This chapter discusses the various tools and methods for constraining and re-constraining Quartus II designs in different design flows, both with the Quartus II GUI and with Tcl to facilitate a scripted flow.

Constraints, sometimes known as assignments or logic options, control the way the Quartus II software implements a design for an FPGA. Constraints are also central in the way that the TimeQuest Timing Analyzer and the PowerPlay Power Analyzer inform synthesis, placement, and routing. There are several types of constraints:

- Global design constraints and software settings, such as device family selection, package type, and pin count.

- Entity-level constraints, such as logic options and placement assignments.

- Instance-level constraints.

- Pin assignments and I/O constraints.

User-created constraints are contained in one of two files: the Quartus II Settings File (**.qsf**) or, in the case of timing constraints, the Synopsys Design Constraints file (**.sdc**). Constraints and assignments made with the **Device** dialog box, **Settings** dialog box, Assignment Editor, Chip Planner, and Pin Planner are contained in the Quartus II Settings File. The **.qsf** file contains project-wide and instance-level assignments for the current revision of the project in Tcl syntax. You can create separate revisions of your project with different settings, and there is a separate **.qsf** file for each revision.

The TimeQuest Timing Analyzer uses industry-standard Synopsys Design Constraints, also using Tcl syntax, that are contained in Synopsys Design Constraints (**.sdc**) files. The TimeQuest Timing Analyzer GUI is a tool for making timing constraints and viewing the results of subsequent analysis.

There are several ways to constrain a design, each potentially more appropriate than the others, depending on your tool chain and design flow. You can constrain designs for compilation and analysis in the Quartus II software using the GUI, as well as using Tcl syntax and scripting. By combining the Tcl syntax of the **.qsf** files and the **.sdc** files with procedural Tcl, you can automate iteration over several different settings, changing constraints and recompiling.

Constraining Designs with the Quartus II GUI

In the Quartus II GUI, the New Project Wizard, **Device** dialog box, and **Settings** dialog box allow you to make global constraints and software settings. The Assignment Editor and Pin Planner are spreadsheet-style interfaces for constraining your design at the instance or entity level. The Assignment Editor and Pin Planner make constraint types and values available based on global design characteristics such as the targeted device. These tools help you verify that your constraints are valid before compilation by allowing you to pick only from valid values for each constraint.

The TimeQuest Timing Analyzer GUI allows you to make timing constraints in SDC format and view the effects of those constraints on the timing in your design. Before running the TimeQuest timing analyzer, you must specify initial timing constraints that describe the clock characteristics, timing exceptions, and external signal arrival and required times. The Quartus II Fitter optimizes the placement of logic in the device to meet your specified constraints.

> For more information about timing constraints and the TimeQuest Timing Analyzer, refer to *About TimeQuest Timing Analysis* in Quartus II Help.

Global Constraints

Global constraints affect the entire Quartus II project and all of the applicable logic in the design. Many of these constraints are simply project settings, such as the targeted device selected for the design. Synthesis optimizations and global timing and power analysis settings can also be applied with globally. Global constraints are often made when running the New Project Wizard, or in the **Device** dialog box or the **Settings** dialog box, early project development.

The following are the most common types of global constraints:

- Target device specification
- Top-level entity of your design, and the names of the design files included in the project
- Operating temperature limits and conditions
- Physical synthesis optimizations
- Analysis and synthesis options and optimization techniques
- Verilog HDL and VHDL language versions used in your project
- Fitter effort and timing driven compilation settings
- **.sdc** files for the TimeQuest timing analyzer to use during analysis as part of a full compilation flow

Settings that direct compilation and analysis flows in the Quartus II software are also stored in the Quartus II Settings File for your project, including the following global software settings:

- Early Timing Estimate mode
- Settings for EDA tool integration such as third-party synthesis tools, simulation tools, timing analysis tools, and formal verification tools.

■ Settings and settings file specifications for the Quartus II Assembler, SignalTap II Logic Analyzer, PowerPlay power analyzer, and SSN Analyzer.

Global constraints and software settings stored in the Quartus II settings file are specific to each revision of your design, allowing you to control the operation of the software differently for different revisions. For example, different revisions can specify different operating temperatures and different devices, so that you can compare results.

Only the valid assignments made in the Assignment Editor are saved in the Quartus II Settings File, which is located in the project directory. When you make a design constraint, the new assignment is placed on a new line at the end of the file.

When you create or update a constraint in the GUI, the Quartus II software displays the equivalent Tcl command in the **System** tab of the Messages window. You can use the displayed messages as references when making assignments using Tcl commands.

⊘ For more information about specifying initial global constraints and software settings, refer to *Setting up and Running a Compilation* in Quartus II Help.

For more information about how the Quartus II software uses Quartus II Settings Files, refer to the *Managing Quartus II Projects* chapter in volume 2 of the *Quartus II Handbook*.

Node, Entity, and Instance-Level Constraints

Node, entity, and instance-level constraints constrain a particular segment of the design hierarchy, as opposed to the entire design. In the Quartus II software GUI, most instance-level constraints are made with the Assignment Editor, Pin Planner, and Chip Planner. Both the Assignment Editor and Pin Planner aid you in correctly constraining your design, both passively, through device-and-assignment-determined pick lists, and actively, through live I/O checking.

You can assign logic functions to physical resources on the device, using location assignments with the Assignment Editor or the Chip Planner. Node, entity, and instance-level constraints take precedence over any global constraints that affect the same sections of the design hierarchy. You can edit and view all node and entity-level constraints you created in the Assignment Editor, or you can filter the assignments by choosing to view assignments only for specific locations, such as DSP blocks.

The Pin Planner helps you visualize, plan, and assign device I/O pins to ensure compatibility with your PCB layout. The Pin Planner provides a graphical view of the I/O resources in the target device package. You can quickly locate various I/O pins and assign them design elements or other properties. The Quartus II software uses these assignments to place and route your design during device programming. The Pin Planner also helps with early pin planning by allowing you to plan and assign IP interface or user nodes not yet defined in the design.

The Pin Planner Task window provides one-click access to common pin planning tasks. After clicking a pin planning task, you view and highlight the results in the Report window by selecting or deselecting I/O types.You can quickly identify I/O banks, VREF groups, edges, and differential pin pairings to assist you in the pin planning process. You can verify the legality of new and existing pin assignments with the live I/O check feature and view the results in the Live I/O Check Status window.

The Chip Planner allows you to view the device from a variety of different perspectives, and you can make precise assignments to specific floorplan locations. With the Chip Planner, you can adjust existing assignments to device resources, such as pins, logic cells, and LABs using drag and drop features and a graphical interface. You can also view equations and routing information, and demote assignments by dragging and dropping assignments to various regions in the Regions window.

? For more information about the Assignment Editor, refer to *About the Assignment Editor* in Quartus II Help. For more information about the Chip Planner, refer to *About the Chip Planner* in Quartus II Help. For more information about the Pin Planner, refer to *Assigning Device I/O Pins in Pin Planner* in Quartus II Help.

Probing Between Components of the Quartus II GUI

The Assignment Editor, Chip Planner, and Pin Planner let you locate nodes and instances in the source files for your design in other Quartus II viewers. You can select a cell in the Assignment Editor spreadsheet and locate the corresponding item in another applicable Quartus II software window, such as the Chip Planner. To locate an item from the Assignment Editor in another window, right-click the item of interest in the spreadsheet, point to **Locate**, and click the appropriate command.

You can also locate nodes in the Assignment Editor and other constraint tools from other windows within the Quartus II software. First, select the node or nodes in the appropriate window. For example, select an entity in the **Entity** list in the **Hierarchy** tab in the Project Navigator, or select nodes in the Chip Planner. Next, right-click the selected object, point to **Locate**, and click **Locate in Assignment Editor**. The Assignment Editor opens, or it is brought to the foreground if it is already open.

? For more information about the Assignment Editor, refer to *About the Assignment Editor* in Quartus II Help. For more information about the Chip Planner, refer to *About the Chip Planner* in Quartus II Help. For more information about the Pin Planner, refer to *Assigning Device I/O Pins in Pin Planner* in Quartus II Help.

SDC and the TimeQuest Timing Analyzer

You can make individual timing constraints for individual entities, nodes, and pins with the Constraints menu of the TimeQuest Timing Analyzer. The TimeQuest Timing Analyzer GUI provides easy access to timing constraints, and reporting, without requiring knowledge of SDC syntax. As you specify commands and options in the GUI, the corresponding SDC or Tcl command appears in the Console. This lets you know exactly what constraint you have added to your Synopsys Design Constraints file, and also enables you to learn SDC syntax for use in scripted flows. The GUI also provides enhanced graphical reporting features.

Individual timing assignments override project-wide requirements. You can also assign timing exceptions to nodes and paths to avoid reporting of incorrect or irrelevant timing violations. The TimeQuest timing analyzer supports point-to-point timing constraints, wildcards to identify specific nodes when making constraints, and assignment groups to make individual constraints to groups of nodes.

? For more information about timing constraints and the TimeQuest Timing Analyzer, refer to *About TimeQuest Timing Analysis* in Quartus II Help.

Constraining Designs with Tcl

Because **.sdc** files and **.qsf** files are both in Tcl syntax, you can modify these files to be part of a scripted constraint and compilation flow. With Quartus II Tcl packages, Tcl scripts can open projects, make the assignments procedurally that would otherwise be specified in a **.qsf** file, compile a design, and compare compilation results against known goals and benchmarks for the design. Such a script can further automate the iterative process by modifying design constraints and recompiling the design.

For more information about controlling the Quartus II software with Tcl, refer to *About Quartus II Tcl Scripting* in Quartus II Help.

Quartus II Settings Files and Tcl

QSF files use Tcl syntax, but, unmodified, are not executable scripts. However, you can embed QSF constraints in a scripted iterative compilation flow, where the script that automates compilation and custom results reporting also contains the design constraints. Example 1–1 shows an example QSF file with boilerplate comments removed.

Example 1–1. Quartus II Settings File

```
set_global_assignment -name FAMILY "Cyclone II"
set_global_assignment -name DEVICE EP2C35F672C6
set_global_assignment -name TOP_LEVEL_ENTITY chiptrip
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 10.0
set_global_assignment -name PROJECT_CREATION_TIME_DATE "11:45:02  JUNE 08, 2010"
set_global_assignment -name LAST_QUARTUS_VERSION 10.0
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING \
-section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
set_global_assignment -name LL_ROOT_REGION ON -section_id "Root Region"
set_global_assignment -name LL_MEMBER_STATE LOCKED -section_id "Root Region"
set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "3.3-V LVTTL"
set_location_assignment PIN_P2 -to clk2
set_location_assignment PIN_AE4 -to ticket[0]
set_location_assignment PIN_J23 -to ticket[2]
set_location_assignment PIN_Y12 -to timeo[1]
set_location_assignment PIN_N2 -to reset
set_location_assignment PIN_R2 -to timeo[7]
set_location_assignment PIN_P1 -to clk1
set_location_assignment PIN_M3 -to ticket[1]
set_location_assignment PIN_AE24 -to ~LVDS150p/nCEO~
set_location_assignment PIN_C2 -to accel
set_location_assignment PIN_K4 -to ticket[3]
set_location_assignment PIN_B3 -to stf
set_location_assignment PIN_T9 -to timeo[0]
set_location_assignment PIN_M5 -to timeo[6]
set_location_assignment PIN_J8 -to dir[1]
set_location_assignment PIN_C5 -to timeo[5]
set_location_assignment PIN_F6 -to gt1
set_location_assignment PIN_P24 -to timeo[2]
set_location_assignment PIN_B2 -to at_altera
set_location_assignment PIN_P3 -to timeo[4]
set_location_assignment PIN_M4 -to enable
set_location_assignment PIN_E3 -to ~ASDO~
set_location_assignment PIN_E5 -to dir[0]
set_location_assignment PIN_R25 -to timeo[3]
set_location_assignment PIN_D3 -to ~nCSO~
set_location_assignment PIN_G4 -to gt2
set_global_assignment -name MISC_FILE "D:/altera/chiptrip/chiptrip.dpf"
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
set_global_assignment -name POWER_PRESET_COOLING_SOLUTION \
"23 MM HEAT SINK WITH 200 LFPM AIRFLOW"
set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
set_global_assignment -name SDC_FILE chiptrip.sdc
```

Example 1–1 shows the way that the `set_global_assignment` Quartus II Tcl command makes all global constraints and software settings, with `set_location_assignment` constraining each I/O node in the design to a physical pin on the device.

However, after you initially create the Quartus II Settings File for your design, you can export the contents to a procedural, executable Tcl (**.tcl)** file. You can then use that generated script to restore certain settings after experimenting with other constraints. You can also use the generated Tcl script to archive your assignments instead of archiving the Quartus II Settings file itself.

To export your constraints as an executable Tcl script, on the Project menu, click **Generate Tcl File for Project**. Example 1–2 shows the constraints in Example 1–1 converted to an executable Tcl script.

Example 1–2. Generated Tcl Script for a Quartus II Project (Part 1 of 2)

```
# Quartus II: Generate Tcl File for Project
# File: chiptrip.tcl
# Generated on: Tue Jun 08 13:08:48 2010

# Load Quartus II Tcl Project package
package require ::quartus::project

set need_to_close_project 0
set make_assignments 1

# Check that the right project is open
if {[is_project_open]} {
    if {[string compare $quartus(project) "chiptrip"]} {
        puts "Project chiptrip is not open"
        set make_assignments 0
    }
} else {
    # Only open if not already open
    if {[project_exists chiptrip]} {
        project_open -revision chiptrip chiptrip
    } else {
        project_new -revision chiptrip chiptrip
    }
    set need_to_close_project 1
}

# Make assignments
if {$make_assignments} {
set_global_assignment -name FAMILY "Cyclone II"
set_global_assignment -name DEVICE EP2C35F672C6
set_global_assignment -name TOP_LEVEL_ENTITY chiptrip
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 10.0
set_global_assignment -name PROJECT_CREATION_TIME_DATE "11:45:02  JUNE 08, 2010"
set_global_assignment -name LAST_QUARTUS_VERSION 10.0
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING \
-section_id Top
```

Example 1–2. Generated Tcl Script for a Quartus II Project (Part 2 of 2)

```
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
set_global_assignment -name LL_ROOT_REGION ON -section_id "Root Region"
set_global_assignment -name LL_MEMBER_STATE LOCKED -section_id "Root Region"
set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "3.3-V LVTTL"
set_location_assignment PIN_P2 -to clk2
set_location_assignment PIN_AE4 -to ticket[0]
set_location_assignment PIN_J23 -to ticket[2]
set_location_assignment PIN_Y12 -to timeo[1]
set_location_assignment PIN_N2 -to reset
set_location_assignment PIN_R2 -to timeo[7]
set_location_assignment PIN_P1 -to clk1
set_location_assignment PIN_M3 -to ticket[1]
set_location_assignment PIN_AE24 -to ~LVDS150p/nCEO~
set_location_assignment PIN_C2 -to accel
set_location_assignment PIN_K4 -to ticket[3]
set_location_assignment PIN_B3 -to stf
set_location_assignment PIN_T9 -to timeo[0]
set_location_assignment PIN_M5 -to timeo[6]
set_location_assignment PIN_J8 -to dir[1]
set_location_assignment PIN_C5 -to timeo[5]
set_location_assignment PIN_F6 -to gt1
set_location_assignment PIN_P24 -to timeo[2]
set_location_assignment PIN_B2 -to at_altera
set_location_assignment PIN_P3 -to timeo[4]
set_location_assignment PIN_M4 -to enable
set_location_assignment PIN_E3 -to ~ASDO~
set_location_assignment PIN_E5 -to dir[0]
set_location_assignment PIN_R25 -to timeo[3]
set_location_assignment PIN_D3 -to ~nCSO~
set_location_assignment PIN_G4 -to gt2
set_global_assignment -name MISC_FILE "D:/altera/chiptrip/chiptrip.dpf"
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
set_global_assignment -name POWER_PRESET_COOLING_SOLUTION \
"23 MM HEAT SINK WITH 200 LFPM AIRFLOW"
set_global_assignment -name POWER_BOARD_THERMAL_MODEL "NONE (CONSERVATIVE)"
set_global_assignment -name SDC_FILE chiptrip.sdc

    # Commit assignments
    export_assignments

    # Close project
    if {$need_to_close_project} {
       project_close
    }
}
```

After setting initial values for variables to control constraint creation and whether or not the project needs to be closed at the end of the script, the generated script checks to see if a project is open. If a project is open but it is not the correct project, in this case, **chiptrip**, the script prints `Project chiptrip is not open` to the console and does nothing else.

If no project is open, the script determines if **chiptrip** exists in the current directory. If the project exists, the script opens the project. If the project does not exist, the script creates a new project and opens the project.

The script then creates the constraints. After creating the constraints, the script writes the constraints to the Quartus II Settings File and then closes the project.

Timing Analysis with Synopsys Design Constraints and Tcl

Timing constraints used in analysis by the Quartus II TimeQuest Timing Analyzer are stored in **.sdc** files. Because they use Tcl syntax, the constraints in **.sdc** files can be incorporated into other scripts for iterative timing analysis. Example 1–3 shows a basic **.sdc** file for the **chiptrip** project.

Example 1–3. Initial .sdc file for the chiptrip Project

```
# ------------------------------------------

set_time_unit ns
set_decimal_places 3

# ------------------------------------------
#
create_clock -period 10.0 -waveform { 0 5.0 } clk2 -name clk2
create_clock -period 4.0 -waveform { 0 2.0 } clk1 -name clk1

# clk1 -> dir* : INPUT_MAX_DELAY = 1 ns
set_input_delay -max 1ns -clock clk1  [get_ports dir*]
# clk2 -> time* : OUTPUT_MAX_DELAY = -2 ns
set_output_delay -max -2ns -clock clk2  [get_ports time*]
```

Similar to the constraints in the Quartus II Settings File, you can make the SDC constraints in Example 1–3 part of an executable timing analysis script, as shown in example Example 1–4.

Example 1–4. Tcl Script Making Basic Timing Constraints and Performing Mult-Corner Timing Analysis

```
project_open chiptrip
create_timing_netlist

#
# Create Constraints
#
create_clock -period 10.0 -waveform { 0 5.0 } clk2 -name clk2
create_clock -period 4.0 -waveform { 0 2.0 } clk1 -name clk1

# clk1 -> dir* : INPUT_MAX_DELAY = 1 ns
set_input_delay -max 1ns -clock clk1  [get_ports dir*]
# clk2 -> time* : OUTPUT_MAX_DELAY = -2 ns
set_output_delay -max -2ns -clock clk2  [get_ports time*]

#
# Perform timing analysis for several different sets of operating conditions
#
foreach_in_collection oc [get_available_operating_conditions] {
    set_operating_conditions $oc
    update_timing_netlist

    report_timing -setup -npaths 1
    report_timing -hold -npaths 1
    report_timing -recovery -npaths 1
    report_timing -removal -npaths 1
    report_min_pulse_width -nworst 1
}

delete_timing_netlist
project_close
```

The script in Example 1–4 opens the project, creates a timing netlist, then constrains the two clocks in the design and applies input and output delay constraints. The clock settings and delay constraints are identical to those in the **.sdc** file shown in Example 1–3. The next section of the script updates the timing netlist for the constraints and performs multi-corner timing analysis on the design.

A Fully Iterative Scripted Flow

You can use the **::quartus::flow** Tcl package and other packages in the Quartus II Tcl API to add flow control to modify constraints and recompile your design in an automated flow. You can combine your timing constraints with the other constraints for your design, and embed them in an executable Tcl script that also iteratively compiles your design as different constraints are applied.

Each time such a modified generated script is run, it can modify the **.qsf** file and **.sdc** file for your project based on the results of iterative compilations, effectively replacing these files for the purposes of archiving and version control using industry-standard source control methods and practices.

This type of scripted flow can include automated compilation of a design, modification of design constraints, and recompilation of the design, based on how you foresee results and pre-determine next-step constraint changes in response to those results.

(?) For more information about the Quartus II Tcl API, refer to *API Functions for Tcl* in Quartus II Help. For more information about controlling the Quartus II software with Tcl scripts, refer to *About Quartus II Tcl Scripting* in Quartus II Help.

Document Revision History

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

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

Date	Version	Changes
November 2012	12.1.0	Update Pin Planner description for task and report windows.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	Rewrote chapter to more broadly cover all design constraint methods. Removed procedural steps and user interface details, and replaced with links to Quartus II Help.
November 2009	9.1.0	■ Added two notes. ■ Minor text edits.
March 2009	9.0.0	■ Revised and reorganized the entire chapter. ■ Added section "Probing to Source Design Files and Other Quartus II Windows" on page 1–2. ■ Added description of node type icons (Table 1–3). ■ Added explanation of wildcard characters.

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

Date	Version	Changes
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated Quartus II software 8.0 revision and date.

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