve trigger or data nodes for SignalTap II incremental routing using the **Trigger Nodes allocated** and **Data Nodes allocated** boxes before

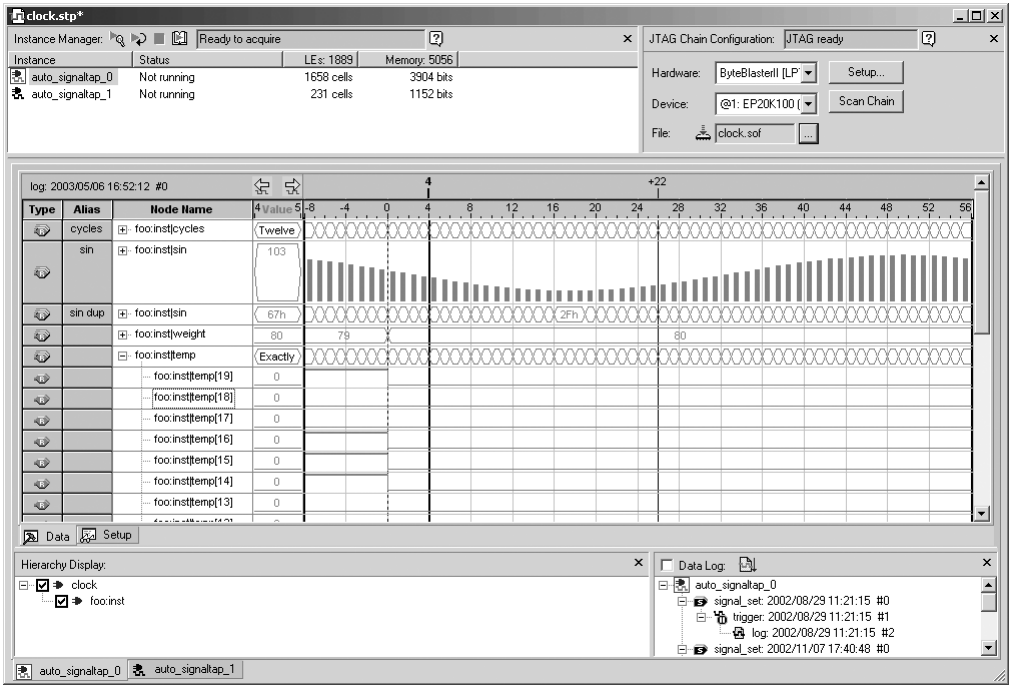
compiling the design. You can find nodes for SignalTap II incremental routing sources by selecting **SignalTap II: post-fitting** in the **Filter** list in the Node Finder.

Analyzing SignalTap II Data

When you are using 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, duplicate nodes; create, rename, and ungroup a bus; specify a data format for bus values; and print the waveform data. The data log that is used to create the waveform shows a history of data that is acquired with the SignalTap II Logic Analyzer. The data is organized in a hierarchical manner; logs of captured data using the same trigger are grouped together in **Trigger Sets**. Figure 4 shows the waveform view.

Figure 4. SignalTap II Waveform View

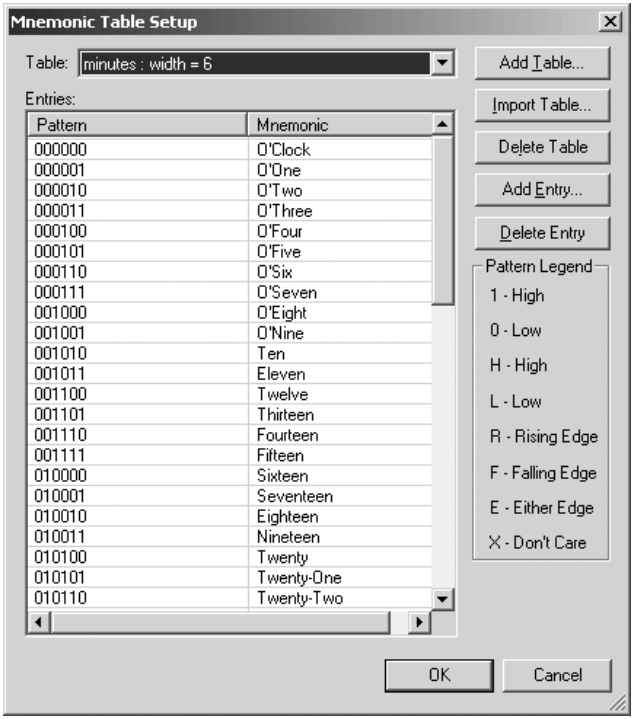


The **Waveform Export** utility allows you to export the acquired data to the following industry-standard formats that can be used by EDA tools:

- Comma Separated Values File (**.csv**)
- Table File (**.tbl**)
- Value Change Dump File (**.vcd**)
- Vector Waveform File (**.vwf**)

You can also configure the SignalTap II Logic Analyzer to create mnemonic tables for a group of signals. The mnemonic table feature allows a predefined name to be assigned to a set of bit patterns, making captured data more meaningful. See Figure 5.

Figure 5. Mnemonic Table Setup Dialog Box



For Information About	Refer To
Using the SignalTap II Logic Analyzer	*Application Note 280 (Design Verification Using the SignalTap II Embedded Logic Analyzer)* on the Altera web site "Overview: Using the SignalTap II Logic Analyzer" in Quartus II Help

Using SignalProbe

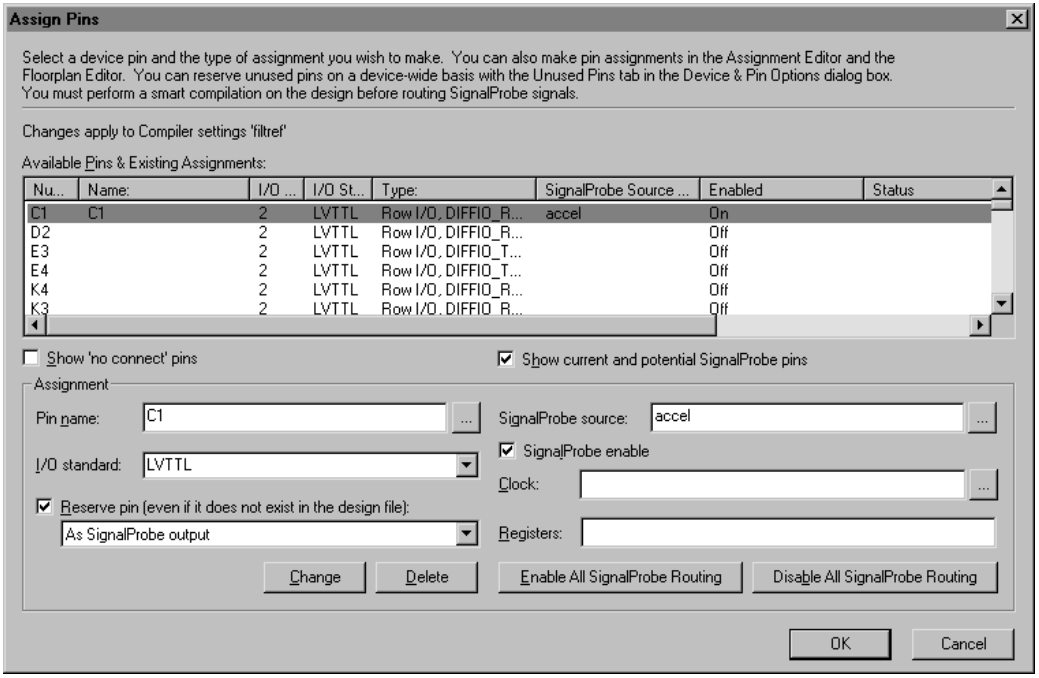
The SignalProbe feature allows you to route user-specified signals to output pins without affecting the existing fitting in a design, so that you can debug signals without needing to perform another a full compilation. Starting with a fully routed design, you can select and route signals for debugging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

The SignalProbe feature allows you to specify which signals in the design to debug, and then perform a SignalProbe compilation that connects those signals to unused or reserved output pins, and then sends 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 10% of the time required for a normal compilation.

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

1. Perform a full compilation of the design.
2. Select signals for debugging and the I/O pins to route the signals, and turn on the SignalProbe feature in the **Assign Pins** dialog box, which is available from the **Device** page of the **Settings** dialog box (Assignments menu). See Figure 6 on page 145.
3. Perform a SignalProbe compilation. Alternatively, you can turn on **Automatically route SignalProbe signals during compilation** in the **Mode** page of the **Settings** dialog box and then choose **Start Compilation** (Processing menu) to include SignalProbe connections in a full compilation.
4. Configure the device with the new programming data to examine the signals.

Figure 6. Assign Pins Dialog Box



When reserving SignalProbe pins, you can also use the register pipelining feature to allow synchronization of debugging signals with clock signals, eliminating routing delay from the source signal to the output pin.

You can keep or remove all or some of the SignalProbe routing after debugging. If you keep SignalProbe routing in a design, you can automatically route SignalProbe routing during a full compilation.

You can also 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	*Technical Brief 82 (SignalProbe Compilation Enables Fast System Debugging with the Quartus II Software)* on the Altera web site "SignalProbe Introduction" in Quartus II Help
Using TCL commands with the SignalProbe feature	*Application Note 195 (Scripting with Tcl in the Quartus II Software)* on the Altera web site

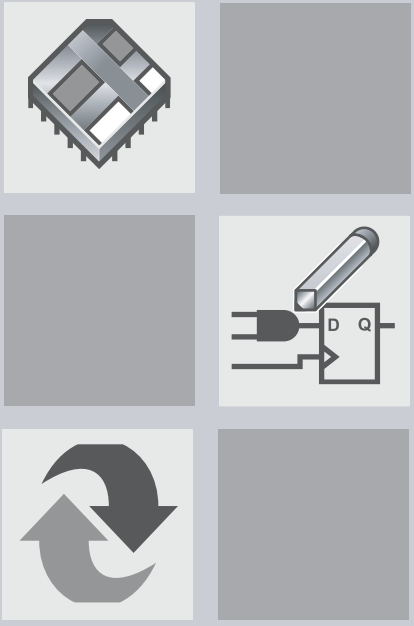
Using the Chip Editor

You can use the Chip Editor in conjunction with the SignalTap II 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 Editor 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.

For more information on using the Chip Editor, refer to the next chapter, "Chapter 11: Engineering Change Management."

Chapter Eleven

Engineering Change Management



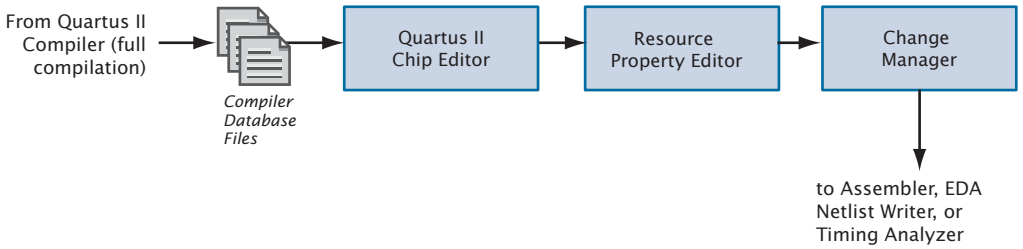
What's in Chapter 11:

11

Introduction

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

Figure 1. Engineering Change Management Design Flow



The following steps outline the design flow for engineering change management in the Quartus II software.

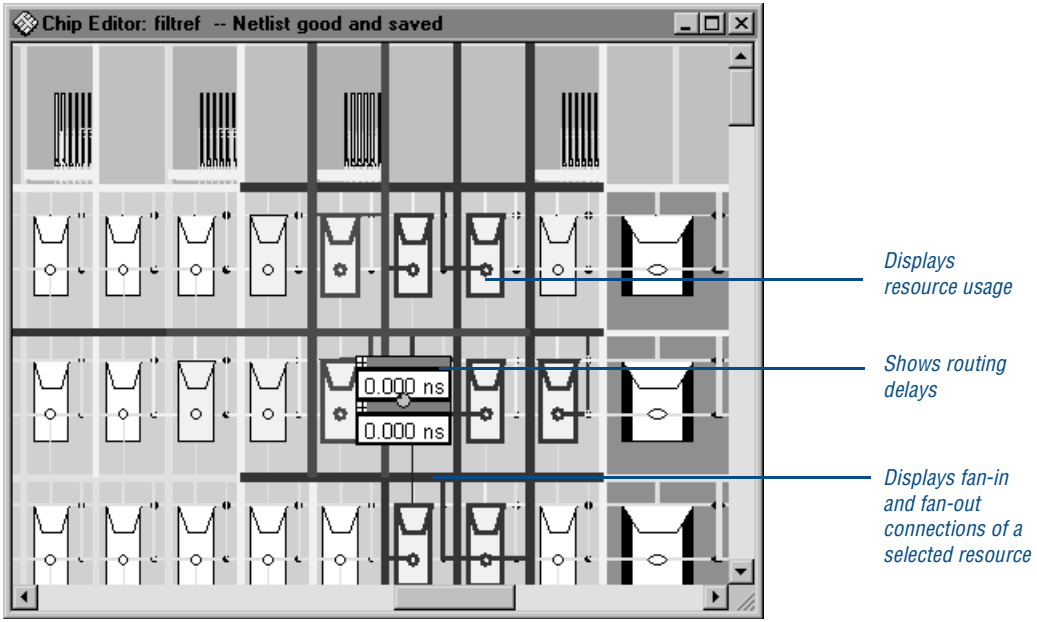
1. After a full compilation, use the Chip Editor to view design placement and routing details and identify which resources you want to change. If you want, you can use the Netlist Explorer to filter and highlight resources.
2. Use the Resource Property Editor to edit internal properties of resources.
3. Use the **Check Resource Properties** command (Edit menu) to check the legality of the change for the resource.
4. View the summary and status of your changes in the Change Manager and control which changes to resource properties are implemented and/or saved. You can also add comments to help you reference each change.
5. Use the **Check and Save All Netlist Changes** command (Edit menu) to check the legality of the change for all of the other resources in the netlist.

6. Run the Assembler to generate a new programming file or run the EDA Netlist Writer again to generate a new netlist. If you want to verify timing changes, you can run the Timing Analyzer.

Identifying Delays & Critical Paths with the Chip Editor

You can use the Chip Editor to view details of placement and routing. The Chip Editor reveals additional details about design placement and routing that are not visible in the Quartus II Floorplan Editor. It shows complete routing information, showing all possible and used routing paths between each device resource. See Figure 2.

Figure 2. Chip Editor

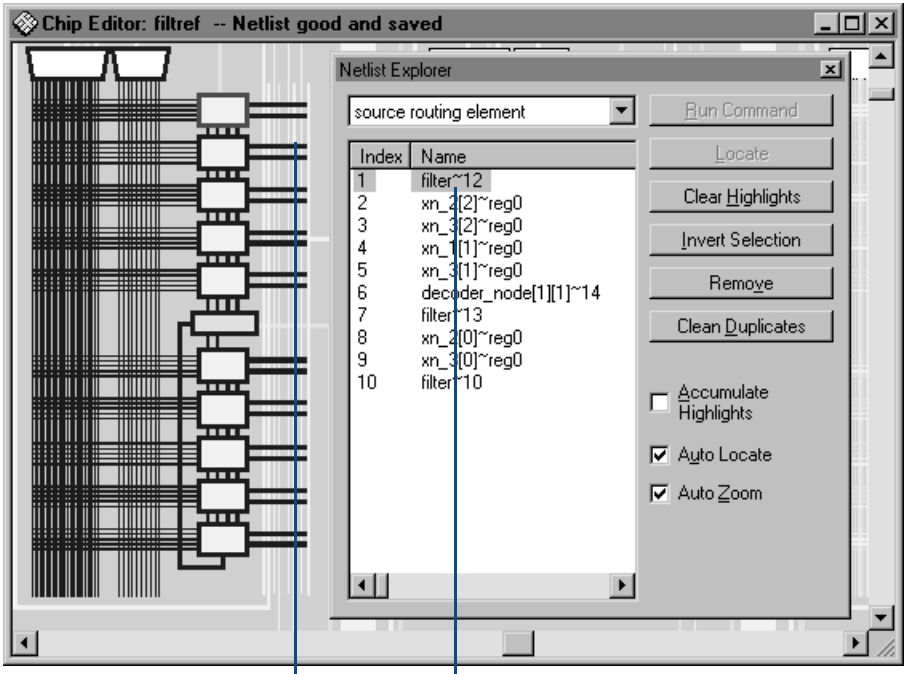


The Chip Editor displays all the resources of the device, such as interconnects and routing lines, logic array blocks (LABs), RAM blocks, DSP blocks, I/Os, rows, columns, and the interfaces between blocks and interconnects and other routing lines.

You can control the level of detail of the Chip Editor display by zooming in and out, selecting specific paths you want to display, and displaying a separate Bird's Eye View window, which shows magnification of the device view. You can also set options that control the display of different resources, as well as fan-in and fan-out, critical paths, and delay estimates on signals. You can then use this information to determine which properties and settings you may want to edit in the Resource Property Editor. You can select a resource in the Chip Editor and choose **Locate in Resource Property Editor** (right button pop-up menu) to open the Resource Property Editor and edit that resource. Refer to "Modifying Resource Properties with the Resource Property Editor" on page 151 for more information.

The Chip Editor also includes a Netlist Explorer window that allows you to highlight and select netlist elements in the Chip Editor. See Figure 3.

Figure 3. Netlist Explorer

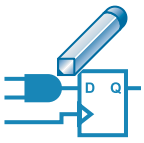


The Netlist Explorer allows you to filter and highlight resources in the Chip Editor

When you select elements in the Chip Editor, they will be displayed in the list in the Netlist Explorer. You can then apply different filters and commands, such as commands to find fan-outs or routing elements, or options to filter the list based on certain criteria, such as slack, name, and so on. The list will be updated and filtered based on the options you select. You can keep "exploring" the netlist by repeating these steps and applying different commands.

For Information About	Refer To
Engineering change management and using the Chip Editor	*Application Note 310 (Using the Quartus II Chip Editor)* on the Altera web site
Using the Chip Editor	"Overview: Using the Chip Editor" and "Making Post-Compilation Changes Introduction" in Quartus II Help

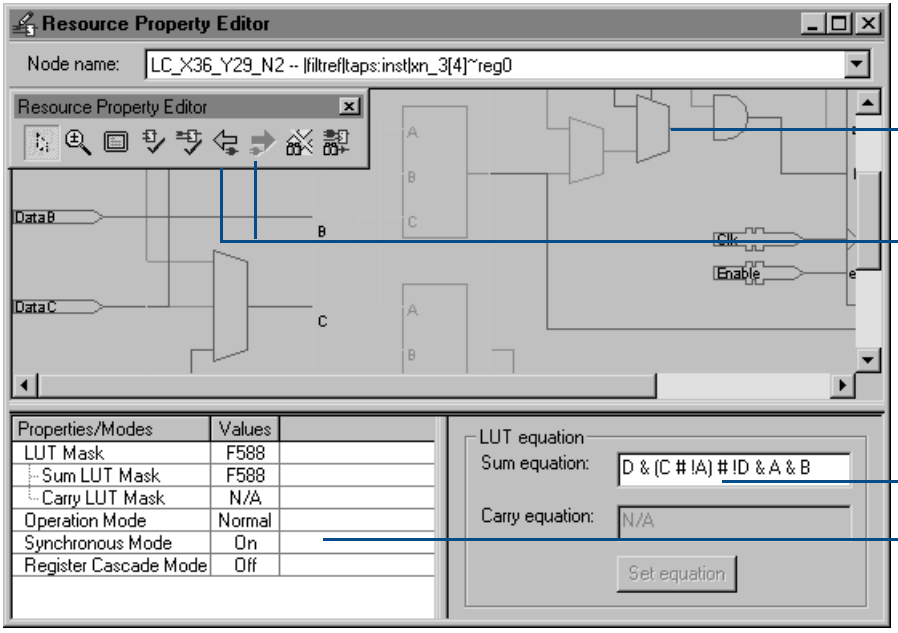
Modifying Resource Properties with the Resource Property Editor



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. You can use the toolbar buttons that allow you to navigate forward and backward among the resources. You can also select and change multiple resources at one time. You can follow the fan-in and fan-out of a resource and can view the resource in the Resource Property Editor.

The Resource Property Editor contains a viewer that shows a schematic diagram of the resource you are modifying, and a property table that displays the properties and parameters that are available for that resource. See Figure 4 on page 152.

Figure 4. Resource Property Editor



Viewer shows schematic diagram of resource

The forward and back buttons allow you to select the next resource

Property table displays the properties and values for resource and allows you to make changes

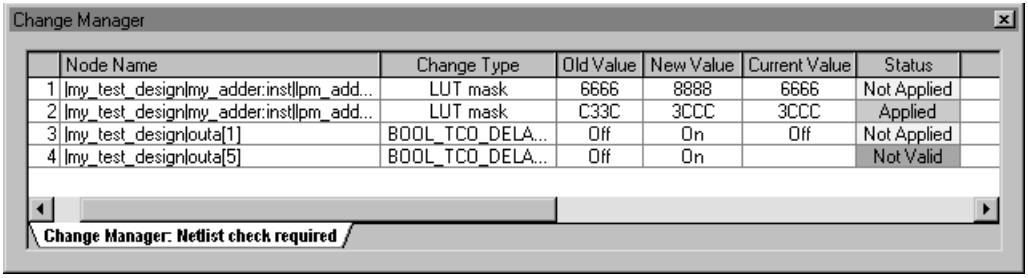
You can make changes to the resource in either the schematic or in the property table. If you make a change in the property table, that change is reflected automatically in the schematic diagram. Once you have made a change, you can use the **Check Resource Properties** command (Edit menu) to perform simple design-rule checking on the resource. 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	*Application Note 310 (Using the Quartus II Chip Editor)* on the Altera web site
Using the Resource Property Editor	"Overview: Using the Resource Property Editor" and "Making Post-Compilation Changes Introduction" in Quartus II Help

Viewing & Managing Changes with the Change Manager

The Change Manager window lists all the ECO changes that you have made. It 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. See Figure 5.

Figure 5. Change Manager



The log view of the Change Manager displays the following information for each ECO change:

- Change number
- Node name
- Change type
- Old value
- New value
- Current Value
- Comments that you have added about the ECO change.
- Status, which can be one of the following indicators:

  – **Pending:** You have made a change in the Resource Property Editor, but it has not yet been checked by using the **Check Resource Properties** command (Edit menu) to make sure it is legal in the context of the single resource.

  – **Applied:** You have made a change in the Resource Property Editor and have checked it by using the **Check Resource Properties** command, and the check has been successful. The change has been entered into the netlist, and the value has been saved in the Compiler Database File (**.cdb**).

– **Not Valid:** You have made a change in the Resource Property Editor and have checked the **Check Resource Properties** command, but the Change Manager has determined that the change is no longer valid and either cannot be applied safely to the design, or is impossible to apply to the design (for example, if the resource no longer exists). The value in the Compiler Database File does not match the values in both the **Old Value** and **New Value** columns.

– **Not Applied:** You have made a change in the Resource Property Editor and have checked it by using the **Check Resource Properties** command, but the design was either recompiled or was modified outside the Quartus II software. The value in the Compiler Database File matches the value in the **Old Value** column, but does not match the value in the **New Value** column. If the change has a **Not Applied** status, you can still apply the change manually by using the **Apply New Value** command (right button pop-up menu).

After you have applied the changes you want, you should choose **Check and Save All Netlist Changes** (Edit menu) to check the legality of the change for all of the other resources in the netlist.

If the ECO change requires that the Assembler module of the Compiler should be rerun, it will also display the message, `POF not up-to-date`, which means you must use the Assembler to generate an updated Programmer Object File (**.pof**).

You can perform the following actions on the ECO changes in the list by using commands from the right button pop-up menu:

- **Apply New Value**
- **Apply New Value To All**
- **Restore Old Value**
- **Delete**

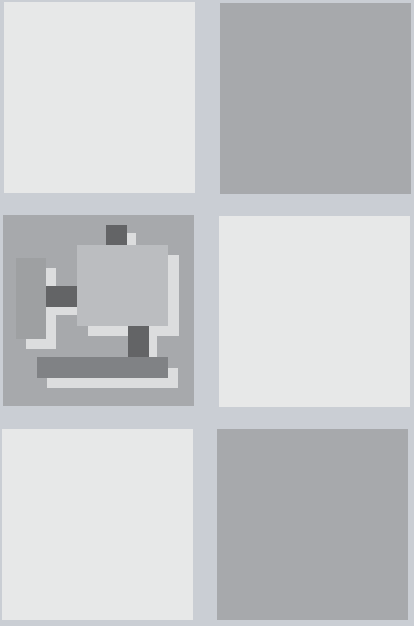
For Information About	Refer To
Engineering change management and using the Change Manager	*Application Note 310 (Using the Quartus II Chip Editor)* on the Altera web site
Using the Change Manager	"Overview: Using the Change Manager" and "Making Post-Compilation Changes Introduction" in Quartus II Help

Verifying the Effect of ECO Changes

After you have made an ECO change, you should not have to run a full compilation again, although you may need to run the Assembler module of the Compiler in order to create a new POF, or you may want to run the EDA Netlist Writer again to generate a new netlist. You may also want to run the Timing Analyzer again to verify that the change results in the appropriate timing improvement. You can run each of these modules separately by using the Compiler Flow window, or by using the **quartus_asm** or **quartus_eda**, and **quartus_tan** executables at the command line or in a script.

Chapter Twelve

System-Level Design



12

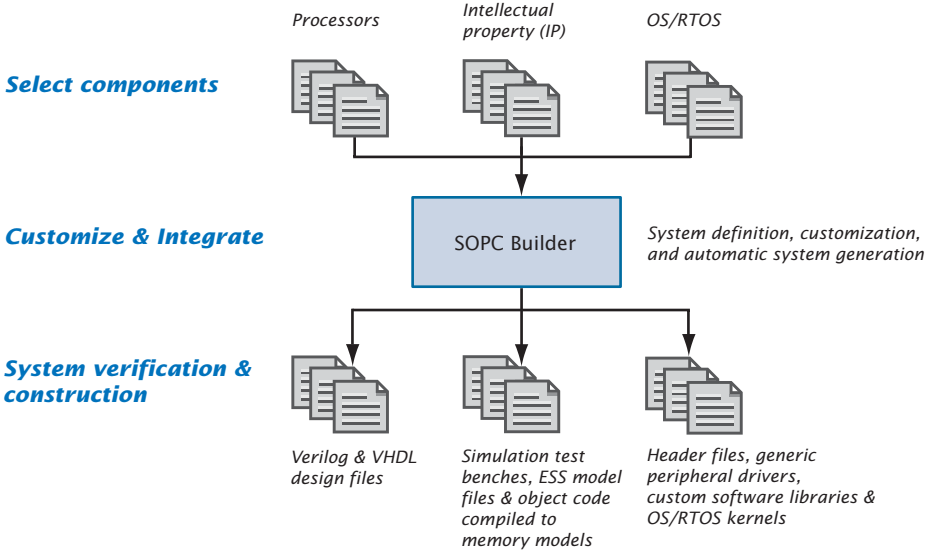
What's in Chapter 12:

Introduction

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

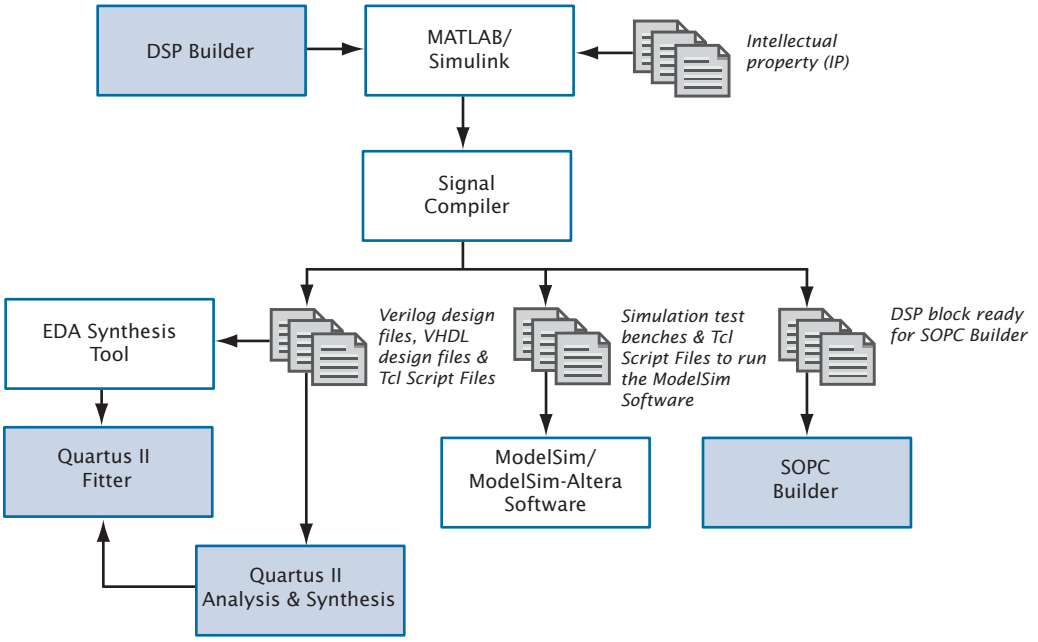
The SOPC Builder is an automated system development tool that dramatically simplifies the task of creating high-performance SOPC designs. The tool automates the system definition and integration phases of SOPC development completely within the Quartus II software. The SOPC Builder allows you to select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

Figure 1. SOPC Builder Design Flow



The Altera® DSP Builder integrates high-level algorithm and HDL development tools by combining the algorithm development, simulation, and verification capabilities of the MathWorks MATLAB and Simulink system-level design tools with VHDL synthesis and simulation tools and the Quartus II software. Figure 2 on page 159 shows the DSP Builder design flow.

Figure 2. DSP Builder Design Flow



Creating SOPC Designs with SOPC Builder

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

SOPC Builder library components include:

- Processors
- Intellectual property (IP) and peripherals

- Memory interfaces
- Communications peripherals
- Buses and interfaces, including the Avalon™ bus and AMBA™ high-performance bus (AHB)
- Digital signal processing (DSP) cores
- Software
- Header files
- Generic C drivers
- Operating system (OS) kernels

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

Creating the System

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

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

SOPC Builder provides library components (modules) for download, including processors, such as the Excalibur embedded processor stripe and NIOS processor, a UART, a timer, a PIO, an Avalon tri-state bridge, several simple memory interfaces, and OS/RTOS kernels. In addition, you can choose from an array of MegaCore®, OpenCore®, and OpenCore Plus megafunctions.

You can use the **System Contents** page of SOPC Builder to define the system. You can select library components in the module pool and display the added components in the module table. You can use the information in the module table or in a separate wizard to define the following component options:

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

Generating the System

Each project in SOPC Builder contains a system description file (PTF File), which contains all the settings, options, and parameters entered in the SOPC Builder. In addition, each module has a corresponding PTF File. During system generation, the SOPC Builder uses these files to generate the source code, software components, and simulation files for the system.

Once system design is complete, you can generate the system using the **System Generation** page of SOPC Builder or using the command line.

The SOPC builder software automatically generates all necessary logic to integrate processors, peripherals, memories, buses, arbitrators, and IP cores, and interfaces to logic and memory outside the system, and creates HDL source code that binds the components together.

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

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

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

For Information About	Refer To
Using SOPC Builder	*SOPC Builder Data Sheet* on the Altera web site *SOPC Builder User Guide* installed with the Quartus II software "Overview: Using SOPC Builder" in Quartus II Help *Application Note 308 (Building Embedded Processor Systems Using SOPC Builder & Excalibur Devices)* on the Altera web site

Creating DSP Designs with the DSP Builder

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

The DSP Builder also provides support for system-level debugging using the SignalTap® II Logic Analyzer. You can synthesize, compile and download the design, and then perform debugging, all through the MATLAB/Simulink interface.

Instantiating Functions

You can combine existing MATLAB functions and Simulink blocks with Altera DSP Builder blocks and MegaCore, OpenCore, and OpenCore Plus functions to link system-level design and implementation with DSP algorithm development.

To use MegaCore, OpenCore and OpenCore Plus functions in your design, you must download them before running the MATLAB/Simulink environment.

Generating Simulation Files

You can use the Simulink software to simulate your design, or use the SignalCompiler in the Simulink software generate files for simulating the design in EDA simulation tools.

The SignalCompiler generates a Tcl script for RTL simulation in the ModelSim software, and a VHDL test bench file that imports the Simulink input stimuli. You can use the Tcl script for automated simulation in the ModelSim software, or simulate in another EDA simulation tool with the VHDL test bench file.

Generating Synthesis Files

After simulation, you can perform synthesis on the SOPC design using an automated flow in the Quartus II, Mentor Graphics LeonardoSpectrum, or Synplicity Synplify software, or a manual flow in other synthesis tools. If the DSP Builder design is the top-level design, you can use either the automated or manual synthesis flows. If the DSP Builder design is not the top-level design, you must use the manual synthesis flow.

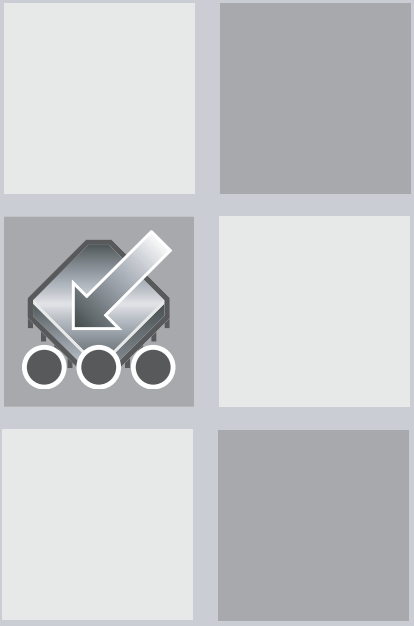
You can use the automated flow to control the entire synthesis and compilation flow from within the MATLAB/Simulink design environment. The SignalCompiler block creates VHDL Design Files and Tcl scripts, performs synthesis in the Quartus II, LeonardoSpectrum, or Synplify software, compiles the design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

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

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

Chapter Thirteen

Software Development



13

What's in Chapter 13:

Introduction

The Quartus® II Software Builder is an integrated programming tool that transforms software source files into a flash programming file or passive programming files for configuring an Excalibur™ device, or files that contain memory initialization data for the embedded processor stripe of an Excalibur device. You can use the Software Builder to process software source files for Excalibur designs, including designs created with the SOPC Builder and DSP Builder system-level design tools.

Using the Software Builder in the Quartus II Software

The Software Builder uses the ADS Standard Tools or GNUPro for ARM® software toolset to process software source files created by the Quartus II Text Editor or other Assembly or C/C++ language development tools. You can use the Software Builder to process the following software source files:

- Assembly Files (**.s**, .**asm**)
- C/C++ Include Files (**.h**)
- C Source Files (**.c**)
- C++ Source Files (**.cpp**)
- Library Files (**.a**)

The Software Builder can perform a software build on software source files with minimal assistance and allows you to customize processing for a particular design. You can also run a program or process for an Excalibur device from within the Quartus II software by using the Software Builder to run a command-line command during or after a software build.

Once you have specified software build settings, you can run the Software Builder by using the **Start Software Build** command (Processing menu).

You can also run a program or process for an Excalibur device from within the Quartus II software by using the Software Builder to run a command-line command during or after a software build.

Using the quartus_swb executable

You can also run the Software Builder separately at the command prompt or in a script by using the **quartus_swb** executable.

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

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

Specifying Software Build Settings

You can use the **Software Build Setting**s wizard or the **Software Build Settings** page of the **Settings** dialog box (Assignments menu) to specify software build settings before performing a software build.

Using the **Software Build Settings** wizard or the **Settings** dialog box, you can specify the following settings:

- The name of the software build settings for the project
- **CPU options**: architecture and software toolset, byte order, output file name, custom-build and post-build command-line commands, and programming file generation options
- **C/C++ Compiler options**: optimization levels, preprocessor definitions and include directories, and command-line commands
- **Assembler options**: preprocessor definitions, additional include directories, and command-line commands
- **Linker options**: object files, Library Files, library directories, link type, and command-line commands

You must use the **Toolset Directorie**s page of the **Settings** dialog box to specify the toolset directory that the Software Builder uses during a software build before you specify the toolset.

Generating Software Output Files

You can process designs and generate files that contain memory initialization data, passive programming files, and flash programming files by performing a software build in the Quartus II software. You can also use the **makeprogfile** utility (which is also used during a software build by the Quartus II software) and the stand-alone **MegaWizard® Plug-In Manager** to generate passive programming files and flash programming files outside the Quartus II software.

For more information on using the **makeprogfile** utility, type `makeprogfile -h` ↵ at a command prompt.

Using the Stand-Alone MegaWizard Plug-In Manager

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

`qmegawiz` ↵

The Software Builder automatically creates simulator initialization files every time you generate flash programming files with the Software Builder, or passive programming files with the Compiler or Software Builder. Simulator initialization files specify the initialization data for each address in the memory regions in the Excalibur embedded processor stripe.

Table 1. Simulator Initialization Files

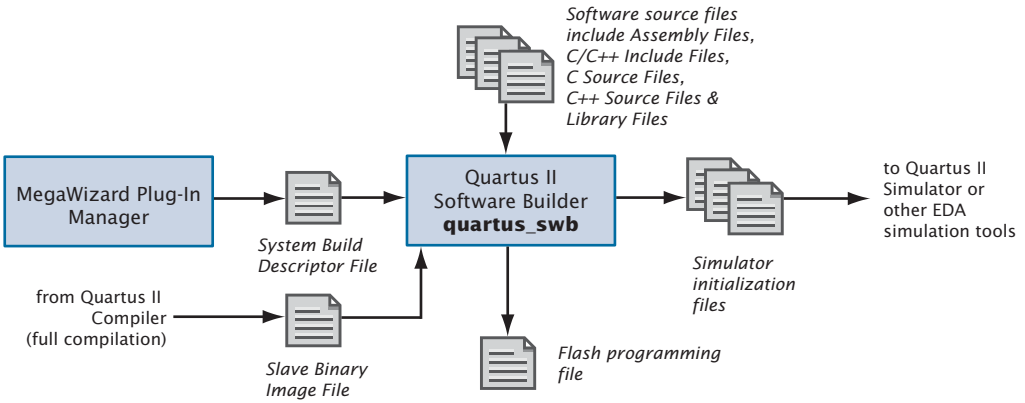
File Name	File Contents
memory.regs	Register initialization data
memory.sram0	SRAM0 initialization data
memory.sram1	SRAM1 initialization data
memory.dpram0	DPRAM0 initialization data
memory.dpram1	DPRAM1 initialization data

Generating Flash Programming Files

A flash programming file is a Hexadecimal (Intel-Format) File (**.hex**) that programs the flash memory from which an Excalibur device loads configuration and memory initialization data. The following steps describe the basic flow for creating a flash programming file with the Software Builder:

1. Create the software source files and add them to the project.
2. Run the **ARM-based Excalibur MegaWizard Plug-In** to generate a System Build Descriptor File (**.sbd**).
3. If you want the flash programming file to contain configuration data for the programmable logic device (PLD) portion of the Excalibur device, compile the design to generate a Slave Binary Image File (**.sbi**).
4. Specify the toolset directory and software build settings. To generate a flash programming file, you must specify the output file type and file name, turn on **Flash memory configuration**, and, if you are using a Slave Binary Image File, specify the optional Slave Binary Image File in the **CPU** page of the **Settings** dialog box (Assignments menu).
5. Start the software build.

Figure 1. Flash Programming Files Flow



To generate the flash programming files, the Software Builder performs the following steps:

- An assembler, C/C++ compiler, linker, and code converter converts software source files into a HEX File that contains Excalibur embedded processor stripe memory initialization data for the Excalibur device.
- A boot data object file is created from the HEX File, System Build Descriptor File, and Slave Binary Image File.
- A linker links the boot data file with a binary bootloader file to create an Executable and Linkable Format File (**.elf**).
- A code converter converts the Executable and Linkable Format File into a flash programming file with the name ***<project name>*_flash.hex**.

You can then use the **exc_flash_programmer** utility to program the information in the flash memory for the Excalibur device via Expansion Bus Interface zero (`EBI0`).

Generating Passive Programming Files

Passive programming files are used to configure Excalibur devices using the Passive Parallel Asynchronous (PPA), Passive Parallel Synchronous (PPS), or Passive Serial (PS) configuration schemes. You can use the Software Builder, the **makeprogfile** utility, or the Compiler to generate the following passive programming files:

- Hexadecimal (Intel-Format) Output Files (**.hexout**)
- Programmer Object Files (**.pof**)
- Raw Binary Files (**.rbf**)
- SRAM Object Files (**.sof**)
- Tabular Text Files (**.ttf**)

The following steps describe the basic flow for using the Software Builder to create a passive programming file:

1. Create the software source files and add them to the project.

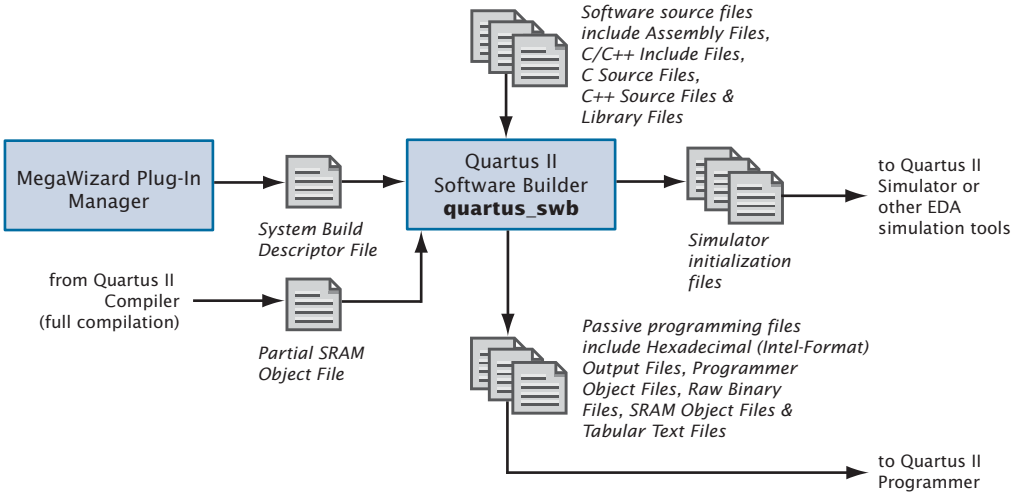
2. Run the **ARM-based Excalibur MegaWizard Plug-In** to generate a System Build Descriptor File.

3. Compile the design to generate a programmable logic Partial SRAM Object File (**.psof**).

4. Specify the software toolset directory and software build settings. To generate a flash programming file, you must specify the output file type and file name, turn on **Passive configuration**, and specify the PSOF in the **CPU** page of the **Settings** dialog box (Assignments menu).

5. Start the Software Builder.

Figure 2. Passive Programming Files Flow



To generate the passive programming files, the Software Builder performs the following steps:

- An assembler, C/C++ compiler, linker, and code converter converts the software source files into a HEX File that contains Excalibur embedded processor stripe memory initialization data for the Excalibur device.

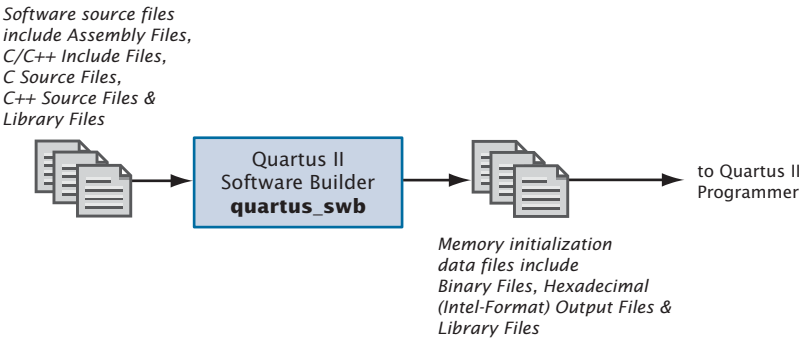
- The **makeprogfile** utility processes the HEX File, System Build Descriptor File, and PSOF to create one or more passive programming files.

Generating Memory Initialization Data Files

Binary Files (**.bin**), HEX Files, and Library Files (**.a**) contain the memory initialization data for the Excalibur embedded processor stripe. The following steps describe the basic flow for creating BIN Files, HEX Files, and Library Files with the Software Builder:

1. Create the software source files and add them to the project.
2. Specify the software toolset directory and software build settings. Use the **CPU** page of the **Settings** dialog box (Assignments menu) to specify the output file type and file name. If you selected a HEX File in the **Output file format** list, and you do not want to generate a flash programming file or generate passive programming files, select **None** under **Programming file generation**.
3. Start the software build.

Figure 3. Memory Initialization Data Files Flow



To generate the memory initialization files, the Software Builder performs the following steps:

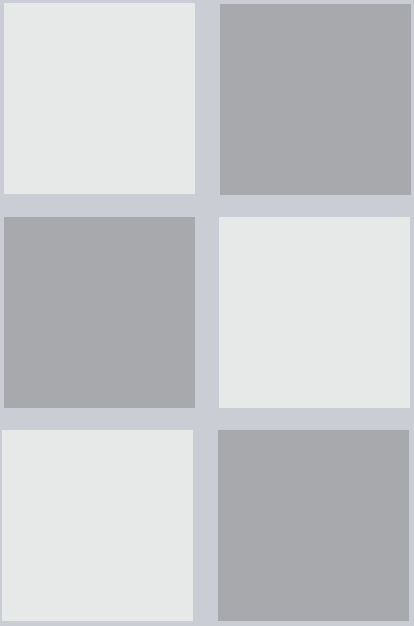
- An assembler and C/C++ compiler generates intermediate object files from the design's software source files.
- If you are generating BIN Files or HEX Files, the linker links the object files and generates an intermediate ELF File, and the code converter converts the ELF File into a BIN File or HEX File.

- If you are generating a Library File, the Software Builder uses the Software Builder Archiver to process the object files into a Library File.

For Information About	Refer To
Performing a Software Build	"Overview: Using the Software Builder" in Quartus II Help *Application Note 299 (System Development Tools for Excalibur Devices)* on the Altera web site Excalibur module in the Quartus II Tutorial
Generating passive programming files, and optional programming files for POFs and SOFs	"Generating Passive Programming Files" in Quartus II Help
Generating BIN Files, HEX Files, and Library Files and generating debugging information	"Generating Binary Files, Hexadecimal (Intel-Format) Files, Library Files & Motorola S-Record Files" in Quartus II Help "Overview: Checking Software Source Files and Output Files" in Quartus II Help

Chapter Fourteen

Installation, Licensing & Technical Support



14

What's in Chapter 14:

Installing the Quartus II Software

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

- Pentium PC operating at 400 MHz or faster, running one of the following operating systems:
  - Microsoft Windows NT version 4.0 (Service Pack 4 or later)
  - Microsoft Windows 2000
  - Microsoft Windows XP
- Pentium PC operating at 400 MHz or faster, running Red Hat Linux version 7.1, 7.2, or 8.0
- Sun Ultra workstation running Solaris version 7 or 8
- HP 9000 Series 700/800 workstation running HP-UX version 11.0 with ACE dated November, 1999 or later

For Information About	Refer To
System requirements and installation instructions	*Quartus II Installation & Licensing for PCs* manual *Quartus II Installation & Licensing for UNIX and Linux Workstations* manual *Altera CD Installation Guide*
Specific information about disk space and memory	Quartus II **readme.txt** file
Latest information on new features, device support, EDA interface support	*Quartus II Software Release Notes* on the Altera web site

Licensing the Quartus II Software

To use Altera®-provided software, you need to set up and obtain an Altera subscription license. An Altera subscription enables the following software:

- Altera Quartus II software
- Altera MAX+PLUS® II software
- Model Technology™ ModelSim®-Altera software

Altera offers several types of software subscriptions. Table 1 shows the different license and subscription options that are available.

Table 1. Altera License and Subscription Options

License Name	Description
FIXEDPC	A stand-alone PC license tied to a software guard (T-guard or "dongle")
FLOATPC	A floating network license for PC users with either a PC or UNIX license server
FLOATNET	A floating network license for PC, Solaris, and HP-UX users that are using a PC, Solaris, or HP-UX license server
FLOATLNX	A floating network license for PC users that are running Red Hat Linux and using either a PC, UNIX, or Linux license server
Quartus II Web Edition	A free, entry-level version of the Quartus II software that supports selected devices. The Quartus II Web Edition software is available from the Altera web site at **www.altera.com**

Customers who purchase selected development kits receive a free version of the Quartus II software for the PC and are given instructions on how to obtain a license for the software.

If you do not have a valid license file, you should request a new license file; however, you are also given the option to evaluate the Quartus II software, without programming file support, for 30 days. To use this option, when you start the Quartus II software, select the **Enable 30-day evaluation period** option. After the 30-day evaluation is over, you must obtain a valid license file in order to use the software.

The following steps describe the basic flow for licensing your software:

1. When you start the Quartus II software, if the software cannot detect a valid ASCII text license file, **license.dat**, you will see a prompt with an option to **Request updated license file from the web**. This option displays the Licensing section of the Altera web site, which allows you to request a license file.

   or

   If you want to request a license file at later time, you can go to the Licensing section of the Altera web site at **www.altera.com/licensing**.

2. Choose the link for the appropriate license type. Refer to Table 1 on page 177.

3. Specify the requested information.

4. After you receive a license file by e-mail, save it to a directory on your system.

5. If necessary, modify the license file for your license.

6. Set up and configure the FLEXlm license manager server for your system.

7. If you start the Quartus II software and have not already specified the license file location, it will give you an option to **Specify valid license file**. This option displays the **License Setup** tab of the **Options** dialog box (Tools menu). You can also specify the license file in your **System** control panel for Windows NT, Windows 2000, or Windows XP or in your **.cshrc** file for UNIX and Linux workstations.

For Information About	Refer To
Detailed information about licensing the Quartus II software, modifying the license file, and specifying the license file location	*Quartus II Installation & Licensing for PCs* manual *Quartus II Installation & Licensing for UNIX and Linux Workstations* manual
General information about Quartus II licensing	"Overview: Obtaining a License File" and "Specifying a License File" in Quartus II Help
Detailed information about Altera software licensing	*Application Note 205 (Altera Software Licensing)* on the Altera web site
Advanced information about Altera software licensing	*Application Note 229 (Troubleshooting Altera Software Licensing)* on the Altera web site

Getting Technical Support

The easiest way to get technical support is to use the mySupport web site and register for an Altera.com account. Your copy of the Quartus II software is registered at the time of purchase; however, in order to use the mySupport web site to view and submit service requests, you must also register for an Altera.com account. An Altera.com account is required only for using the mySupport web site; however, having an Altera.com account will also make it easier for you to use many other Altera web site features, such as the Download Center, Licensing Center, Altera Technical Training online class registration, or Buy On-Line-Altera eStore features.

To register for an Altera.com account user name and password, follow these steps:

1. Go to the mySupport web site:

   ✓ To start your web browser and connect to the mySupport web site while running the Quartus II software, choose **Altera on the Web > Quartus II Home Page** (Help menu).

   or

   ✓ Point your web browser to the mySupport web site at **www.altera.com/mysupport**.

2. Follow the instructions on the mySupport web site to register for an Altera.com account.

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

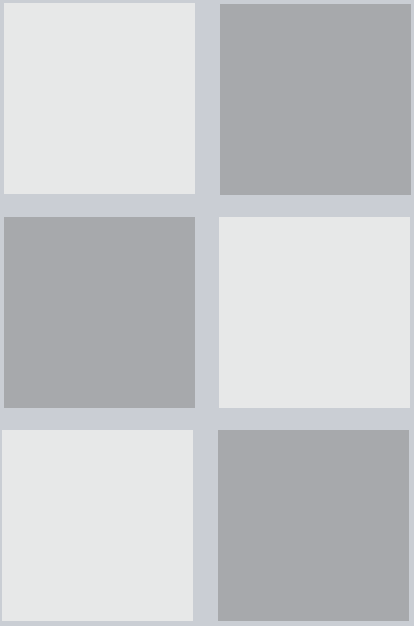
For information about other technical support resources, refer to Table 2.

Table 2. Quartus II Technical Support Resources

Resource	Description
Altera web site	**www.altera.com** The Altera web site provides information on Altera and all of its products.
Support Center	**www.altera.com/support** The Support Center section of the Altera web site gives you access to the mySupport web site, and also provides the Knowledge Database. In addition, it provides software and device support information as well as design examples that you can integrate into your design.
mySupport web site	**www.altera.com/mysupport** or choose **Altera on the Web > Quartus II Home Page** (Help menu) in the Quartus II software. The mySupport web site allows you to submit, view, and update technical support service requests.
Knowledge Database	**www.altera.com/kdb** The Knowledge Database helps you search for answers to technical questions and lets you search for key words and phrases in the Solutions documents and other Altera literature.
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) 544-7000 (7:00 a.m. to 5:00 p.m. Pacific time, M–F)

Chapter Fifteen

Documentation & Other Resources



15

What's in Chapter 15:

Getting Online Help

The Quartus® II software includes a platform-independent 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:

To search through a list of Help topics Choose **Index** (Help menu) to perform a search by using the **Index** tab.

To search through the full text of the Help system Choose **Search** (Help menu) to perform a search by using the **Search** tab.

To search an outline of Help topic categories Choose **Contents** (Help menu) to view the **Contents** tab.

To view help on a message Select the message on which you want to receive Help, and choose **Help** (right button pop-up menu). You can also choose **Messages** (Help menu) for a scrollable list of all messages.

To get Help on a menu command or dialog box Press F1 from a highlighted menu command or active dialog box for context-sensitive Help on that item.

To find a definition of a term Choose **Glossary** (Help menu) to view the Glossary list.

Working with Help Topics

To print Help topics from the **Contents** tab, select the Help folder or individual Help topic that you want to print, and choose **Print** (right button pop-up menu) or click the **Print** button on the toolbar. If you select a Help folder to print, you can choose to print all the topics in the folder. You can also use the **Print** command or **Print** button to print any individual Help topic you are viewing.

To search for a keyword in an open Quartus II Help topic, press Ctrl+F to open the **Find** dialog box, and type the search text, and then click **Find Next**.

For Information About	Refer To
Using Quartus II Help	"Using Quartus II Help Effectively" and "Help Menu Commands" in Quartus II Help "Using Quartus II Help" in the *Quartus II Installation & Licensing for PCs* manual and *Quartus II Installation & Licensing for UNIX and Linux Workstations* manual

Using the Quartus II Online Tutorial

The online tutorial introduces you to the features of the Quartus II design software. It shows you how to create and process your own logic designs quickly and easily. The modular design of the Basic and Advanced tutorials allows you to choose the areas of the Quartus II software that you want to learn about:

- The Basic tutorial guides you through the steps required to create, perform timing analysis on, simulate, and program a sample finite impulse response (FIR) filter design, called **fir_filter**.
- The Advanced tutorial builds on the training in the Basic tutorial, focusing on the LogicLock feature and Excalibur and Stratix device features.

To start the Quartus II tutorial after you have successfully installed the Quartus II software:

✓ Choose **Tutorial** (Help menu).

After you start the tutorial, the Quartus II window resizes to allow you to view the Tutorial window and the Quartus II software simultaneously.

Using the Quartus II Tutorial to Search for Literature

You must have installed support for the APEX 20K EP20K100E device if you want to complete the Basic or LogicLock tutorial. In addition, you must have installed support for the Excalibur EPXA10 and Stratix EP1S25 devices if you want to complete the Excalibur and Stratix Advanced tutorial modules.

The tutorial is designed for display online. However, if you want to print one or more of the tutorial modules, click the **Printing Options** button located at the beginning of each module and then click the link to open the appropriate printable version.

Other Quartus II Software Documentation

Table 1 shows the additional software documentation that is available for the Quartus II software:

Table 1. Additional Quartus II Documentation (Part 1 of 2)

Document	Description	Where to Find It
Quartus II Software Release Notes	Provides late-breaking information about new features, device support, EDA interface support, and known issues and workarounds	The Altera® web site
Quartus II Installation & Licensing for PCs manual	Provides detailed information about software requirements, installation, and licensing for PCs	In Quartus II subscription packages and on the Altera web site
Quartus II Installation & Licensing for UNIX and Linux Workstations manual	Provides detailed information about software requirements, installation, and licensing for UNIX and Linux workstations	In Quartus II subscription packages and on the Altera web site

Table 1. Additional Quartus II Documentation (Part 2 of 2)

Document	Description	Where to Find It
Altera CD Installation Guide	Provides basic installation instructions for all of the Altera CD-ROMs that are included in Quartus II subscription packages	In Quartus II subscription packages and on the Altera web site
Quartus II **readme.txt** file	Provides information about memory, disk space, and system requirements	On Quartus II software CD-ROMs and installed with the Quartus II software
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 web site

Other Altera Literature

The Literature section of the Altera web site at **www.altera.com** provides documentation on many subjects that are related to the Quartus II software. Many of these documents are also available on the Altera Documentation Library CD or from Altera Literature Services. You can also purchase printed sets of documentation from the ShopAltera web site at **www.shopaltera.com**.

Altera provides literature that includes some of 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 web site 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.

Searching the Knowledge Database for Altera Literature

You can use the Knowledge Database, which is available from the Support Center section of the Altera web site at **www.altera.com/kdb**, to search for key words or phrases in all the literature that is available on the Altera web site.

Revision History

The information contained in the *Introduction to Quartus II* manual version 3.0 revision 1 supersedes information published in previous versions.

Minor typographical changes were made to this version.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W