Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Introduction to the Quartus® II Software

Version 10.0

QUARTUS® II

Introduction to the Quartus® II Software

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

Contents

Preface vii

Chapter 1: Design Flow 1
- Introduction 2
- Graphical User Interface Design Flow 3
- Command-Line Executables 7
  - Using Standard Command-Line Commands & Scripts 10
  - Using Tcl Commands 12
- Design Methodologies and Planning 14
  - Incremental Design Flows 14
  - Using LogicLock Regions 15
  - Using LogicLock Regions in Incremental Compilation Flows 16

Chapter 2: Design Entry 19
- Introduction 20
- Creating a Project 21
- Creating a Design 22
  - Using the Quartus II Block Editor 22
  - Using the Quartus II Symbol Editor 22
  - Using the Quartus II Text Editor 23
  - Using Verilog HDL, VHDL, & AHDL 23
  - Using the State Machine Editor 24
- Using Altera Megafunctions 24
  - Using Intellectual Property (IP) Megafunctions 25
  - Using the MegaWizard Plug-In Manager 27
  - Instantiating Megafunctions in the Quartus II Software 27
    - Instantiation in Verilog HDL & VHDL 28
    - Using the Port & Parameter Definition 28
    - Inferring Megafunctions 28
  - Instantiating Megafunctions in EDA Tools 28
    - Using the Black Box Methodology 29
    - Instantiation by Inference 29
    - Using the Clear Box Methodology 29
- Constraint Entry 31
  - Using the Assignment Editor 32
  - Using the Pin Planner 33
  - The Settings Dialog Box 35
  - Making Timing Constraints 36
  - Creating Design Partitions 36
  - Creating Design Partitions with the Design Partitions Planner 37

Chapter 3: Synthesis 39
- Introduction 40
- Using Quartus II Verilog HDL & VHDL Integrated Synthesis 41
  - Using Quartus II Synthesis Netlist Optimization Options 43
- Using the Design Assistant to Check Design Reliability 44
- Analyzing Synthesis Results With the Netlist Viewers 45

The RTL Viewer 45
The State Machine Viewer 47
The Technology Map Viewer 48

Chapter 4: Place and Route 51
Introduction 52
Using Incremental Compilation 53
Analyzing Fitting Results 54
Using the Messages Window to View Fitting Results 55
Using the Report Window or Report File to View Fitting Results 56
Using the Chip Planner to Analyze Results 56
Using the Design Assistant to Check Design Reliability 58
Optimizing the Fit 58
Using Location Assignments 58
Setting Options that Control Place & Route 59
Setting Fitter Options 59
Setting Physical Synthesis Optimization Options 59
Setting Individual Logic Options that Affect Fitting 60
Using the Resource Optimization Advisor 60
Using the Design Space Explorer 63

Chapter 5: Timing Analysis and Design Optimization 65
Introduction 66
Running the TimeQuest Timing Analyzer 66
Specifying Timing Constraints 68
Viewing Timing Information for a Path 70
Viewing Timing Delays with the Technology Map Viewer 72
Timing Closure 73
Using the Chip Planner 74
Chip Planner Tasks And Layers 74
Making Assignments 74
Using the Timing Optimization Advisor 75
Using Netlist Optimizations to Achieve Timing Closure 75
Using LogicLock Regions to Preserve Timing 77
Using the Design Space Explorer to Achieve Timing Closure 78
Power Analysis with the PowerPlay Power Analyzer 78
PowerPlay Early Power Estimator Spreadsheets 80

Chapter 6: Programming & Configuration 83
Introduction 84
Creating and Using Programming Files 85

Chapter 7: Debugging and Engineering Change Managment 89
Introduction 90
Using the SignalTap II Logic Analyzer 91
Analyzing SignalTap II Data 92
Using an External Logic Analyzer 93

Using SignalProbe 94
Using the In-System Memory Content Editor 94
Using the In-System Sources and Probes Editor 96
Using the RTL Viewer & Technology Map Viewer For Debugging 97
Using the Chip Planner for Debugging 97
Identifying Delays & Critical Paths With the Chip Planner 99
Modifying Resource Properties With the Resource Property Editor 101
Viewing & Managing Changes with the Change Manager 103
Verifying ECO Changes 104

Chapter 8: EDA Tool Support 105
Introduction 106
EDA Synthesis Tools 108
EDA Simulation Tools 109
Generating Simulation Output Files 110
Simulation Libraries 111
Timing Analysis with EDA Tools 112
Using the PrimeTime Software 113
Using the Tau Software 113
Formal Verification 114
Using the Cadence Encounter Conformal Software 116

Chapter 9: System Requirements, Licensing & Technical Support 117
Installing the Quartus II Software 118
Licensing the Quartus II Software 119
Getting Technical Support 119
Getting Online Help 121
Starting the Quartus II Interactive Tutorial 122
Other Quartus II Software Documentation 123
Other Altera Literature 124
Documentation Conventions 125

Preface

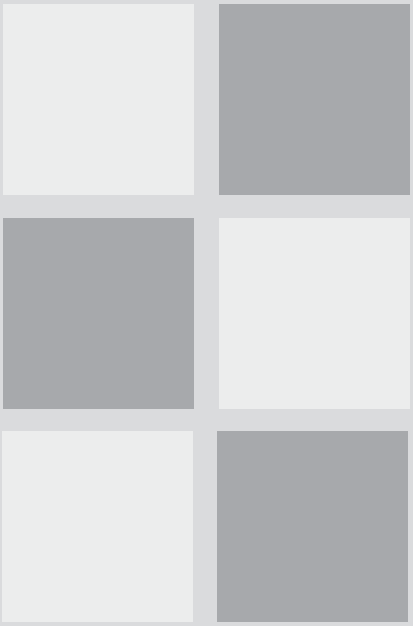
This manual is designed for the novice Altera® Quartus® II design software user and provides an overview of the capabilities of the Quartus II software in programmable logic design. The Altera Quartus II software is the most comprehensive environment available for system-on-a-programmable-chip (SOPC)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

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

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

Chapter One

Design Flow



1

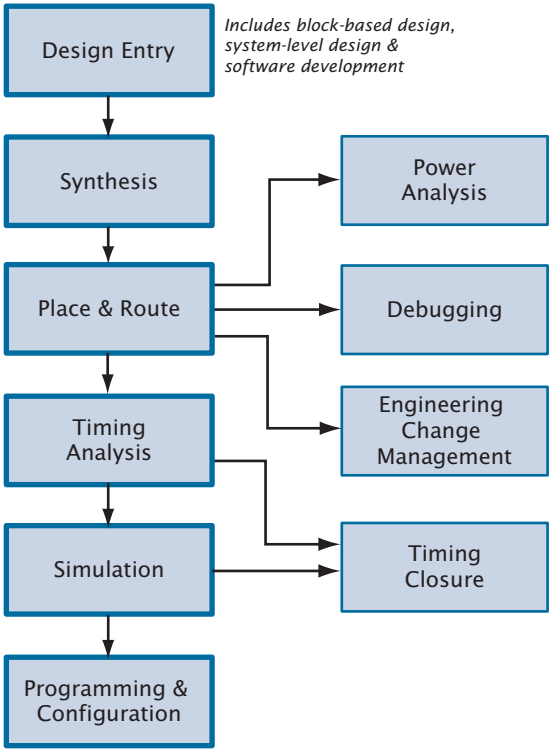
What's in Chapter 1:

Introduction	2
Graphical User Interface Design Flow	3
Command-Line Executables	7
Design Methodologies and Planning	14

Introduction

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

Figure 1. Quartus II Design Flow

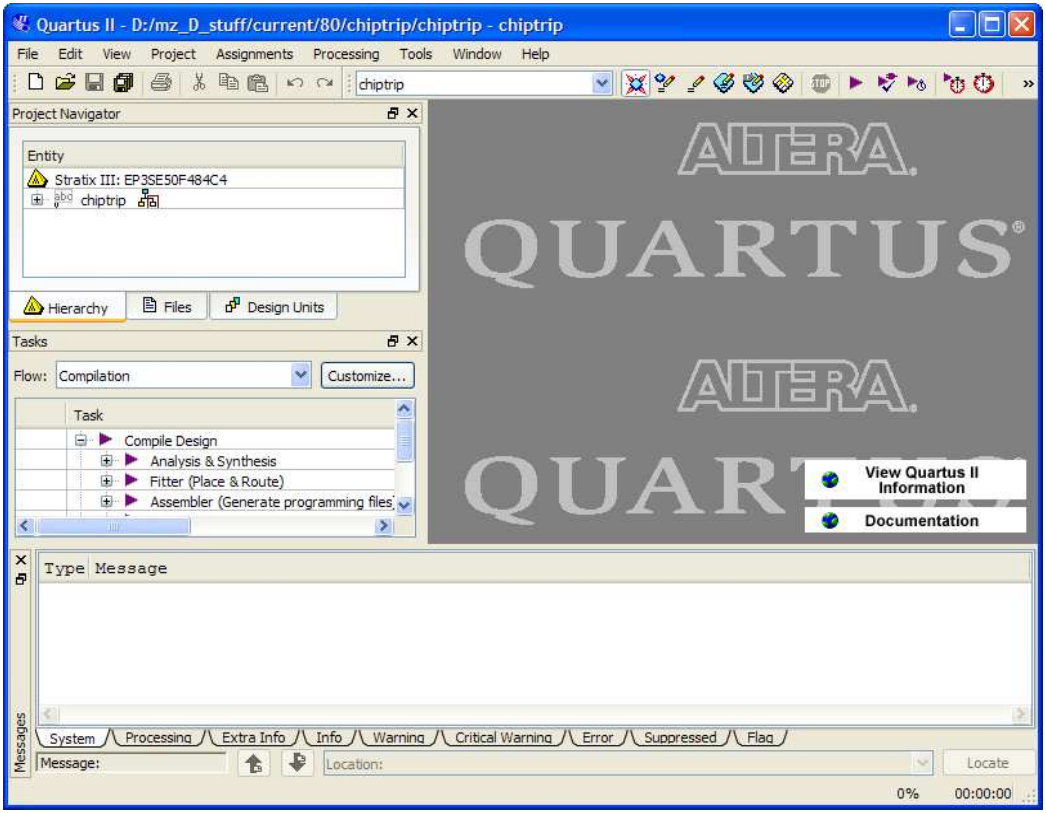


In addition, the Quartus II software allows you to use the Quartus II graphical user interface and command-line interface for each phase of the design flow. You can use one of these interfaces for the entire flow, or you can use different options at different phases.

Graphical User Interface Design Flow

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

Figure 2. Quartus II Graphical User Interface



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

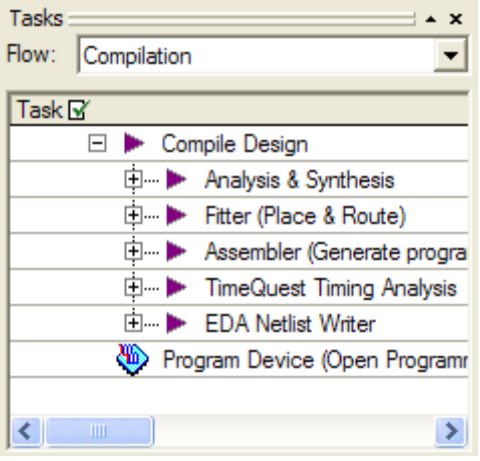
- Analysis & Synthesis
- Partition Merge*

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

To run all Compiler modules as part of a full compilation, on the Processing menu, click **Start Compilation**. You can also run each module individually by pointing to **Start** on the Processing menu, and then clicking the command for the module you want to start.

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

Figure 3. Tasks Window



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

Table 1. Commands for Common Compiler Flows

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

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

The following steps describe the basic design flow for using the Quartus II GUI:

1. To create a new project and specify a target device or device family, on the File menu, click **New Project Wizard**.
2. Use the Text Editor to create a Verilog HDL, VHDL, or Altera Hardware Description Language (AHDL) design.
3. Use the Block Editor to create a block diagram with symbols that represent other design files, or to create a schematic.
4. Use the **MegaWizard® Plug-In Manager** to generate custom variations of megafunctions and IP functions to instantiate in your design, or create a system-level design by using SOPC Builder or DSP Builder.
5. Specify any initial design constraints using the Assignment Editor, the Pin Planner, the **Settings** dialog box, the **Device** dialog box, the Chip Planner, the Design Partitions window, or the Design Partition Planner.

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

7. Synthesize the design with Analysis & Synthesis.

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

9. (Optional) Generate a functional simulation netlist for your design and perform a functional simulation with an EDA simulation tool.

10. Place and route the design with the Fitter.

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

12. Use an EDA simulation tool to perform timing simulation for the design.

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

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

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

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

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

Command-Line Executables

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

Figure 4. Command-Line Design Flow

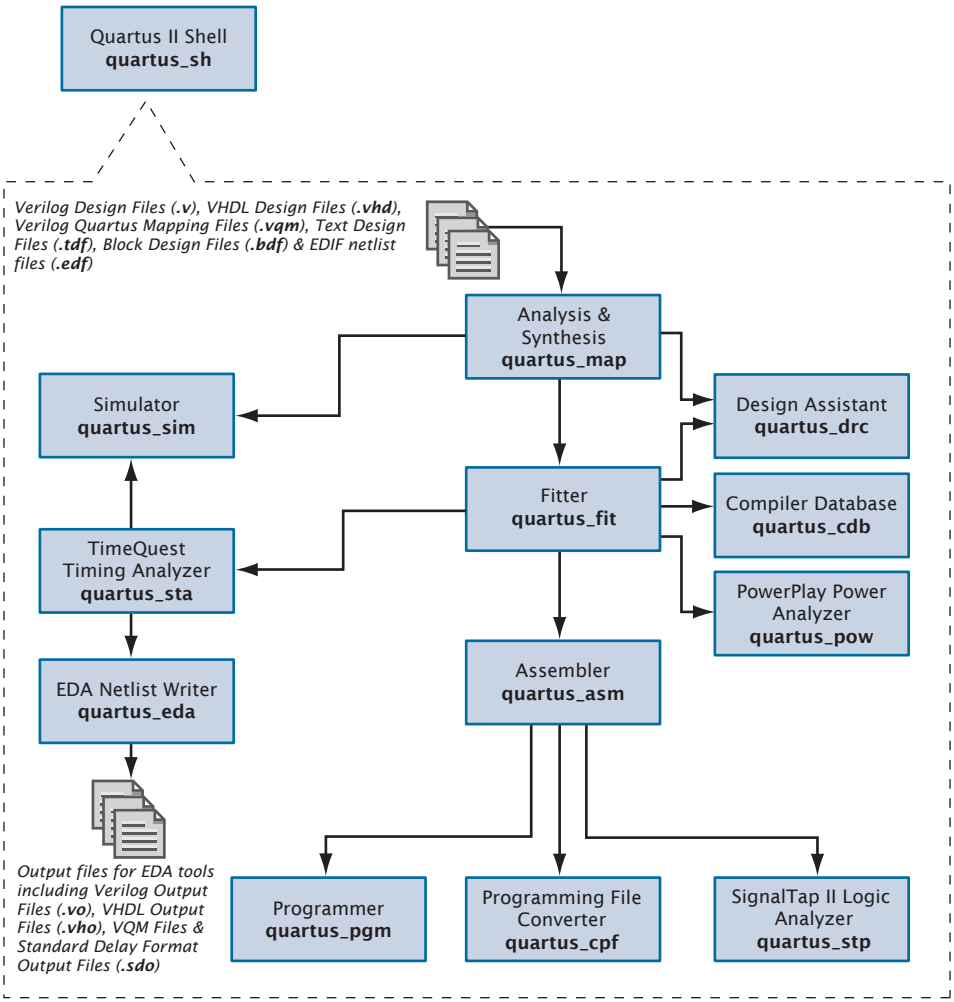


Table 2. Command-Line Executables (Part 1 of 2)

Executable Name	Title	Function
quartus_map	Analysis & Synthesis	Creates a project if one does not already exist, and then creates the project database, synthesizes your design, and performs technology mapping on design files of the project.
quartus_fit	Fitter	Places and routes a design. Analysis & Synthesis must be run successfully before running the Fitter.
quartus_drc	Design Assistant	Checks the reliability of a design based on a set of design rules. Design Assistant is especially useful for checking the reliability of a design before migrating the design to HardCopy devices. Either Analysis & Synthesis or the Fitter must be run successfully before running the Design Assistant.
quartus_sta	TimeQuest Timing Analyzer	Performs ASIC-style timing analysis of the circuit using constraints entered in Synopsys Design Constraint format.
quartus_asm	Assembler	Creates one or more programming files for programming or configuring the target device. The Fitter must be run successfully before running the Assembler.
quartus_eda	EDA Netlist Writer	Generates netlist files and other output files for use with other EDA tools. Analysis & Synthesis, the Fitter, or the Timing Analyzer must be run successfully before running the EDA Netlist Writer, depending on the options used.
quartus_cdb	Compiler Database Interface (including VQM Writer)	Imports and exports version-compatible databases and merges partitions. Generates internal netlist files, including Verilog Quartus Mapping Files, for the Quartus II Compiler database so they can be used for back-annotation and for the LogicLock feature, and back-annotates device and resource assignments to preserve the fit for future compilations. Either the Fitter or Analysis & Synthesis must be run successfully before running the Compiler Database Interface.

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

Executable Name	Title	Function
quartus_jli	Jam STAPL Player	Reads and executes Jam Files (**.jam**) in the STAPL format. A single Jam File can perform several functions, such as programming, configuring, verifying, erasing, and blank-checking a programmable device in a JTAG chain.
quartus_jbcc	Jam Compiler	The Quartus II JAM Compiler converts Jam/STAPL files to Jam Byte Code Files (**.jbc**) which store data for programming, configuring, verifying, and blank-checking one or more devices in a JTAG chain.
quartus_sim	Simulator	Performs functional or timing simulation on your design. Analysis & Synthesis must be run before performing a functional simulation. Timing Analysis must be run before performing a timing simulation.
quartus_pow	Power Analyzer	Analyzes and estimates total dynamic and static power consumed by a design. Computes toggle rates and static probabilities for output signals. The Fitter must be run successfully before running the PowerPlay Power Analyzer.
quartus_pgm	Programmer	Programs Altera devices.
quartus_cpf	Programming File Converter	Converts programming files to secondary programming file formats.
quartus_stp	SignalTap II Logic Analyzer	Sets up your SignalTap II File (**.stp**). When it is run after the Assembler, the SignalTap II Logic Analyzer captures signals from internal device nodes while the device is running at speed.
quartus_si	SSN Analyzer	Estimates and reports the simultaneous switching noise contributions to voltage and timing noise for device pins.
quartus_sh	Tcl Shell	Provides overall control of Quartus II projects and compilation flows, as well as a Tcl shell.

Getting Help On the Quartus II Executables

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

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

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

`quartus_sh --qhelp` ↵

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

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

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

Using Standard Command-Line Commands & Scripts

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

Figure 5 shows an example of a standard batch file. The example demonstrates how to create a project, perform Analysis & Synthesis, perform place and route, perform timing analysis, and generate programming files for the **filtref** design that is included with the Quartus II software. If you have installed the **filtref** design, it is in the **/altera/***<version number>***/qdesigns/fir_filter** directory. You can run the four commands in Figure 5 from a command prompt in the new project directory, or you can store them in a batch file or shell script.

Figure 5. Example of a Command-Line Script

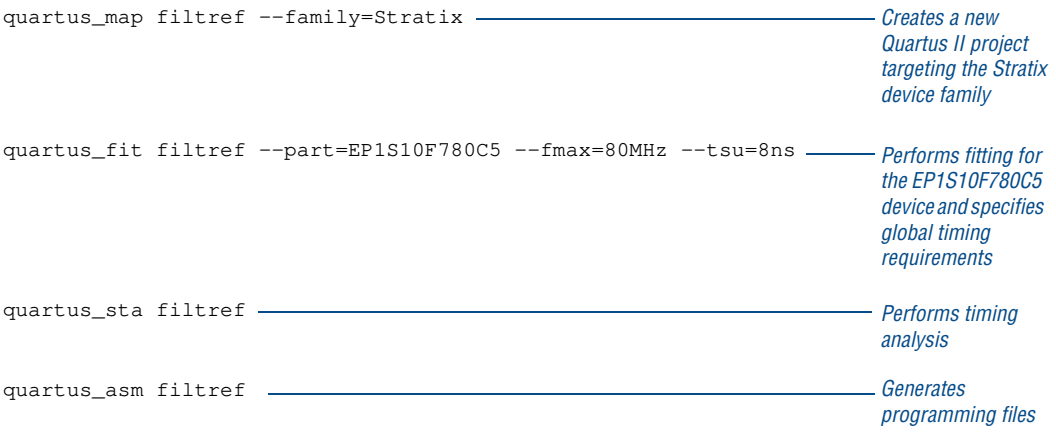


Figure 6 shows an excerpt from a command-line script for use on a Linux workstation. The script assumes that the Quartus II tutorial project called **fir_filter** exists in the current directory. The script analyzes every design file in the **fir_filter** project and reports any files that contain syntax errors.

Figure 6. Example of a Linux Command-Line Shell Script

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

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

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

Using Tcl Commands

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

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

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

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

Getting Help On Tcl Commands

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

```
quartus_sh --qhelp ↵
```

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

Figure 7 shows an example of a Tcl script.

Figure 7. Example of a Tcl Script

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

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

# Set Family
set_global_assignment -name family CYCLONE

# Set Device
set_global_assignment -name device ep1c6f256c6

# Optimize for speed
set_global_assignment -name optimization_technique speed

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

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

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

load_package flow
execute_flow -compile

# Close Project
project_close
```

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

Design Methodologies and Planning

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

Incremental Design Flows

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

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

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

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

Using LogicLock Regions

A LogicLock region is defined by its size and location on the device. You can specify the size and location of a region, or direct the Quartus II software to create them automatically.

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

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

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

In addition, you can add path-based assignments (based on source and destination nodes), wildcard assignments, and Fitter priority for path-based and wildcard assignments to LogicLock regions. Setting the priority allows you to specify the order in which the Quartus II software resolves conflicting path-based and wildcard assignments.

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

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

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

Using LogicLock Regions in Incremental Compilation Flows

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

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