Quartus Prime Pro Edition Handbook Volume 2: Design Implementation and Optimization

Constraints, sometimes known as assignments or logic options, control the way the Quartus® Prime 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 Prime 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 Prime 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 Prime 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 GUI

In the Quartus Prime 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 Prime Fitter optimizes the placement of logic in the device to meet your specified constraints.

Global Constraints

Global constraints affect the entire Quartus Prime 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.

Common Types of Global Constraints

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

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

- 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 Prime Assembler, SignalTap II Logic Analyzer, PowerPlay power analyzer, and SSN Analyzer.

Global Constraints and Software Settings

Global constraints and software settings stored in the Quartus Prime 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 Prime 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 Prime 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.

Related Information
Managing Quartus Prime Projects
For more information about how the Quartus Prime software uses Quartus Prime Settings Files

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 Prime 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.

Constraining Designs with the Pin Planner

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 Prime 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.

Constraining Designs with the Chip Planner

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.

Probing Between Components of the Quartus Prime 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 Prime viewers.

You can select a cell in the Assignment Editor spreadsheet and locate the corresponding item in another applicable Quartus Prime 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.

Send Feedback

You can also locate nodes in the Assignment Editor and other constraint tools from other windows within the Quartus Prime 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.

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.

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 Prime 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.

Quartus Prime 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.

```
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
```

The example shows the way that the `set_global_assignment` Quartus Prime 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 Prime 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 Prime Settings file itself.

To export your constraints as an executable Tcl script, on the Project menu, click **Generate Tcl File for Project**.

```
# Quartus Prime: Generate Tcl File for Project
# File: chiptrip.tcl
# Generated on: Tue Jun 08 13:08:48 2010
# Load Quartus Prime 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
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 Prime Settings File and then closes the project.

Timing Analysis with Synopsys Design Constraints and Tcl

Timing constraints used in analysis by the Quartus Prime 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.

```
# ----------------------------------------
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 Prime Settings File, you can make the SDC constraints part of an executable timing analysis script.

```
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 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 the first example. 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 Prime 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.

Related Information

- **API Functions for Tcl**
- **About Quartus Prime Tcl Scripting**

Document Revision History

Table 1-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*.
June 2014	14.0.0	Formatting updates.
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 Prime Help.
November 2009	9.1.0	• Added two notes. • Minor text edits.

Date	Version	Changes
March 2009	9.0.0	• Revised and reorganized the entire chapter. • Added section "Probing to Source Design Files and Other Quartus Prime Windows" on page1–2. • Added description of node type icons (Table1–3). • Added explanation of wildcard characters.
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated Quartus Prime software 8.0 revision and date.

Related Information

Quartus Handbook Archive

For previous versions of the *Quartus Prime Handbook*, refer to the Quartus Handbook Archive.
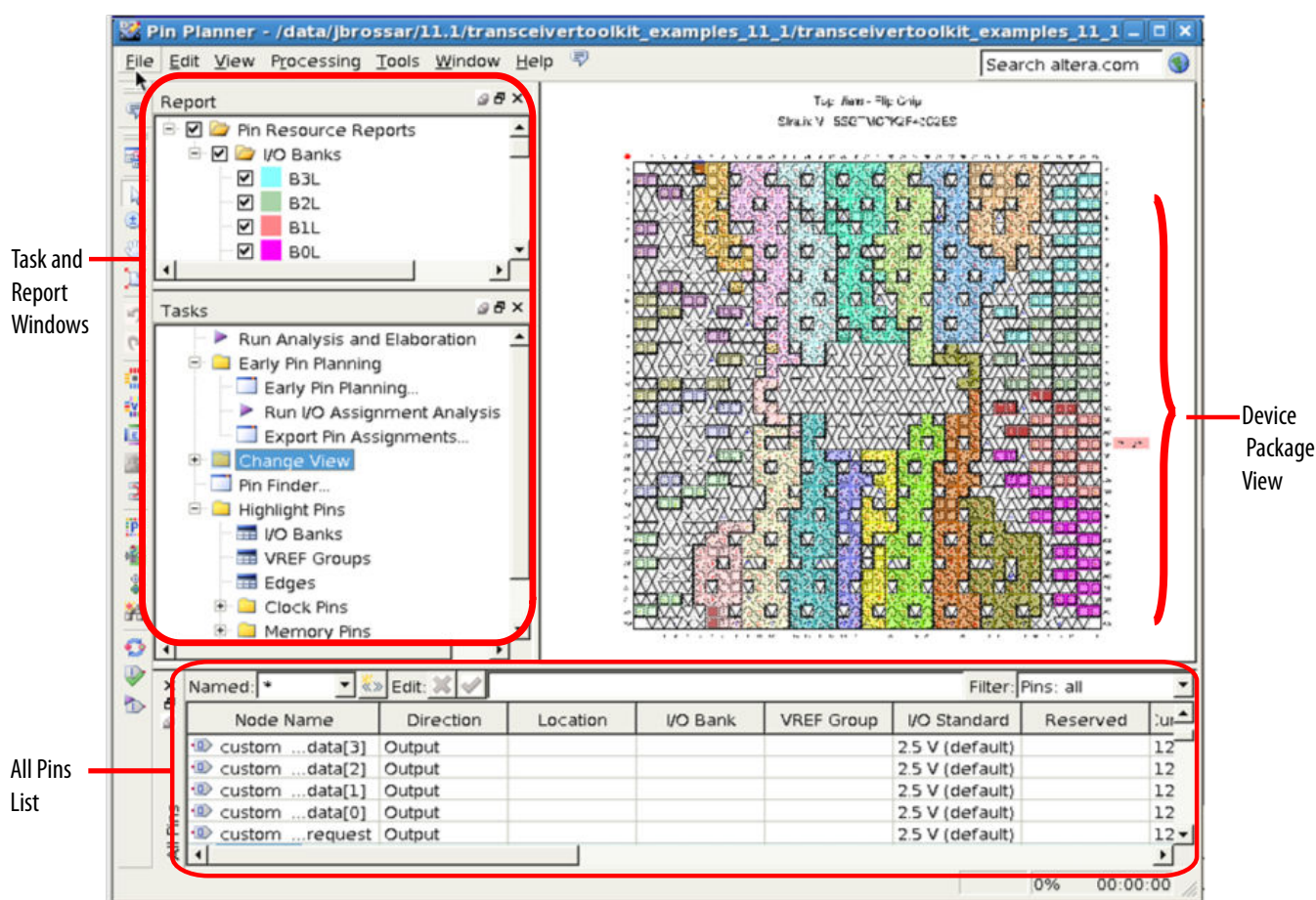
This document describes efficient planning and assignment of the I/O pins in your target device. You should consider I/O standards, pin placement rules, and your PCB characteristics early in the design phase.

Figure 2-1: Pin Planner GUI

Task and Report Windows

Device Package View

All Pins List

Table 2-1: Quartus Prime I/O Pin Planning Tools

I/O Planning Task	Click to Access
Plan interfaces and device periphery	**Tools** > **BluePrint Platform Designer**
Edit, validate, or export pin assignments	**Assignments** > **Pin Planner**
Validate pin assignments against design rules	**Processing** > **Start I/O Assignment Analysis**

For more information about special pin assignment features for the Arria 10 SoC devices, refer to *Instantiating the HPS Component* in the *Arria 10 Hard Processor System Technical Reference Manual.*

Related Information

- **BluePrint Planning Overview** on page 3-2
- **Using the BluePrint Interface** on page 3-3
- **Instantiating the HPS Component**

I/O Planning Overview

You should plan and assign I/O pins in your design for compatibility with your target device and PCB characteristics. Plan I/O pins early to reduce design iterations and develop an accurate PCB layout sooner. You can assign expected nodes not yet defined in design files, including interface IP core signals, and then generate a top-level file. Specify interfaces for memory, high-speed I/O, device configuration, and debugging tools in your top-level file. The top-level file instantiates the next level of design hierarchy and includes interface port information.

Use the Pin Planner to view, assign, and validate device I/O pin logic and properties. Alternatively, you can enter I/O assignments in a Tcl script, or directly in HDL code. The Pin Planner Task window provides one-click access to I/O planning steps. You can filter and search the nodes in the design. You can define custom groups of pins for assignment. Instantly locate and highlight specific pin types for assignment or evaluation, such as I/O banks, VREF groups, edges, DQ/DQS pins, hard memory interface pins, PCIe hard IP interface pins, hard processor system pins, and clock region input pins. Assign design elements, I/O standards, interface IP, and other properties to the device I/O pins by name or by drag and drop. You can then generate a top-level design file for I/O validation.

Use I/O assignment analysis to fully analyze I/O analysis against VCCIO, VREF, electromigration (current density), Simultaneous Switching Output (SSO), drive strength, I/O standard, PCI_IO clamp diode, and I/O pin direction compatibility rules.

Basic I/O Planning Flow

The following steps describe the basic flow for assigning and verifying I/O pin assignments:

1. Click **Assignments > Device** and select a target device that meets your logic, performance, and I/O requirements. Consider and specify /O standards, voltage and power supply requirements, and available I/O pins.
2. Click **Assignments > Pin Planner**.
3. Assign I/O properties to match your device and PCB characteristics, including assigning logic, I/O standards, output loading, slew rate, and current strength.
4. Click **Run I/O Assignment Analysis** in the Tasks pane to validate assignments and generate a synthesized design netlist. Correct any problems reported.
5. Click **Processing > Start Compilation**. During compilation, the Quartus Prime software runs I/O assignment analysis.
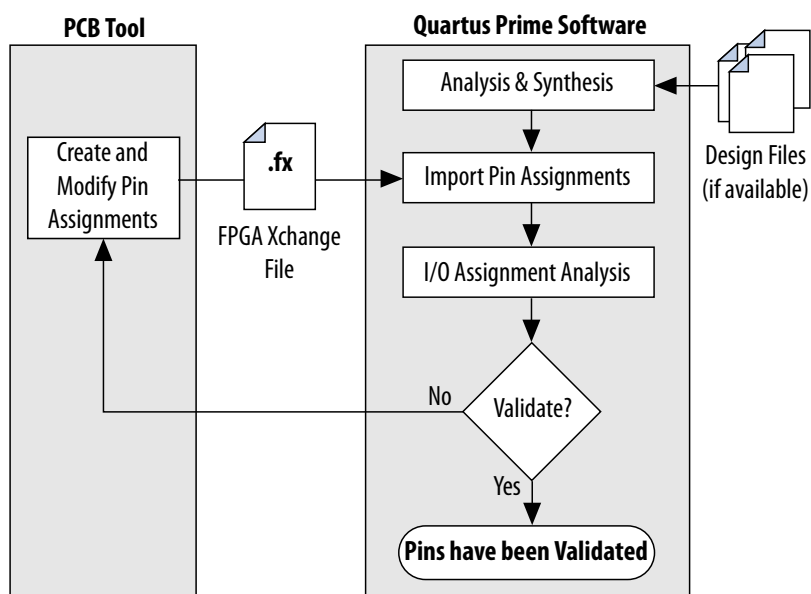
Integrating PCB Design Tools

You can integrate PCB design tools into your work flow to help correctly map pin assignments to the symbols your system circuit schematics and board layout. The Quartus Prime software integrates with board layout tools by allowing import and export of pin assignment information in Quartus Prime Settings Files (**.qsf**), Pin-Out File (**.pin**), and FPGA Xchange-Format File (**.fx**) files. You can integrate PCB tools in the the following ways:

Table 2-2: Integrating PCB Design Tools

PCB Tool Integration	Supported PCB Tool
Define and validate I/O assignments in the Pin Planner, and then export the assignments to the PCB tool for validation	Mentor Graphics® I/O DesignerCadence Allegro
Define I/O assignments in your PCB tool, and then import the assignments into the Pin Planner for validation	Mentor Graphics® I/O DesignerCadence Allegro

Figure 2-2: PCB Tool Integration
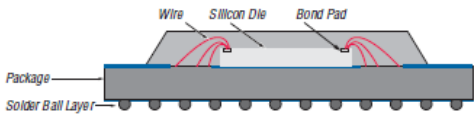
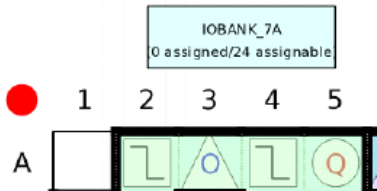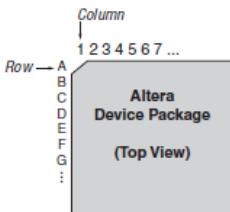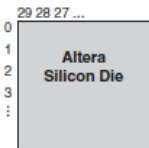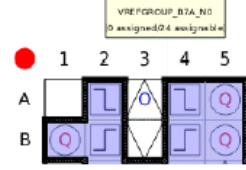For more information about incorporating PCB design tools, refer to the *Cadence PCB Design Tools Support* and *Mentor Graphics PCB Design Tools Support* chapters in volume 2 of the *Quartus Prime Handbook.*

Related Information

- **Mentor Graphics PCB Design Tools Support** on page 16-1

Altera Device Terms

The following terms describe Altera device and I/O structures:

Terms	Description	Diagram
Device Package (BGA example)	Ceramic or plastic heat sink surface mounted with FPGA die and I/O pins or solder balls. In a wire bond BGA example, copper wires connect the bond pads to the solder balls of the package. Click **View > Show > Package Top** or **View > Show > Package Bottom** in Pin Planner	
I/O Bank	I/O pins are grouped in I/O banks for assignment of I/O standards. Each numbered bank has its own voltage source pins, called VCCIO pins, for high I/O performance. The specified VCCIO pin voltage is between 1.5 V and 3.3 V. Each bank supports multiple pins with different I/O standards. All pins in a bank must use the same VCCIO signal. Click **View > Show > I/O Banks** in Pin Planner.	
I/O Pin	A wire lead or small solder ball on the package bottom or periphery. Each pin has an alphanumeric row and column number. I, O, Q, S, X, and Z are never used. The alphabet is repeated and prefixed with the letter A when exceeded. All I/O pins display by default.	
Pad	I/O pins are connected to pads located on the perimeter of the top metal layer of the silicon die. Each pad is numbered with an ID starting at 0, and increments by one in a counterclockwise direction around the device. Click **View > Pad View** in Pin Planner.	
VREF Pin Group	A group of pins including one dedicated VREF pin required by voltage-referenced I/O standards. A VREF group contains a smaller number of pins than an I/O bank. This maintains the signal integrity of the VREF pin. One or more VREF groups exist in an I/O bank. The pins in a VREF group share the same VCCIO and VREF voltages. Click **View > Show > Show VREF Groups** in Pin Planner..	

Assigning I/O Pins

Use the Pin Planner to visualize, modify, and validate I/O assignments in a graphical representation of the target device. To assign I/O pins, locate the device I/O pin(s) for assignment, enter properties for the pin(s), and validate the legality of the assignment.

Send Feedback

Note: You can increase the accuracy of I/O assignment analysis by reserving specific device pins to accommodate undefined but expected I/O.

To assign I/O pins in the Pin Planner, follow these steps:

1. Open a Quartus Prime project, and then click **Assignments** > **Pin Planner**.
2. Click **Processing** > **Start Analysis & Elaboration** to elaborate the design and display **All Pins** in the device view.
3. To locate or highlight pins for assignment, click **Pin Finder** or a pin type under **Highlight Pins** in the Tasks pane.
4. (Optional) To define a custom group of nodes for assignment, select one or more nodes in the **Groups** or **All Pins** list, and then click **Create Group**.
5. Enter assignments of logic, I/O standards, interface IP, and properties for device I/O pins in the **All Pins** spreadsheet, or by drag and drop into the package view.
6. To assign properties to differential pin pairs, click **Show Differential Pin Pair Connections**. A red connection line appears between positive (p) and negative (n) differential pins.
7. (Optional) To create board trace model assignments, right-click an output or bidirectional pin, and then click **Board Trace Model**. For differential I/O standards, the board trace model uses a differential pin pair with two symmetrical board trace models. Specify board trace parameters on the positive end of the differential pin pair. The assignment applies to the corresponding value on the negative end of the differential pin pair.
8. To run a full I/O assignment analysis, click **Run I/O Assignment Analysis**. The Fitter reports analysis results. Only reserved pins are analyzed prior to design synthesis.

Assigning to Exclusive Pin Groups

You can designate groups of pins for exclusive assignment. When you assign pins to an **Exclusive I/O Group**, the Fitter does not place the signals in the same I/O bank with any other exclusive I/O group. For example, if you have a set of signals assigned exclusively to group_a, and another set of signals assigned to group_b, the Fitter ensures placement of each group in different I/O banks.

Assigning Slew Rate and Drive Strength

You can designate the device pin slew rate and drive strength. These properties affect the pin's outgoing signal integrity. Use either the **Slew Rate** or **Slow Slew Rate** assignment to adjust the drive strength of a pin with the **Current Strength** assignment.

Note: The slew rate and drive strength apply during I/O assignment analysis.
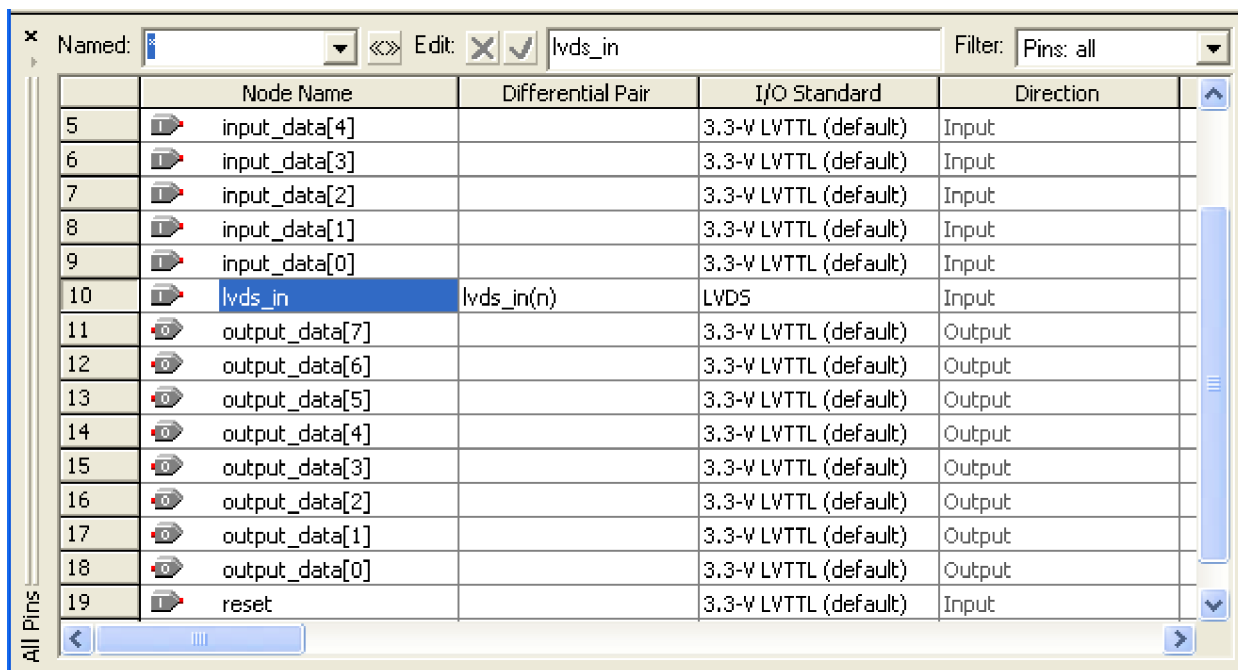
Assigning Differential Pins

When you use the Pin Planner to assign a differential I/O standard to a single-ended top-level pin in your design, it automatically recognizes the negative pin as part of the differential pin pair assignment and creates the negative pin for you. The Quartus Prime software writes the location assignment for the negative pin to the **.qsf**; however, the I/O standard assignment is not added to the **.qsf** for the negative pin of the differential pair.

The following example shows a design with lvds_in top-level pin, to which you assign a differential I/O standard. The Pin Planner automatically creates the differential pin, lvds_in(n) to complete the differential pin pair.

Note: If you have a single-ended clock that feeds a PLL, assign the pin only to the positive clock pin of a differential pair in the target device. Single-ended pins that feed a PLL and are assigned to the negative clock pin device cause the design to not fit.

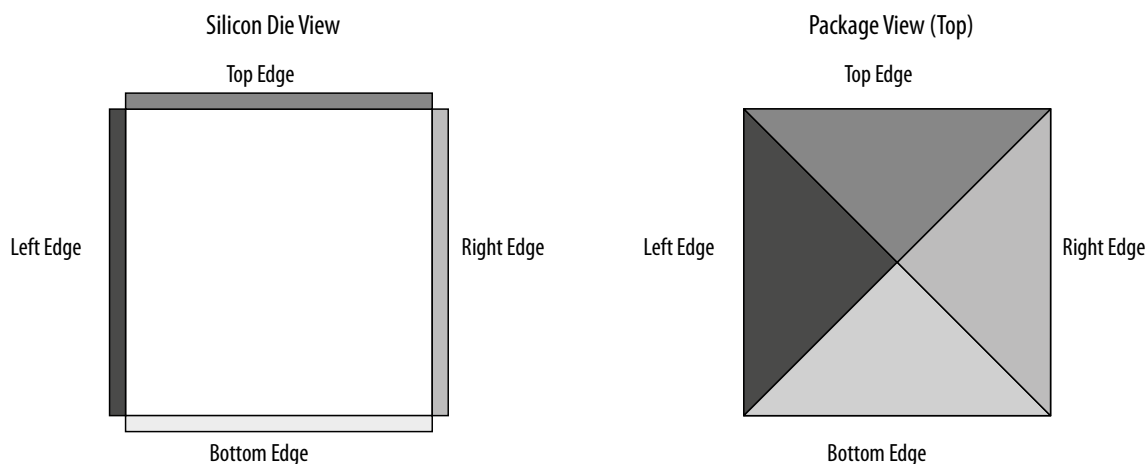**Figure 2-3: Creating a Differential Pin Pair in the Pin Planner**

		Node Name	Differential Pair	I/O Standard	Direction
5		input_data[4]		3.3-V LVTTL (default)	Input
6		input_data[3]		3.3-V LVTTL (default)	Input
7		input_data[2]		3.3-V LVTTL (default)	Input
8		input_data[1]		3.3-V LVTTL (default)	Input
9		input_data[0]		3.3-V LVTTL (default)	Input
10		lvds_in	lvds_in(n)	LVDS	Input
11		output_data[7]		3.3-V LVTTL (default)	Output
12		output_data[6]		3.3-V LVTTL (default)	Output
13		output_data[5]		3.3-V LVTTL (default)	Output
14		output_data[4]		3.3-V LVTTL (default)	Output
15		output_data[3]		3.3-V LVTTL (default)	Output
16		output_data[2]		3.3-V LVTTL (default)	Output
17		output_data[1]		3.3-V LVTTL (default)	Output
18		output_data[0]		3.3-V LVTTL (default)	Output
19		reset		3.3-V LVTTL (default)	Input

Named: | ◄► Edit: ✕ ✓ lvds_in   Filter: Pins: all

All Pins

If your design contains a large bus that exceeds the pins available in a particular I/O bank, you can use edge location assignments to place the bus. Edge location assignments improve the circuit board routing ability of large buses, because they are close together near an edge. The following shows Altera device package edges.

Figure 2-4: Die View and Package View of the Four Edges on an Altera Device

Silicon Die View — Top Edge, Left Edge, Right Edge, Bottom Edge

Package View (Top) — Top Edge, Left Edge, Right Edge, Bottom Edge

Overriding I/O Placement Rules on Differential Pins

Each device family has predefined I/O placement rules. The I/O placement rules ensure that noisy signals do not corrupt neighboring signals. For example, I/O placement rules define the allowed placement of single-ended I/O with respect to differential pins, or how many output and bidirectional pins can be placed within a VREF group when using voltage referenced input standards. You can use the IO_MAXIMUM_TOGGLE_RATE assignment to override I/O placement rules on pins, such as for system reset pins that do not switch during normal design activity. Setting a value of 0 MHz for this assignment causes the Fitter to recognize the pin at a DC state throughout device operation. The Fitter excludes the assigned pin from placement rule analysis. Do not assign an IO_MAXIMUM_TOGGLE_RATE of 0 MHz to any actively switching pin or your design may not function as intended.

Entering Pin Assignments with Tcl Commands

You can use Tcl scripts to apply pin assignments rather than using the GUI. Enter individual Tcl commands in the Tcl Console, or type the following to apply the assignments contained in a Tcl script:

Example 2-1: Applying Tcl Script Assignments

```
quartus_sh -t <my_tcl_script>.tcl
```

The following example shows use of the `set_location_assignment` and `set_instance_assignment` Tcl commands to assign a pin to a specific location, I/O standard, and drive strength.

Example 2-2: Scripted Pin Assignment

```
set_location_assignment PIN_M20 -to address[10]
set_instance_assignment -name IO_STANDARD "2.5 V" -to address[10]
set_instance_assignment -name
        CURRENT_STRENGTH_NEW "MAXIMUM CURRENT" -to address[10]
```

Related Information

- **Tcl Scripting** on page 5-1

Entering Pin Assignments in HDL Code

You can use synthesis attributes or low-level I/O primitives to embed I/O pin assignments directly in your HDL code. When you analyze and synthesize the HDL code, the information is converted into the appropriate I/O pin assignments. You can use either of the following methods to specify pin-related assignments with HDL code:

- Assigning synthesis attributes for signal names that are top-level pins
- Using low-level I/O primitives, such as ALT_BUF_IN, to specify input, output, and differential buffers, and for setting parameters or attributes

Using Low-Level I/O Primitives

You can alternatively enter I/O pin assignments using low-level I/O primitives. You can assign pin locations, I/O standards, drive strengths, slew rates, and on-chip termination (OCT) value assignments. You can also use low-level differential I/O primitives to define both positive and negative pins of a differential pair in the HDL code for your design.

Primitive-based assignments do not appear in the Pin Planner until after you perform a full compilation and back-annotate pin assignments (**Assignments > Back Annotate Assignments**).

Related Information
Designing with Low Level Primitives User Guide

Importing and Exporting I/O Pin Assignments

The Quartus Prime software supports transfer of I/O pin assignments across projects, or for analysis in third-party PCB tools. You can import or export I/O pin assignments in the following ways:

Table 2-3: Importing and Exporting I/O Pin Assignments

	Import Assignments	Export Assignments
Scenario	• From your PCB design tool or spreadsheet into Pin Planner during early pin planning or after optimization in PCB tool • From another Quartus Prime project with common constraints	• From Quartus Prime project for optimization in a PCB design tool • From Quartus Prime project for spreadsheet analysis or use in scripting assignments • From Quartus Prime project for import into another Quartus Prime project with similar constraints
Command	**Assignments > Import Assignments**	**Assignments > Export Assignments**
File formats	**.qsf**, **.es**f, **.acf**, **.csv**, **.txt**, **.sdc**	**.pin**, **.fx**, **.csv**, **.tcl**, **.qsf**
Notes	N/A	Exported **.csv** files retain column and row order and format. Do not modify the row of column headings if importing the **.csv** file

Importing and Exporting for PCB Tools

The Pin Planner supports import and export of assignments with PCB tools. You can export valid assignments as a **.pin** file for analysis in other supported PCB tools. You can also import optimized assignment from supported PCB tools. The **.pin** file contains pin name, number, and detailed properties.

Mentor Graphics I/O Designer requires you to generate and import both an **.fx** and a **.pin** file to transfer assignments. However, the Quartus Prime software requires only the **.fx** to import pin assignments from I/O Designer.

Table 2-4: Contents of .pin File

File Column Name	Description
Pin Name/Usage	The name of the design pin, or whether the pin is GND or V_{CC} pin
Location	The pin number of the location on the device package
Dir	The direction of the pin
I/O Standard	The name of the I/O standard to which the pin is configured
Voltage	The voltage level that is required to be connected to the pin
I/O Bank	The I/O bank to which the pin belongs
User Assignment	Y or N indicating if the location assignment for the design pin was user assigned (Y) or assigned by the Fitter (N)

Related Information

Pin-Out Files for Altera Devices

Mentor Graphics PCB Tools Support on page 16-1

Migrating Assignments to Another Target Device

You can migrate compatible pin assignments from one target device to another. You can migrate to a different density and the same device package. You can also migrate between device packages with different densities and pin counts. Click **View** > **Pin Migration Window** to verify whether your pin assignments are compatible with migration to a different Altera device.

The Quartus Prime software ignores invalid assignments and generates an error message during compilation. After evaluating migration compatibility, modify any incompatible assignments, and then click **Export** to export the assignments to another project.

Figure 2-5: Device Migration Compatibility (AC24 does not exist in migration device)

The migration result for the pin function of highlighted PIN_AC23 is not an NC but a voltage reference VREFB1N2 even though the pin is an NC in the migration device. VREF standards have a higher priority than an NC, thus the migration result display the voltage reference. Even if you do not use that pin for a port connection in your design, you must use the VREF standard for I/O standards that require it on the actual board for the migration device.

If one of the migration devices has pins intended for connection to V_{CC} or GND and these same pins are I/O pins on a different device in the migration path, the Quartus Prime software ensures these pins are not used for I/O. Ensure that these pins are connected to the correct PCB plane.

When migrating between two devices in the same package, pins that are not connected to the smaller die may be intended to connect to V_{CC} or GND on the larger die. To facilitate migration, you can connect these pins to V_{CC} or GND in your original design because the pins are not physically connected to the smaller die.

Related Information

AN90: SameFrame PinOut Design for FineLine BGA Packages

Send Feedback

Validating Pin Assignments

The Quartus Prime software validates I/O pin assignments against predefined I/O rules for your target device. You can use the following tools to validate your I/O pin assignments throughout the pin planning process:

Table 2-5: I/O Validation Tools

I/O Validation Tool	Description	Click to Run
I/O Assignment Analysis	Verifies I/O assignment legality of synthesized design against full set of I/O rules for the target device	**Processing > Start I/O Assignment Analysis**
Advanced I/O Timing	Fully validates I/O assignments against all I/O and timing checks during compilation	**Processing > Start Compilation**

I/O Assignment Validation Rules

I/O Assignment Analysis validates your assignments against the following rules:

Table 2-6: Examples of I/O Rule Checks

Rule	Description	HDL Required?
I/O bank capacity	Checks the number of pins assigned to an I/O bank against the number of pins allowed in the I/O bank.	No
I/O bank VCCIO voltage compatibility	Checks that no more than one VCCIO is required for the pins assigned to the I/O bank.	No
I/O bank VREF voltage compatibility	Checks that no more than one VREF is required for the pins assigned to the I/O bank.	No
I/O standard and location conflicts	Checks whether the pin location supports the assigned I/O standard.	No
I/O standard and signal direction conflicts	Checks whether the pin location supports the assigned I/O standard and direction. For example, certain I/O standards on a particular pin location can only support output pins.	No
Differential I/O standards cannot have open drain turned on	Checks that open drain is turned off for all pins with a differential I/O standard.	No

Rule	Description	HDL Required?
I/O standard and drive strength conflicts	Checks whether the drive strength assignments are within the specifications of the I/O standard.	No
Drive strength and location conflicts	Checks whether the pin location supports the assigned drive strength.	No
BUSHOLD and location conflicts	Checks whether the pin location supports BUSHOLD. For example, dedicated clock pins do not support BUSHOLD.	No
WEAK_PULLUP and location conflicts	Checks whether the pin location supports WEAK_PULLUP (for example, dedicated clock pins do not support WEAK_PULLUP).	No
Electromigration check	Checks whether combined drive strength of consecutive pads exceeds a certain limit. For example, the total current drive for 10 consecutive pads on a Stratix II device cannot exceed 200 mA.	No
PCI_IO clamp diode, location, and I/O standard conflicts	Checks whether the pin location along with the I/O standard assigned supports PCI_IO clamp diode.	No
SERDES and I/O pin location compatibility check	Checks that all pins connected to a SERDES in your design are assigned to dedicated SERDES pin locations.	Yes
PLL and I/O pin location compatibility check	Checks whether pins connected to a PLL are assigned to the dedicated PLL pin locations.	Yes

Table 2-7: Signal Switching Noise Rules

Rule	Description	HDL Required?
I/O bank can not have single-ended I/O when DPA exists	Checks that no single-ended I/O pin exists in the same I/O bank as a DPA.	No
A PLL I/O bank does not support both a single-ended I/O and a differential signal simultaneously	Checks that there are no single-ended I/O pins present in the PLL I/O Bank when a differential signal exists.	No
Single-ended output is required to be a certain distance away from a differential I/O pin	Checks whether single-ended output pins are a certain distance away from a differential I/O pin.	No

Rule	Description	HDL Required?
Single-ended output has to be a certain distance away from a VREF pad	Checks whether single-ended output pins are a certain distance away from a VREF pad.	No
Single-ended input is required to be a certain distance away from a differential I/O pin	Checks whether single-ended input pins are a certain distance away from a differential I/O pin.	No
Too many outputs or bidirectional pins in a VREFGROUP when a VREF is used	Checks that there are no more than a certain number of outputs or bidirectional pins in a VREFGROUP when a VREF is used.	No
Too many outputs in a VREFGROUP	Checks whether too many outputs are in a VREFGROUP.	No

Running I/O Assignment Analysis

I/O assignment analysis validates I/O assignments against the complete set of I/O system and board layout rules. Full I/O assignment analysis validates blocks that directly feed or are fed by resources such as a PLL, LVDS, or gigabit transceiver blocks. In addition, the checker validates the legality of proper VREF pin use, pin locations, and acceptable mixed I/O standards

Run I/O assignment analysis during early pin planning to validate initial reserved pin assignments before compilation. Once you define design files, run I/O assignment analysis to perform more thorough legality checks with respect to the synthesized netlist. Run I/O assignment analysis whenever you modify I/O assignments.

The Fitter assigns pins to accommodate your constraints. For example, if you assign an edge location to a group of LVDS pins, the Fitter assigns pin locations for each LVDS pin in the specified edge location and then performs legality checks. To display the Fitter-placed pins, click **Show Fitter Placements** in the Pin Planner. To accept these suggested pin locations, you must back-annotate your pin assignments.

View the I/O Assignment Warnings report to view and resolve all assignment warnings. For example, a warning that some design pins have undefined drive strength or slew rate. The Fitter recognizes undefined, single-ended output and bidirectional pins as non-calibrated OCT. To resolve the warning, assign the **Current Strength**, **Slew Rate** or **Slow Slew Rate** for the reported pin. Alternatively, you could assign the **Termination** to the pin. You cannot assign drive strength or slew rate settings when a pin has an OCT assignment.

Running Early I/O Assignment Analysis (without Design Files)

You can perform basic I/O legality checks before defining HDL design files. This technique produces a preliminary board layout. For example, you can specify a target device and enter pin assignments that correspond to PCB characteristics. You can reserve and assign an I/O standards to each pin, and then run I/O assignment analysis to ensure that there are no I/O standard conflicts in each I/O bank.

Figure 2-6: Assigning and Analyzing Pin-Outs without Design Files

```
                    ( Create a Quartus Prime Project )
                                  │
                                  ▼
                    ┌─────────────────────────────┐
                    │  Create Pin-Related Assignments │
                    │  (Stored in the .qsf file)      │
                    └─────────────────────────────┘
                                  │
                                  ▼
        ┌─────────────────────────────┐       ┌─────────────────────────────┐
        │  Start I/O Assignment Analysis │◄──────│  Modify and Correct Illegal   │
        └─────────────────────────────┘       │  Assignments Found in Report File │
                                  │             └─────────────────────────────┘
                                  ▼                           ▲
                            ╱─────────╲      No                │
                           ╱ Assignments ╲──────────────────────┘
                           ╲  Correct?   ╱
                            ╲─────────╱
                                  │ Yes
                                  ▼
                    ( **Pin Assignments Complete** )
```
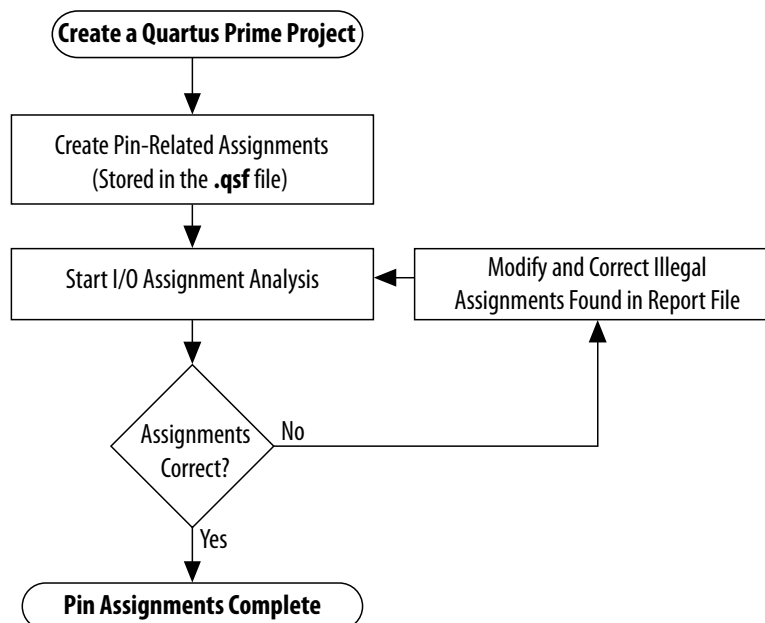
You must reserve all pins you intend to use as I/O pins, so that the Fitter can determine each pin type. After performing I/O assignment analysis, correct any errors reported by the Fitter and rerun I/O assignment analysis until all errors are corrected. A complete I/O assignment analysis requires all design files.

Running I/O Assignment Analysis (with Design Files)

Use I/O assignment analysis to perform full I/O legality checks after fully defining HDL design files. When you run I/O assignment analysis on a complete design, the tool verifies all I/O pin assignments against all I/O rules. When you run I/O assignment analysis on a partial designs, the tool checks legality only for defined portions of the design. The following figure shows the work flow for analyzing pin-outs with design files.

Figure 2-7: I/O Assignment Analysis Flow

```
                    ┌─────────────────────────────────────────┐
                    │      Quartus Prime Project & Design Files │
                    │  ┌───┐ ┌───┐ ┌───┐ ┌───┐ ┌───┐ ┌───┐ ┌───┐│
                    │  │.qpf│ │.edf│ │.vqm│ │ .v │ │.vhd│ │.bdf│ │.tdf││
                    │  └───┘ └───┘ └───┘ └───┘ └───┘ └───┘ └───┘│
                    └─────────────────────────────────────────┘
                                      │
                                      ▼
                  ( Open Quartus Prime Project or Design File )
                                      │
                                      ▼
                    ┌─────────────────────────────────────────┐
                    │       Create Pin-Related Assignments      │
                    │       (Stored in the .qsf file)           │
                    └─────────────────────────────────────────┘
                                      │
                                      ▼
                    ┌─────────────────────────────────────────┐
                    │       Perform Analysis & Synthesis        │
                    │       to Create a Mapped Netlist          │
                    └─────────────────────────────────────────┘
                                      │
                                      ▼
    ┌──────────────────────────┐   ┌──────────────────────────────────┐
    │  Start I/O Assignment    │◄──│   Modify and Correct Illegal      │
    │  Analysis                │   │   Assignments Found in Report File│
    └──────────────────────────┘   └──────────────────────────────────┘
                 │                                  ▲
                 ▼                                  │
            ◇ Assignments ◇ ──No──────────────────►│
            ◇  Correct?   ◇
                 │
                Yes
                 │
                 ▼
    ┌──────────────────────────┐
    │  Back-Annotate I/O Assignment │
    │  Analysis Pin Placements  │
    └──────────────────────────┘
                 │
                 ▼
    ( Pin-Related Assignments Complete )
```

Even if I/O assignment analysis passes on incomplete design files, you may still encounter errors during full compilation. For example, you can assign a clock to a user I/O pin instead of assigning it to a dedicated clock pin, or design the clock to drive a PLL that you have not yet instantiated in the design. This occurs because I/O assignment analysis does not account for the logic that the pin drives, and does not verify that only dedicated clock inputs can drive the a PLL clock port.

To obtain better coverage, analyze as much of the design as possible over time, especially logic that connects to pins. For example, if your design includes PLLs or LVDS blocks, define these files prior to full analysis. After performing I/O assignment analysis, correct any errors reported by the Fitter and rerun I/O assignment analysis until all errors are corrected.

The following figure shows the compilation time benefit of performing I/O assignment analysis before running a full compilation.

Figure 2-8: I/O Assignment Analysis Reduces Compilation Time
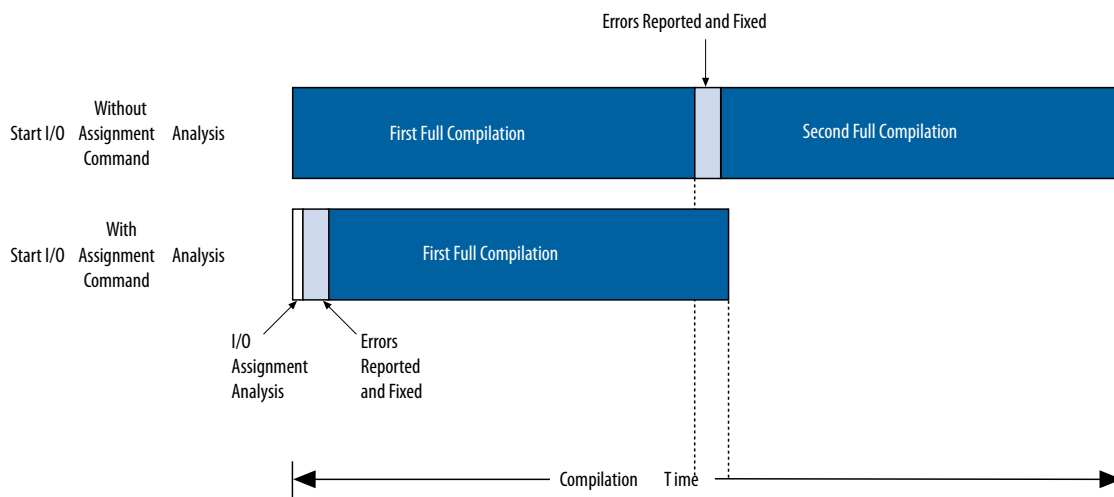
Overriding Default I/O Pin Analysis

You can override the default I/O analysis of various pins to accommodate I/O rule exceptions, such as for analyzing VREF or inactive pins.

Each device contains a number of VREF pins, each supporting a number of I/O pins. A VREF pin and its I/O pins comprise a VREF bank. The VREF pins are typically assigned inputs with VREF I/O standards, such as HSTL- and SSTL-type I/O standards. Conversely, VREF outputs do not require the VREF pin. When a voltage-referenced input is present in a VREF bank, only a certain number of outputs can be present in that VREF bank. I/O assignment analysis treats bidirectional signals controlled by different output enables as independent output enables.

To assign the **Output Enable Group** option to bidirectional signals to analyze the signals as a single output enable group, follow these steps:

1. To access this assignment in the Pin Planner, right-click the **All pins** list and click **Customize Columns**.
2. Under **Available columns**, add **Output Enable Group** to **Show these columns in this order**. The column appears in the **All Pins** list.
3. Enter the same integer value for the **Output Enable Group** assignment for all sets of signals that are driving in the same direction.

This assignment is especially important for external memory interfaces. For example, consider a DDR2 interface in a Stratix II device. The device allows 30 pins in a VREF group. Each byte lane for a ×8 DDR2 interface includes one DQS pin and eight DQ pins, for a total of nine pins per byte lane. The DDR2 interface uses the **SSTL 18 Class I** VREF I/O standard. In typical interfaces, each byte lane has its own output enable. In this example, the DDR2 interface has four byte lanes. Using 30 I/O pins in a VREF group, there are three byte lanes and an extra byte lane that supports the three remaining pins. Without the **Output Enable Group** assignment, the Fitter analyzes each byte lane as an independent group driven by a unique output enable. In this worst-case scenario the three pins are inputs, and the other 27 pins are outputs violating the 20 output pin limit.

Because DDR2 DQS and DQ pins are always driven in the same direction, the analysis reports an error that is not applicable to your design. The **Output Enable Group** assignment designates the DQS and DQ pins as a single group driven by a common output enable for I/O assignment analysis. When you use the **Output Enable Group** logic option assignment, the DQS and DQ pins are checked as all input pins or all output pins and are not in violation of the I/O rules.

You can also use the **Output Enable Group** assignment to designate pins that are driven only at certain times. For example, the data mask signal in DDR2 interfaces is an output signal, but it is driven only when the DDR2 is writing (bidirectional signals are outputs). To avoid I/O assignment analysis errors, use the **Output Enable Group** logic option assignment to assign the data mask to the same value as the DQ and DQS signals.

You can also use the **Output Enable Group** to designate VREF input pins that are inactive during the time the outputs are driving. This assignment removes the VREF input pins from the VREF analysis. For example, the QVLD signal for an RLDRAM II interface is active only during a read. During a write, the QVLD pin is not active and does not count as an active VREF input pin in the VREF group. Place the QVLD pins in the same output enable group as the RLDRAM II data pins.

Related Information
[TimeQuest Timing Analyzer](#)

Understanding I/O Analysis Reports

The detailed I/O assignment analysis reports include the affected pin name and a problem description. The Fitter section of the Compilation report contains information generated during I/O assignment analysis, including the following reports:

- I/O Assignment Warnings—lists warnings generated for each pin
- Resource Section—quantifies use of various pin types and I/O banks
- I/O Rules Section—lists summary, details, and matrix information about the I/O rules tested

The **Status** column indicates whether rules passed, failed, or could not be checked. A severity rating indicates the rule's importance for effective analysis. "Inapplicable" rules do not apply to the target device family.

Figure 2-9: I/O Rules Matrix

Verifying I/O Timing

You must verify board-level signal integrity and I/O timing when assigning I/O pins. High-speed interface operation requires a quality signal and low propagation delay at the far end of the board route. Click **Tools** > **TimeQuest Timing Analyzer** to confirm timing after making I/O pin assignments. For example, if you change the slew rates or drive strengths of some I/O pins with ECOs, you can verify timing without recompiling the design. You must understand I/O timing and what factors affect I/O timing paths in your design. The accuracy of the output load specification of the output and bidirectional pins affects the I/O timing results.

The Quartus Prime software supports three different methods of I/O timing analysis:

Table 2-8: I/O Timing Analysis Methods

I/O Timing Analysis	Description
Advanced I/O timing analysis	Analyze I/O timing with your board trace model to report accurate, "board-aware" simulation models. Configures a complete board trace model for each I/O standard or pin. TimeQuest applies simulation results of the I/O buffer, package, and board trace model to generate accurate I/O delays and system level signal information. Use this information to improve timing and signal integrity.
I/O timing analysis	Analyze I/O timing with default or specified capacitive load without signal integrity analysis. TimeQuest reports tCO to an I/O pin using a default or user-specified value for a capacitive load.

I/O Timing Analysis	Description
Full board routing simulation	Use Altera-provided or Quartus Prime software-generated IBIS or HSPICE I/O models for simulation in Mentor Graphics HyperLynx and Synopsys HSPICE.

For more information about advanced I/O timing support, refer to the appropriate device handbook for your target device. For more information about board-level signal integrity and tips on how to improve signal integrity in your high-speed designs, refer to the Altera Signal Integrity Center page of the Altera website.

For information about creating IBIS and HSPICE models with the Quartus Prime software and integrating those models into HyperLynx and HSPICE simulations, refer to the *Signal Integrity Analysis with Third Party Tools* chapter in volume 2 of the *Quartus Prime Handbook*.

Related Information

- **Literature and Technical Documentation**
- **Altera Signal Integrity Center**
- **Signal Integrity Analysis with Third-Party Tools** on page 14-1

Running Advanced I/O Timing

Advanced I/O timing analysis uses your board trace model and termination network specification to report accurate output buffer-to-pin timing estimates, FPGA pin and board trace signal integrity and delay values. Advanced I/O timing runs automatically for supported devices during compilation.

Understanding the Board Trace Models

The Quartus Prime software provides board trace model templates for various I/O standards. The following figure shows the template for a **2.5 V** I/O standard. This model consists of near-end and far-end board component parameters.

Near-end board trace modeling includes the elements which are close to the device. Far-end modeling includes the elements which are at the receiver end of the link, closer to the receiving device. Board trace model topology is conceptual and does not necessarily match the actual board trace for every component. For example, near-end model parameters can represent device-end discrete termination and breakout traces. Far-end modeling can represent the bulk of the board trace to discrete external memory components, and the far end termination network. You can analyze the same circuit with near-end modeling of the entire board, including memory component termination, and far-end modeling of the actual memory component.

Figure 2-10: 2.5-V I/O Standard Board Trace Model
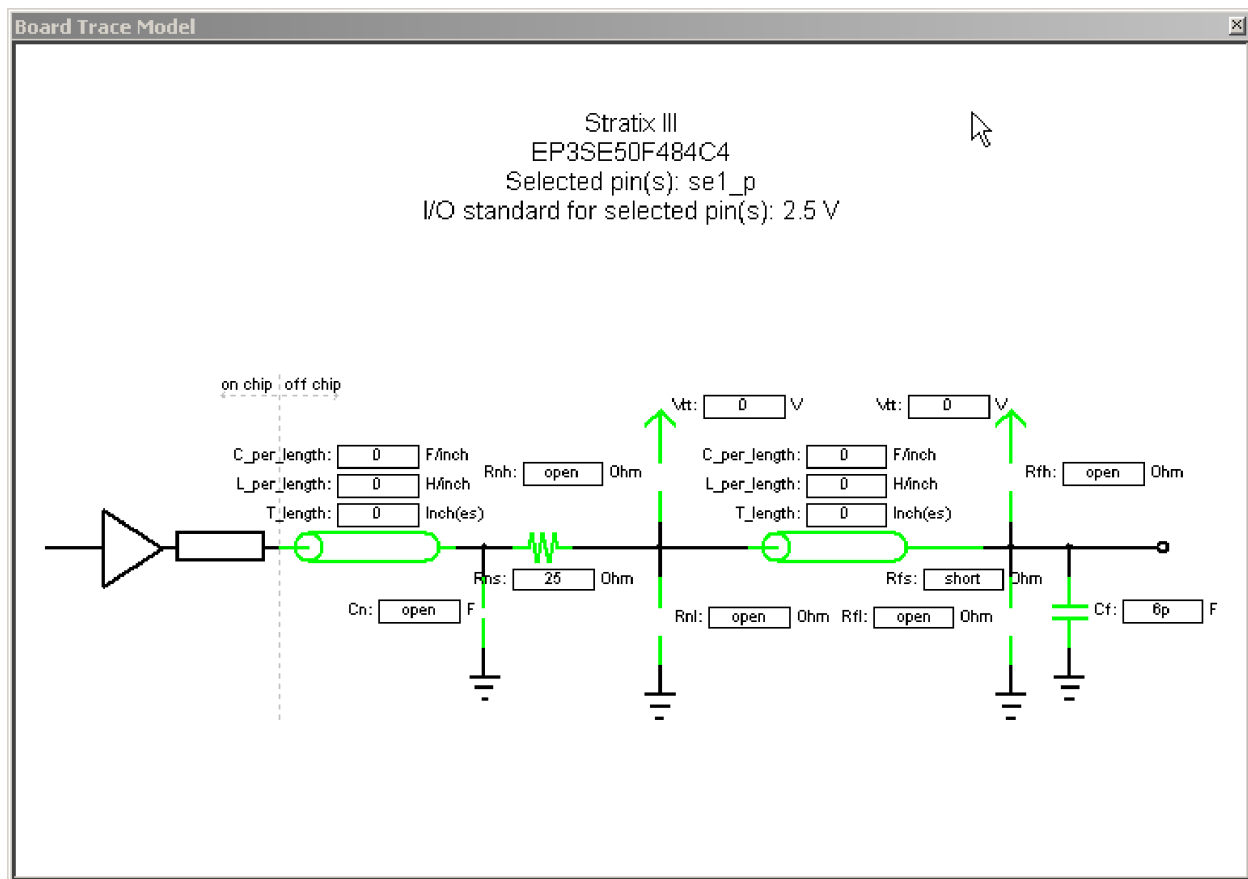
The following figure shows the template for the **LVDS** I/O standard. The far-end capacitance (Cf) represents the external-device or multiple-device capacitive load. If you have multiple devices on the far-end, you must find the equivalent capacitance at the far-end, taking into account all receiver capacitances. The far-end capacitance can be the sum of all the receiver capacitances.

The Quartus Prime software models lossless transmission lines, and does not require a transmission-line resistance value. Only distributed inductance (L) and capacitance (C) values are needed. The distributed L and C values of transmission lines must be entered on a per-inch basis, and can be obtained from the PCB vendor or manufacturer, the CAD Design tool, or a signal integrity tool, such as the Mentor Graphics Hyperlynx software.

Figure 2-11: LVDS Differential Board Trace Model
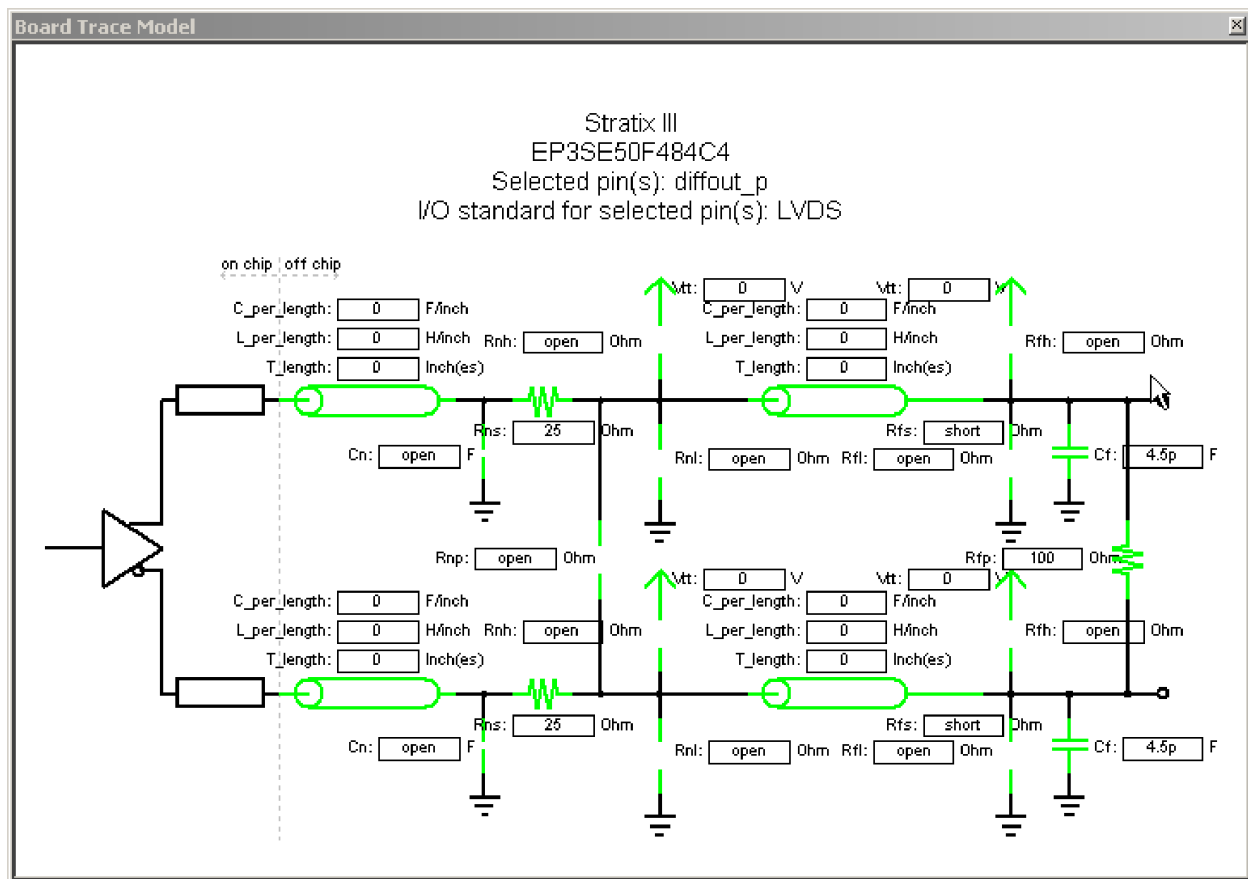
Defining the Board Trace Model

The board trace model describes a board trace and termination network as a set of capacitive, resistive, and inductive parameters. Advanced I/O Timing and the SSN Analyzer use the model to simulate the output signal from the output buffer to the far end of the board trace. You can define the capacitive load, any termination components, and trace impedances in the board routing for any output pin or bidirectional pin in output mode. You can configure an overall board trace model for each I/O standard or for specific pins. Define an overall board trace model for each I/O standard in your design. Use that model for all pins that use the I/O standard. You can customize the model for specific pins using the Board Trace Model window in the Pin Planner.

1. Click **Assignments > Device** and then click **Device and Pin Options**.
2. Click **Board Trace Model** and define board trace model values for each I/O standard.
3. Click **I/O Timing** and define default I/O timing options at board trace near and far ends.
4. Click **Assignments > Pin Planner** and assign board trace model values to individual pins.

Example 2-3: Specifying Board Trace Model

```
## setting the near end series resistance model of sel_p output pin to 25
ohms
```

```
set_instance_assignment -name BOARD_MODEL_NEAR_SERIES_R 25 -to sel_p
## Setting the far end capacitance model for sel_p output signal to 6
picofarads
set_instance_assignment -name BOARD_MODEL_FAR_C 6P -to sel_p
```

Related Information

Board Trace Model

For more information about configuring component values for a board trace model, including a complete list of the supported unit prefixes and setting the values with Tcl scripts, refer to Quartus Prime Help.

Modifying the Board Trace Model

To modify the board trace model, click **View** > **Board Trace Model** in the Pin Planner. You can modify any of the board trace model parameters within a graphical representation of the board trace model.

The Board Trace Model window displays the routing and components for positive and negative signals in a differential signal pair. Only modify the positive signal of the pair, as the setting automatically applies to the negative signal. Use standard unit prefixes such as *p*, *n*, and *k* to represent pico, nano, and kilo, respectively. Use the **short** or **open** value to designate a short or open circuit for a parallel component.

Specifying Near End vs Far End I/O Timing Analysis

You can select a nearend or far end point for I/O timing analysis. Near end timing analysis extends to the device pin. You can apply the `set_output_delay` constraint during near end analysis to account for the delay across the board.

Far end I/O timing analysis, then advanced I/O timing analysis extends to the external device input, at the far end of the board trace. Whether you choose a near end or far end timing endpoint, the board trace models are taken into account during timing analysis.

Understanding Advanced I/O Timing Analysis Reports

View I/O timing analysis information in the following reports:

Table 2-9: Advanced I/O Timing Reports

I/O Timing Report	Description
TimeQuest Report	Reports signal integrity and board delay data.
Board Trace Model Assignments report	Summarizes the board trace model component settings for each output and bidirectional signal.
Signal Integrity Metrics report	Contains all the signal integrity metrics calculated during advanced I/O timing analysis based on the board trace model settings for each output or bidirectional pin. Includes measurements at both the FPGA pin and at the far-end load of board delay, steady state voltages, and rise and fall times.

Note: By default, the TimeQuest analyzer generates the Slow-Corner Signal Integrity Metrics report. To generate a Fast-Corner Signal Integrity Metrics report you must change the delay model by clicking **Tools** > **TimeQuest Timing Analyzer**.

Related Information
The TimeQuest Timing Analyzer

Adjusting I/O Timing and Power with Capacitive Loading

When calculating t_{CO} and power for output and bidirectional pins, the TimeQuest analyzer and the PowerPlay Power Analyzer use a bulk capacitive load. You can adjust the value of the capacitive load per I/O standard to obtain more precise t_{CO} and power measurements, reflecting the behavior of the output or bidirectional net on your PCB. The Quartus Prime software ignores capacitive load settings on input pins. You can adjust the capacitive load settings per I/O standard, in picofarads (pF), for your entire design. During compilation, the Compiler measures power and t_{CO} measurements based on your settings. You can also adjust the capacitive load on an individual pin with the **Output Pin Load** logic option.

Viewing Routing and Timing Delays

Right-click any node and click **Locate** > **Locate in Chip Planner** to visualize and adjust I/O timing delays and routing between user I/O pads and V_{CC}, GND, and V_{REF} pads. The Chip Planner graphically displays logic placement, LogicLock® Plus regions, relative resource usage, detailed routing information, fan-in and fan-out, register paths, and high-speed transceiver channels. You can view physical timing estimates, routing congestion, and clock regions. Use the Chip Planner to change connections between resources and make post-compilation changes to logic cell and I/O atom placement. When you select items in the Pin Planner, the corresponding item is highlighted in Chip Planner.

Scripting API

You can alternatively use Tcl commands to access I/O management functions, rather than using the GUI. For detailed information about specific scripting command options and Tcl API packages, type the following command at a system command prompt to view the Tcl API Help browser:

```
quartus_sh --qhelp
```

Related Information

- **Tcl Scripting** on page 5-1
- **Command Line Scripting** on page 4-1

Generate Mapped Netlist

Enter the following in the Tcl console or in a Tcl script:

```
execute_module -tool map
```

The `execute_module` command is in the flow package.

Type the following at a system command prompt:

```
quartus_syn <project name>
```

Reserve Pins

Use the following Tcl command to reserve a pin:

```
set_instance_assignment -name RESERVE_PIN <value> -to <signal name>
```

Use one of the following valid reserved pin values:

- `"AS BIDIRECTIONAL"`
- `"AS INPUT TRI STATED"`
- `"AS OUTPUT DRIVING AN UNSPECIFIED SIGNAL"`
- `"AS OUTPUT DRIVING GROUND"`
- `"AS SIGNALPROBE OUTPUT"`

Note: You must include the quotation marks when specifying the reserved pin value.

Set Location

Use the following Tcl command to assign a signal to a pin or device location:

```
set_location_assignment <location> -to <signal name>
```

Valid locations are pin locations, I/O bank locations, or edge locations. Pin locations include pin names, such as `PIN_A3`. I/O bank locations include `IOBANK_1` up to `IOBANK_` *n*, where *n* is the number of I/O banks in the device.

Use one of the following valid edge location values:

- `EDGE_BOTTOM`
- `EDGE_LEFT`
- `EDGE_TOP`
- `EDGE_RIGHT`

Exclusive I/O Group

Use the following Tcl command to create an exclusive I/O group assignments:

```
set_instance_assignment -name "EXCLUSIVE_IO_GROUP" -to pin
```

Slew Rate and Current Strength

Use the following Tcl commands to create an slew rate and drive strength assignments:

```
set_instance_assignment -name CURRENT_STRENGTH_NEW 8MA -to e[0]
set_instance_assignment -name SLEW_RATE 2 -to e[0]
```

Related Information
Altera Device Package Information Data Sheet

Document Revision History

The following table shows the revision history for this chapter.

Table 2-10: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Removed early pin planning and Live I/O Check support from Quartus Prime Pro Edition handbook • Changed instances of *Quartus II* to *Quartus Prime*.
2014.12.15	14.1.0	• Updated Live I/O check device support to include only limited device families.
2014.08.30	14.0a10.0	• Added link to information about special pin assignment features for Arria 10 SoC devices.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2013	13.1.0	• Reorganization and conversion to DITA.
May 2013	13.0.0	• Added information about overriding I/O placement rules.
November 2012	12.1.0	• Updated Pin Planner description for new task and report windows.
June 2012	12.0.0	• Removed survey link.
November 2011	11.1.0	• Minor updates and corrections. • Updated the document template.
December 2010	10.0.1	Template update
July 2010	10.0.0	• Reorganized and edited the chapter • Added links to Help for procedural information previously included in the chapter • Added information on rules marked Inapplicable in the I/O Rules Matrix Report • Added information on assigning slew rate and drive strength settings to pins to fix I/O assignment warnings
November 2009	9.1.0	• Reorganized entire chapter to include links to Help for procedural information previously included in the chapter • Added documentation on near-end and far-end advanced I/O timing

Date	Version	Changes
March 2009	9.0.0	• Updated "Pad View Window" on page 5–20 • Added new figures: • Figure 5–15 • Figure 5–16 • Added new section "Viewing Simultaneous Switching Noise (SSN) Results" on page 5–17 • Added new section "Creating Exclusive I/O Group Assignments" on page 5–18

Related Information

Quartus Handbook Archive

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

Design planning—the feasibility analysis of physical constraints—is a fundamental early step in advanced FPGA design. The BluePrint Platform Designer helps you to accurately plan constraints for physical implementation. Use BluePrint to prototype interface implementations and rapidly define a legal device floorplan. BluePrint supports only Arria 10 devices.

BluePrint interacts dynamically with the Quartus Prime Fitter to accurately verify placement legality while you plan. You can evaluate different floorplans using interactive, real-time reporting to accurately plan the best implementation without iterative compilation. Fitter verification ensures the highest correlation between your BluePrint plan and actual implementation results. You can apply the BluePrint plan constraints to your project with high confidence in the final implementation.

Figure 3-1: BluePrint Platform Designer GUI

BluePrint Platform Designer Functions

- Determines legal design floorplans at any stage of the design process.
- Analyzes clock network scenarios and ensure that specific device regions are fed by high fan-out signals.
- Evaluates placement legality using the Fitter.
- Reports alternative placement to improve performance and resources.
- Supports GUI and Tcl command-line operation.

Related Information

Video Demo: Using BluePrint to Place DDR-3 and PCI Express Gen3

BluePrint Planning Overview

After synthesizing your design, BluePrint helps you to rapidly define a legal device floorplan. BluePrint design planning involves initializing BluePrint, resolving project assignments, placing periphery design elements, and applying the BluePrint plan to your Quartus Prime project.

Note: Implementation constraints that you define in BluePrint do not apply to your Quartus Prime project until you apply the generated BluePrint constraints to your project. Apply the BluePrint constraints to your project by running the generated Tcl script.

Figure 3-2: BluePrint Streamlines Legal Placement
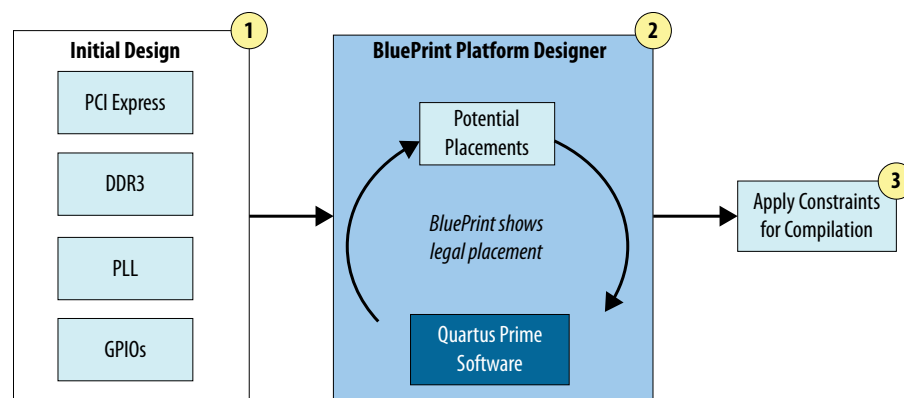
Altera FPGAs contain core and periphery device locations. The device core locations are adaptive look-up tables (ALUTs), core flip-flops, RAMs, and digital signal processors (DSPs). Device periphery locations include I/O elements, phase-locked loops (PLLs), clock buffers, and hard processor systems (HPS).

Altera's advanced FPGAs contain many silicon features in the periphery, such as hard PCI Express® IP cores, high speed transceivers, hard memory interface circuitry, and embedded processors. Interactions among these periphery atoms can be complex. BluePrint allows you to quickly plan I/O interface and periphery locations for interfaces such as:

- I/O elements
- LVDS interfaces
- PLLs
- Clocks

- Hard interface IP Cores
- High-Speed Transceivers
- Hard Memory Interface IP Cores
- Embedded Processors

After initialization, BluePrint displays your project's logical hierarchy, post-synthesis design elements, and Fitter-created design elements, alongside a view of target device locations. You can directly place these elements and instantly verify placement in the Fitter to ensure accurate correlation with the final implementation.

Figure 3-3: BluePrint Planning Flow

Using the BluePrint Interface

The BluePrint user interface helps you to quickly define a plan. You can directly place elements in the graphical floorplan, and verify placement to ensure accurate correlation with the final implementation. Place design elements onto the Chip or Package view of the target device to view real-time, accurate feedback about legal placement options. BluePrint validates your plan against any existing location assignments imported from the project.

Related Information

- **Managing Device I/O Pins** on page 2-1
- **Using the BluePrint Interface** on page 3-3

Figure 3-4: BluePrint GUI Overview
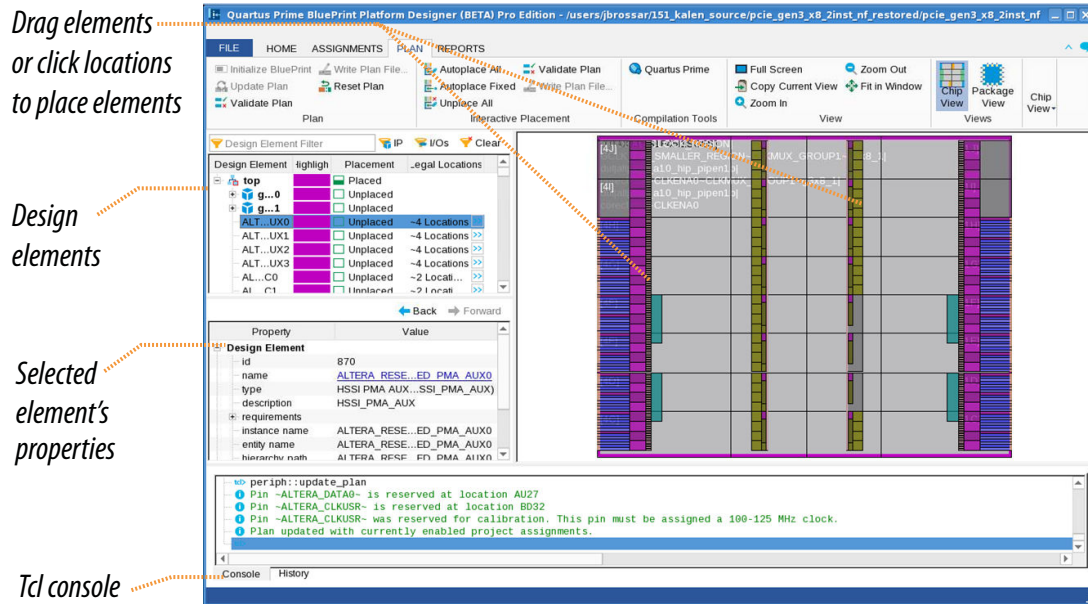
You can disable or enable imported project assignments to resolve any conflicts and evaluate different implementations. Once you are satisfied with the BluePrint plan for important interfaces, you validate the plan to confirm that the Fitter can place all remaining logic. Once validated, you can apply the BluePrint plan to your project via a generated Tcl file.

Figure 3-5: BluePrint Chip and Package Views
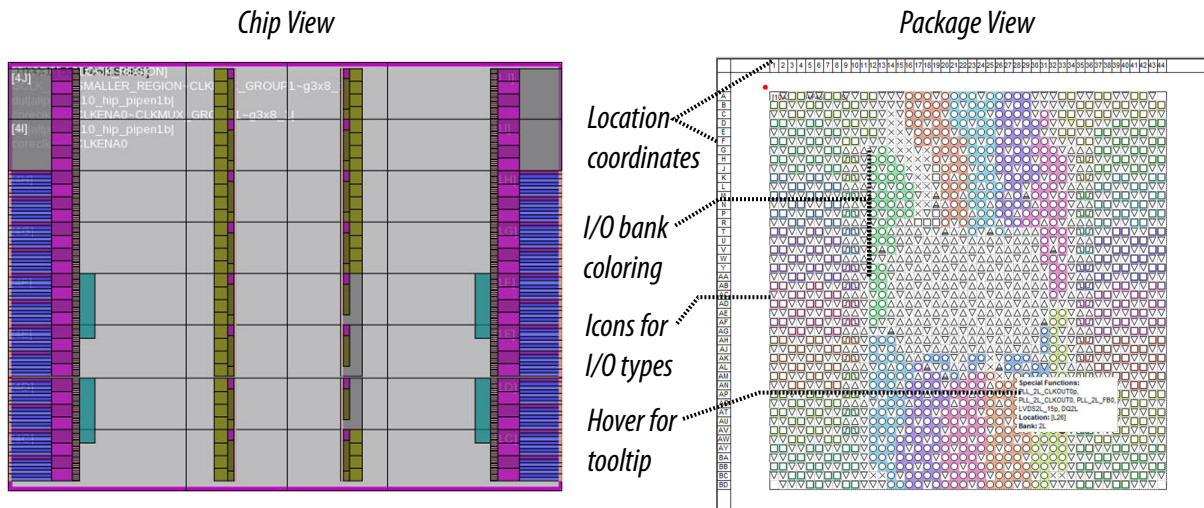
Table 3-1: BluePrint User Interface Reference

BluePrint Interface	Planning Task
File tab	Open or close a Quartus Prime project in BluePrint.

BluePrint Interface	Planning Task
Home tab	Initialize BluePrint legality engine and load project assignments.
Assignments tab	Resolve (disable) any conflicting assignments from your synthesized project. Click **Update Plan** to apply your enabled project assignments to the BluePrint plan.
Plan tab	Interactively plan placement with real-time validation and feedback in a graphical floorplan of the device.
Reports tab	Generate detailed, interactive reports to inform implementation decisions.
Tcl Console window	The BluePrint Tcl console displays the underlying Tcl commands executed by the BluePrint user interface. You can also run valid Tcl commands by typing them in the console directly or by sourcing them from a Tcl script.

Related Information

Step 1: Setup the Project

BluePrint requires at least a partially completed, synthesized Quartus Prime desogn project as input. BluePrint also supports planning with a fully complete design project. Before planning in BluePrint, create a Quartus Prime project and set up your initial design:

- Fully define known device periphery interfaces
- Instantiate all known interface IP cores
- Declare all general purpose I/Os
- Define the I/O standard, voltage, drive strength, and slew rate for all general purpose I/Os
- Define the core clocking (optional, but recommended)
- Connect all interfaces of the periphery IP to virtual pins, loopback interference, or test logic to ensure the interfaces are not removed by synthesis

Step 2: Initialize BluePrint

To open the current project in BluePrint, click **Tools** > **BluePrint Platform Designer**. The Quartus Prime software closes and BluePrint opens. Click **Open Project** on the **File** tab to open a different project. Click **Revisions** to open a specific project revision. Click **Home** to return to that tab.

After opening a project in BluePrint, click **Initialize BluePrint** on the **Home** tab to start the Fitter validation and import any project assignments. After initialization, the Fitter runs in the background to dynamically validate your BluePrint plan as you make changes. Changes made in BluePrint are not applied to your Quartus Prime project until you apply the BluePrint plan constraints to your project.

Figure 3-6: BluePrint Home Tab

Starts Fitter verification engine and imports assignments

Check shows complete steps
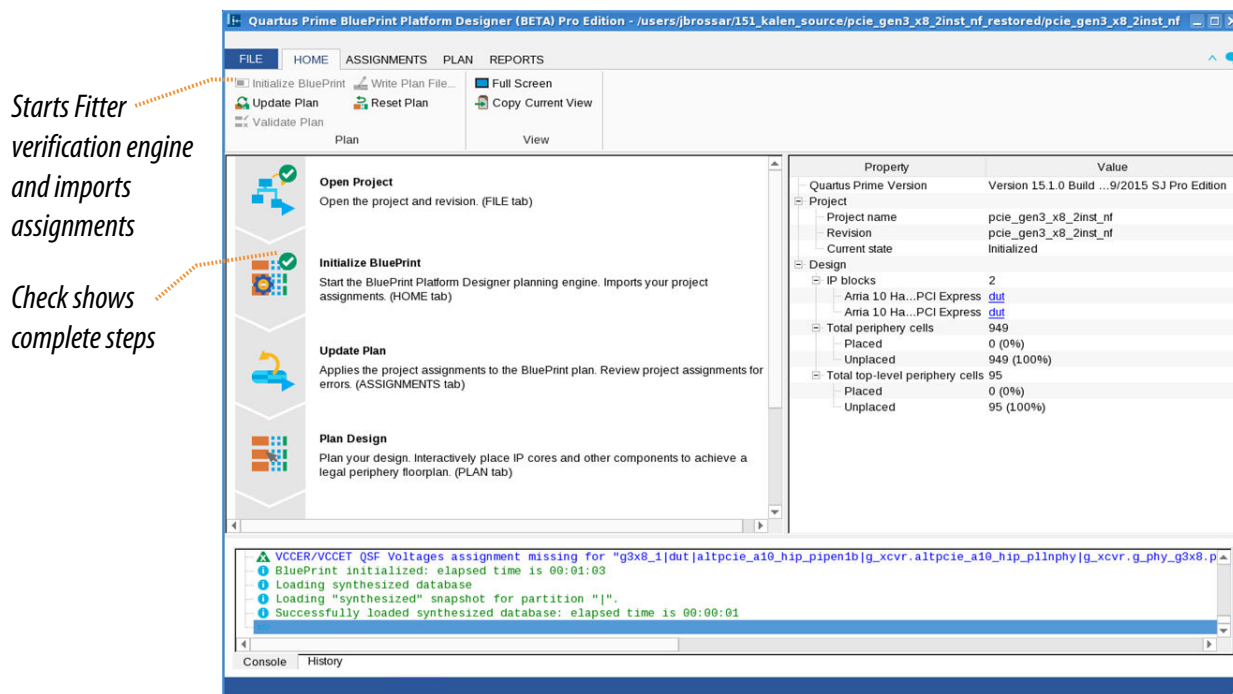
Home Tab Controls

The BluePrint **Home** tab contains the following controls.

Table 3-2: Home Tab Controls

Command	Description
Initialize BluePrint	Loads the synthesis netlist, starts the Fitter verification engine, and imports assignments from your Quartus Prime project.
Reset Plan	Unplaces all placed design elements and removes applied project assignments from the BluePrint plan. Resets all project assignments to the enabled state. You must subsequently run **Update Plan** prior to placing design elements. This command only applies to your BluePrint plan and does not impact your Quartus Prime project assignmnets until you apply the BluePrint script.

Step 3: Update Plan with Project Assignments

Before periphery planning you must update the BluePrint plan with the imported project assignments. You must resolve any conflicting project assignments before clicking **Update Plan** to apply the enabled project assignments to your BluePrint plan.

Click the **Assignments** tab to review and reconcile any conflicting assignments that BluePrint imports from your project. Enable or disable specific project assignments to resolve any conflicts. After resolving all conflicts, click **Update Plan** to apply the enabled project assignments to your BluePrint plan. You can click **Reset Plan** to reset all project assignments to the enabled state.

Figure 3-7: BluePrint Platform Designer (Assignments Tab)
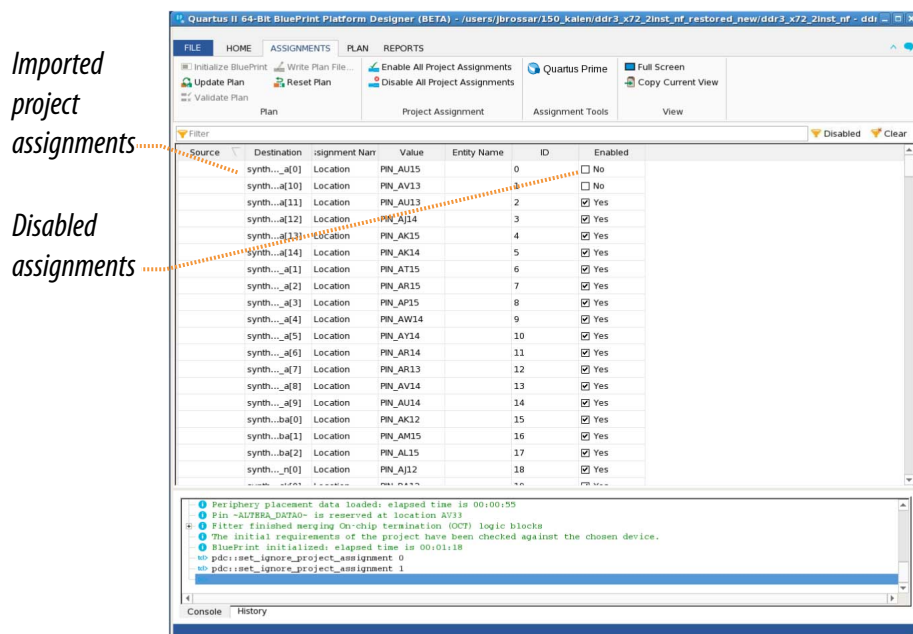
Imported project assignments

Disabled assignments

Assignments Tab Controls

The BluePrint **Assignments** tab contains the following controls for resolving any potential conflicts with imported project assignments. Enable or disable specific or classes of assignments until you resolve all potential conflicts. After you are satisfied with the status of all project assignments, click **Update Plan** to update your BluePrint plan with the enabled project assignments. BluePrint reports an error for any remaining assignment conflicts.

Table 3-3: Assignments Tab Controls

Command	Description
Filter field	Supports creation of wildcard expressions for assignment targets.
Enable All Project Assignments	Enables all imported project assignments for application to your BluePrint plan.
Disable All Project Assignments	Disables all imported project assignments in the plan.
Update Plan	Applies the enabled project assignments to your BluePrint plan. You cannot perform periphery planning on the **Plan** tab until you update the plan.
Enabled/Disabled	Filters the list to show only enabled or disabled assignments in the BluePrint plan.
Clear	Clears any filter from the **Assignments** list.
Copy Current View	Copies the current view to the clipboard.

Send Feedback

Command	Description
Quartus Prime	Closes BluePrint and opens the Quartus Prime software, allowing you to change and resynthesize your project assignments before planning with Blueprint. The Quartus Prime software automatically closes when you open BluePrint.

Step 4: Plan Periphery Placement

Click the **Plan** tab to interactively place IP cores and other design elements in legal locations in the device periphery. The **Plan** tab displays a list of your project's design elements, alongside a graphical abstraction of the target device architecture.

Figure 3-8: BluePrint Platform Designer (Plan Tab)
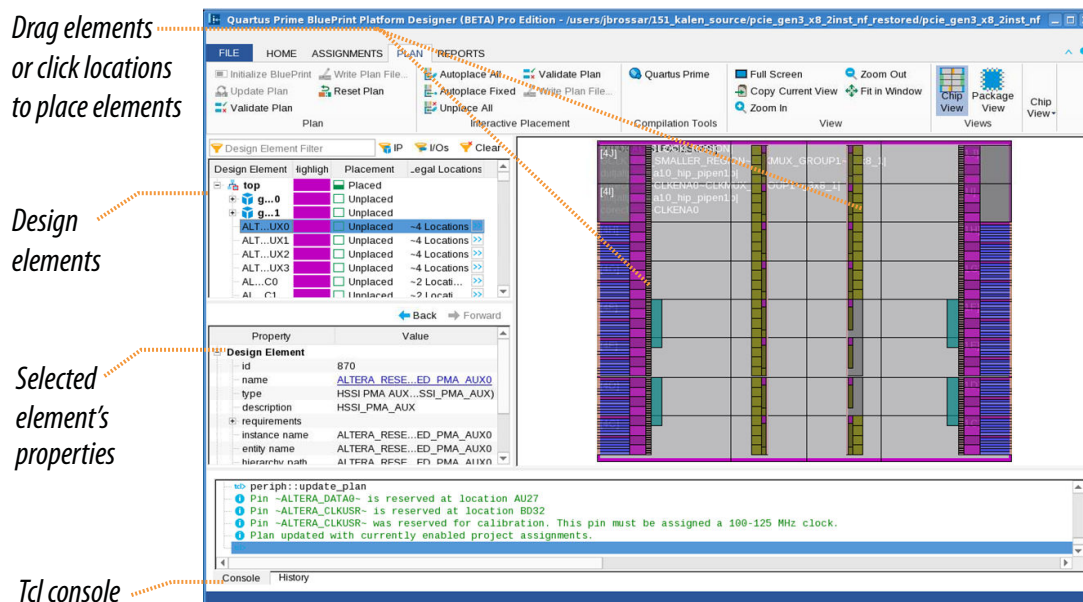
You can assign design elements by dragging them from the **Design Elements** list and dropping them onto available device resources in the **Chip** or **Package** view. Alternatively, click the ⟩⟩ button next to any design element to display a list of legal locations. Single-click any legal location in the list to highlight the location in the floorplan. Double-click any legal location in the list to place in the floorplan. BluePrint displays the approximate number of legal locations in the Legal Locations column. Clicking the ⟩⟩ button verifies the number of legal locations.

Figure 3-9: Listing Legal Locations
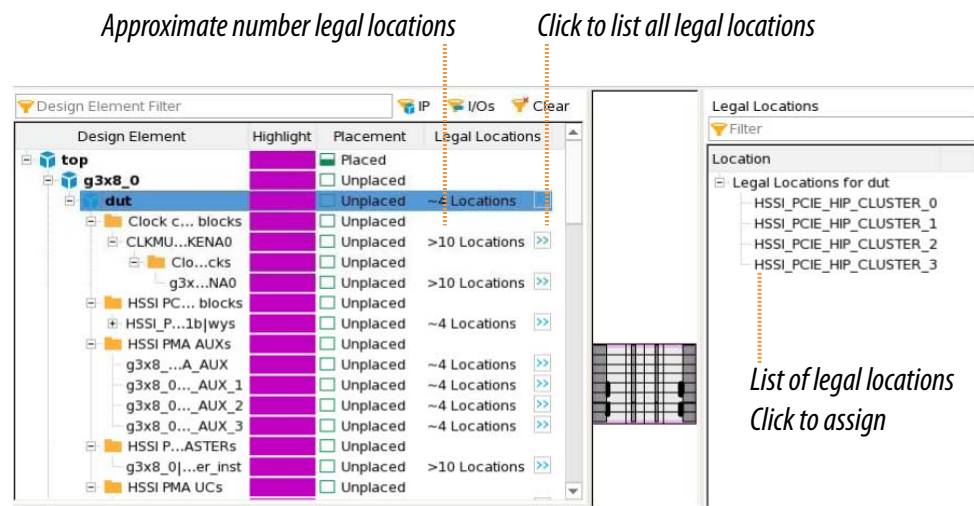
You can apply various filters to locate specific design elements or target device resources. Device resources are color coded by type, allowing quick access to any structure. BluePrint accurately evaluates the constraints using the Fitter in real time. Place elements in the following order for efficiency:

1. All I/O pins or elements, such as PLLs, with known specific location requirements
2. All known periphery interface IP
3. (Optional) Remaining unplaced cells (**Autoplace Fixed** or right-click any individual unplaced element to **Autoplace Selected Element**).

Select any physical design element and click **Report Placeability of Selected Element** to generate a report listing all legal placement options on the **Reports** tab. You can **Export** the contents of the report for analysis.

Note: Changes made in BluePrint do not apply to your Quartus Prime project until you apply the generated BluePrint plan constraints script to your project.

Plan Tab Controls

The BluePrint **Plan** tab contains the following controls to help you place logic in the BluePrint plan. Placement or unplacement in the BluePrint plan does not apply to your Quartus Prime project until you add the generated BluePrint constraints script to your project.

Table 3-4: Plan Tab Controls

Command	Description
>>	Click to list legal locations.
Autoplace All	Attempts to place all unplaced design elements in legal locations in the BluePrint plan.
Autoplace Fixed	Attempts to place all unplaced design elements that have only one legal location into the BluePrint plan.
Unplace All	Unplaces all placed design elements in the BluePrint plan.

Command	Description
Validate plan	Verifies that all constraints in the BluePrint plan are compatible with placement of all remaining unplaced design elements. You can then directly locate and resolve the source of any reported validation errors. You must successfully validate the plan before running **Write Plan File**.
Write Plan File	Saves your BluePrint plan to a file for subsequent application to your project. This command is available only after successfully running **Validate Plan**.
Right-click > Auto-place selected element	Attempts to place the selected design element and all of its children in a legal location in the BluePrint plan.
Chip View	Displays the target device chip. Zoom in to display chip details.
Package View	Displays the target device package. Zoom in to display chip details.
Right-click > Report Placeability of Selected Element	Displays detailed information on the **Reports** tab, showing legal locations in the BluePrint plan for the selected cell in order of suitability for fitting.
Copy Current View	Copies the current BluePrint plan to the clipboard for pasting into other files, such as word processing or presentation files.

Step 5: Report Placement Data

BluePrint provides detailed, actionable reporting to help you quickly implement the best plan for your design. Generate BluePrint placement and connectivity reports to help locate cells and make the best decisions about placement for the interfaces and elements in your design. You can access reporting commands from the **Reports** or **Plan** tab after you initialize (**Initialize BluePrint**) and apply enabled project assignments (**Update Plan**) in BluePrint.

Select any element in the **Plan** tab to **Report Placeability of Selected Element**, or **Report Legal Locations of Selected Element**. The generated reports appear on the BluePrint **Reports** tab. The BluePrint reports are dynamic, allowing you to select elements in the report and place, unplace, or report detailed data about the selected element(s) or location(s). This document describes each report in detail.

Figure 3-10: Reports Tab
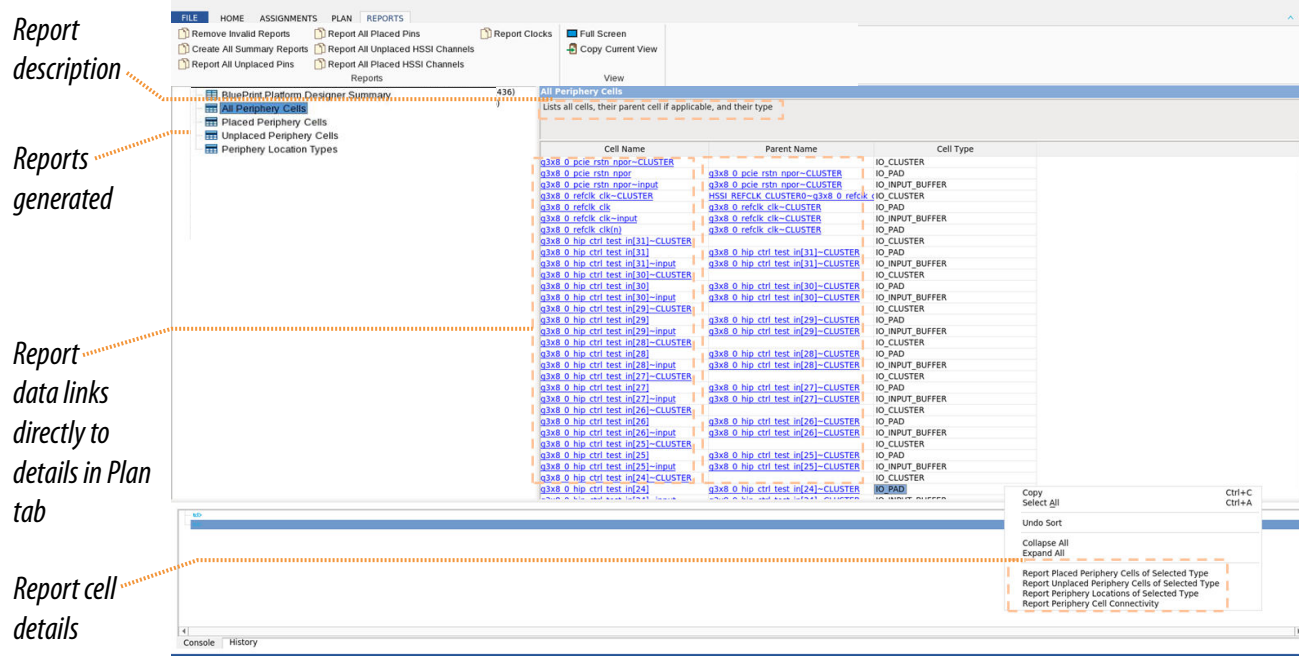
Step 6: Validate and Apply the Plan

You must validate your BluePrint plan before applying the BluePrint plan constraints to your project using a generated Tcl script. Validation confirms that the Fitter can place all remaining unplaced design elements. When you are satisfied with your BluePrint plan, follow these steps to validate and apply the BluePrint plan to your Quartus Prime project:

1. Click **Validate Plan** to confirm that the Fitter can place all remaining unplaced design elements.
2. After validation, click **Write Plan File** to generate a Tcl script that applies the plan to your project. The output Tcl file contains instructions to apply the BluePrint plan to your Quartus Prime project. Close BluePrint and open the Quartus Prime software.
3. Type the following from a command prompt to run the Tcl script and apply the BluePrint plan to your Quartus Prime project:

   ```
   quartus_sh -t <assignments_file>.tcl
   ```

   Alternatively, you can source the script from the Quartus Prime GUI.
4. In the Quartus Prime software, click **Start** > **Start Analysis & Synthesis** to synthesis and apply the BluePrint plan in your project.

BluePrint User Interface Controls

The BluePrint user interface includes the following controls for planning your design platform.

Home Tab Controls

The BluePrint **Home** tab contains the following controls.

Table 3-5: Home Tab Controls

Command	Description
Initialize BluePrint	Loads the synthesis netlist, starts the Fitter verification engine, and imports assignments from your Quartus Prime project.
Reset Plan	Unplaces all placed design elements and removes applied project assignments from the BluePrint plan. Resets all project assignments to the enabled state. You must subsequently run **Update Plan** prior to placing design elements. This command only applies to your BluePrint plan and does not impact your Quartus Prime project assignmnets until you apply the BluePrint script.

Assignments Tab Controls

The BluePrint **Assignments** tab contains the following controls for resolving any potential conflicts with imported project assignments. Enable or disable specific or classes of assignments until you resolve all potential conflicts. After you are satisfied with the status of all project assignments, click **Update Plan** to update your BluePrint plan with the enabled project assignments. BluePrint reports an error for any remaining assignment conflicts.

Table 3-6: Assignments Tab Controls

Command	Description
Filter field	Supports creation of wildcard expressions for assignment targets.
Enable All Project Assignments	Enables all imported project assignments for application to your BluePrint plan.
Disable All Project Assignments	Disables all imported project assignments in the plan.
Update Plan	Applies the enabled project assignments to your BluePrint plan. You cannot perform periphery planning on the **Plan** tab until you update the plan.
Enabled/Disabled	Filters the list to show only enabled or disabled assignments in the BluePrint plan.
Clear	Clears any filter from the **Assignments** list.
Copy Current View	Copies the current view to the clipboard.
Quartus Prime	Closes BluePrint and opens the Quartus Prime software, allowing you to change and resynthesize your project assignments before planning with Blueprint. The Quartus Prime software automatically closes when you open BluePrint.

Plan Tab Controls

The BluePrint **Plan** tab contains the following controls to help you place logic in the BluePrint plan. Placement or unplacement in the BluePrint plan does not apply to your Quartus Prime project until you add the generated BluePrint constraints script to your project.

Table 3-7: Plan Tab Controls

Command	Description
>>	Click to list legal locations.
Autoplace All	Attempts to place all unplaced design elements in legal locations in the BluePrint plan.
Autoplace Fixed	Attempts to place all unplaced design elements that have only one legal location into the BluePrint plan.
Unplace All	Unplaces all placed design elements in the BluePrint plan.
Validate plan	Verifies that all constraints in the BluePrint plan are compatible with placement of all remaining unplaced design elements. You can then directly locate and resolve the source of any reported validation errors. You must successfully validate the plan before running **Write Plan File**.
Write Plan File	Saves your BluePrint plan to a file for subsequent application to your project. This command is available only after successfully running **Validate Plan**.
Right-click > Auto-place selected element	Attempts to place the selected design element and all of its children in a legal location in the BluePrint plan.
Chip View	Displays the target device chip. Zoom in to display chip details.
Package View	Displays the target device package. Zoom in to display chip details.
Right-click > Report Placeability of Selected Element	Displays detailed information on the **Reports** tab, showing legal locations in the BluePrint plan for the selected cell in order of suitability for fitting.
Copy Current View	Copies the current BluePrint plan to the clipboard for pasting into other files, such as word processing or presentation files.

Reports Tab Controls

The BluePrint **Reports** tab contains the following controls to help you filter data and find entities and locations.

Table 3-8: Reports Tab Controls

Command	Description
Create all Summary Reports	Creates the following summary reports: • **BluePrint Platform Designer Summary** • **All Periphery Cells** • **Placed/Unplaced Periphery Cells** • **Periphery Location Types**.

Command	Description
Report All Placed/Unplaced Pins	Reports the name, parent (if any), and type of all placed (**Report All Placed Pins**) or unplaced (**Report All Unplaced Pins**) pins in the BluePrint plan, respectively. The Placed Pins report includes the placement location name. The Unplaced Pins report includes the number of potential placement locations. Right-click any cell to place, unplace, or report connectivity or location information.
Report All Placed/Unplaced Channels	Reports the name, parent (if any), and type of all placed (**Report All Placed HSSI Channels**) or unplaced (**Report All Unplaced HSSI Channels**) channels in the BluePrint plan, respectively. The Placed HSSI Channels report includes the placement location name. The Unplaced HSSI Channels report includes the number of potential placement locations. Right-click any cell to place, unplace, or report connectivity or location information.
Right-click > Report Placed/ Unplaced Periphery Cells of Selected Type	Reports the name, parent (if any), and type of placed (**Report Placed Periphery Cells of Selected Type**) or unplaced (**Report Unplaced Periphery Cells of Selected Type**) cells matching the selected type. The placed cells report includes the placement location name. The unplaced cells report includes the number of potential placement locations. Right-click any cell to place, unplace, or report connectivity or location information.
Right-click > Report Periphery Locations of Selected Type	Reports all locations in the device of the selected type, and whether the location supports merging.
Right-click > Report Periphery Cell Connectivity	Reports the source port and type, destination port and type, of connections to the selected cell. Right-click any cell to report the individual cell connectivity.
Right-click > Place/Unplace Cell	Places the cell in the selected location of the BluePrint plan. Similarly, you can right-click any cell and then click **Place Cell of Selected Type** or **Unplace Cell of Selected Type** to place or unplace multiple cells of the same type.
Right-click > Report Cell Locations for Custom Placement	Reports the preferred legal locations for the selected cell in the BluePrint plan in the Legal Location report. Right-click to immediately place the cell in a location or report all periphery location of the selected type.
Remove Invalid Reports	Removes outdated BluePrint reports that you invalidate by changes to the BluePrint plan.
Copy Current View	Copies the current report view to the clipboard for pasting into other applications.

BluePrint Reports

BluePrint provides detailed, actionable feedback to help you quickly implement the best plan for your design. You can access placement and further reporting functions directly from BluePrint reports. Use the reports to help locate cells and assign suitable placement locations for specific interfaces and elements in your design. BluePrint generates the following reports that provide detailed planning information.

Report Summary

Click the **Create all Summary Reports** on the **Reports** tab to generate summary reports about periphery cells in the BluePrint plan. Right-click any cell type to report placed, unplaced, connectivity, or location information.

Figure 3-11: Summary Reports

Summary reports

Click link to locate in Plan tab

Table 3-9: Report Summary

Command	Description
Create all Summary Reports	Creates the following summary reports: • **BluePrint Platform Designer Summary**—reports software version and total number of periphery and top-level periphery cells. • **All Periphery Cells**— reports the name, parent, and type of all periphery cells in the design. • **Placed/Unplaced Periphery Cells**—reports the name, parent, and type of all placed and unplaced periphery cells in the BluePrint plan. • **Periphery Location Types**—reports the number of each type of periphery location available in the target device and the number required by your design.

Report Pins

Click the following **Reports** tab commands to generate reports about I/O pins in the design. Right-click any cell type to place, unplace, or report connectivity or location information.

Table 3-10: Report Pin Commands

Command	Description
Report All Placed Pins	Generates the Placed Pins report. This report lists the name, parent, type, and location of all placed pins in the BluePrint plan.
Report All Unplaced Pins	Generates the Unplaced Pins report. This report lists the name, parent, type, and the number of potential placements for all unplaced pins in the BluePrint plan.

Figure 3-12: Placed Pins Report

Figure 3-13: Unplaced Pins Report

Report HSSI Channels

Click the following **Reports** tab commands to generate reports about HSSI channels in the BluePrint plan. Right-click any cell type to place, unplace, or report connectivity or location information.

Table 3-11: Report Channel Commands
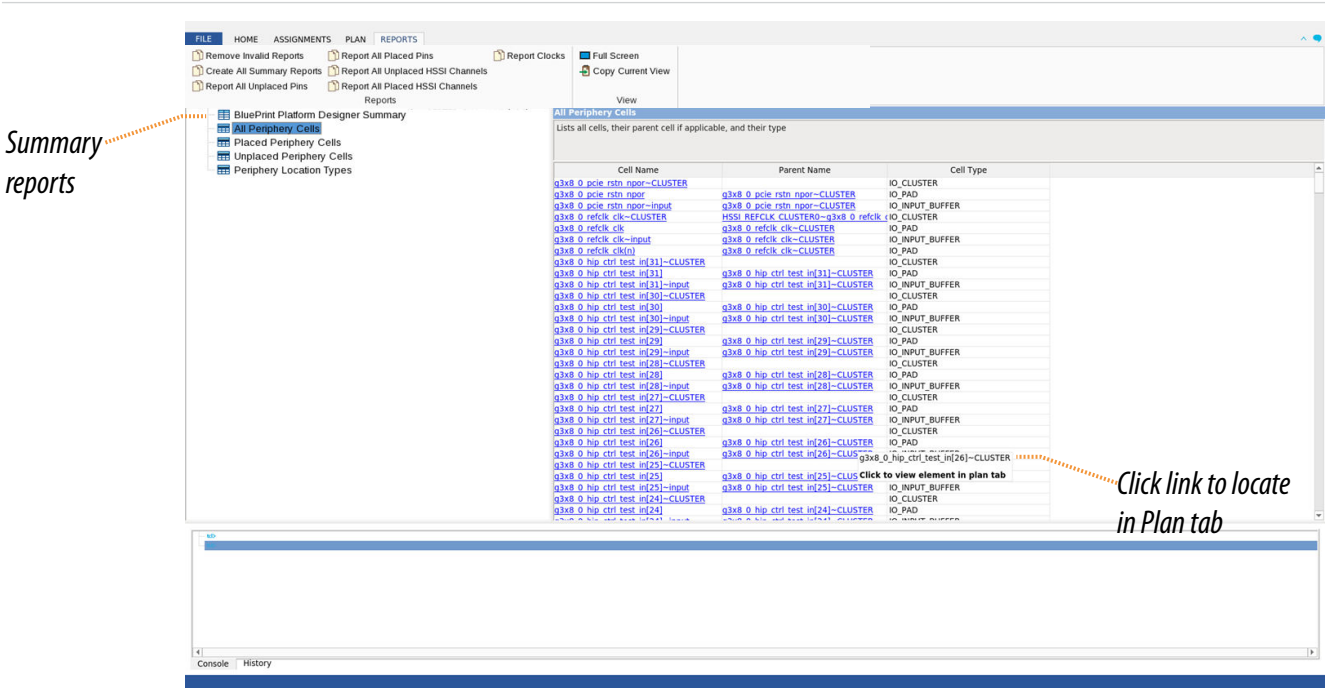
Command	Description
Report All Placed HSSI Channels	Generates the Placed HSSI Channels report. This report lists the name, parent, type, and location of all placed HSSI RX/TX channels in the BluePrint plan.
Report All Unplaced HSSI Channels	Generates the Unplaced HSSI Channels report. This report lists the name, parent, type, and location of all unplaced HSSI RX/TX channels in the BluePrint plan.

Figure 3-14: Unplaced HSSI Channels Report

Report Clocks

Click the following **Reports** tab commands to generate reports showing clock networks in the plan. Use this report to analyze clock network scenarios and ensure that specific device regions are fed by high fan-out signals.

Table 3-12: Report Clocks Commands

Command	Description
Report Clocks	Generates the Global and other Fast Signals report.

Figure 3-15: Global & Other Fast Signals Report

Global & Other Fast Signals

	Source	Source Location	Fan-Out	Signal Type	Global Buffer	Global Buffer Location	Clock Region	Clock Region Location
1	dcm_test_pin_clk[0]~pad	PIN_AR36	1	Global	dcm_test_pin_clk[0]~inputCLKENA0	CLKCTRL_X78_Y124_N85	GCLK_OR_SMALLER_REGI	Global Clock Region
2	dut_pll_ref_clk_clk~pad	PIN_F35	23	Global	dut_pll_ref_clk_clk~inputCLKENA0	CLKCTRL_X78_Y168_N76	GCLK_OR_SMALLER_REGI	Global Clock Region
3	emif_ed_ddr3_x8dut|archla	TILECTR...Y168_N2	1518	Global	emif_ed_ddr3_x8dut|arch|arch_ins!	CLKCTRL_X78_Y178_N82	GCLK_OR_SMALLER_REGI	Global Clock Region
4	emif_ed_ddr3_x8dut|archla	Core Location	2	Global	emif_ed_ddr3_x8dut|arch|arch_ins!	CLKCTRL_X78_Y168_N79	GCLK_OR_SMALLER_REGI	Global Clock Region
5	emif_ed_ddr3_x8dut|archla	IOPLL_X...Y168_N1	174	Global	emif_ed_ddr3_x8dut|arch|arch_ins!	CLKCTRL_X78_Y185_N76	GCLK_OR_SMALLER_REGI	Global Clock Region
6	iopll_normal_mode_inst|iopll	IOPLL_X...Y114_N1	1	Global	iopll_normal_mode_inst|iopll_inst~O	CLKCTRL_X78_Y124_N76	GCLK_OR_SMALLER_REGI	Global Clock Region
7	iopll_normal_mode_inst|outc	IOPLL_X...Y114_N1	1499	Global	iopll_normal_mode_inst|outclk[0]~CL	CLKCTRL_X78_Y124_N79	GCLK_OR_SMALLER_REGI	Global Clock Region
8	m20k_top_inst|rclk~pad	PIN_G18	36	Global	m20k_top_inst|rclk~inputCLKENA0	CLKCTRL_X142_Y178_N82	GCLK_OR_SMALLER_REGI	Global Clock Region
9	m20k_top_inst|wclk~pad	PIN_D23	4	Global	m20k_top_inst|wclk~inputCLKENA0	CLKCTRL_X142_Y195_N76	GCLK_OR_SMALLER_REGI	Global Clock Region
10	mac_clock~pad	PIN_C23	38	Global	mac_clock~inputCLKENA0	CLKCTRL_X142_Y195_N77	GCLK_OR_SMALLER_REGI	Global Clock Region
11	sys_reset_n~pad	PIN_P14	9	Global	sys_reset_n~inputCLKENA0	CLKCTRL_X142_Y114_N79	GCLK_OR_SMALLER_REGI	Global Clock Region
12	ts_mc_clk_in~pad	PIN_R14	130	Global	ts_mc_clk_in~inputCLKENA0	CLKCTRL_X142_Y124_N85	GCLK_OR_SMALLER_REGI	Global Clock Region
13	dcm_test_pin_clk[1]~pad	PIN_AH31	1	Regional	dcm_test_pin_clk[1]~inputCLKENA0	CLKCTRL_X78_Y141_N82	QCLK_OR_SMALLER_REGI	Regional Clock Region 2
14	dcm_test_pin_clk[2]~pad	PIN_C10	1	Periphery	dcm_test_pin_clk[2]~inputCLKENA0	CLKCTRL_X142_Y151_N94	PCLK_OR_SMALLER_REGI	Periphery Clock Region 10
15	top_inst|twentynm_channel	HSSIRXP...ACE_1G5	1	Periphery	top_inst|twentynm_channel_wrapp	CLKCTRL_X0_Y132_N130	GCLK_OR_SMALLER_REGI	Periphery Clock Region 3
16	top_inst|twentynm_channel	HSSITXP...ACE_1G5	1	Periphery	top_inst|twentynm_channel_wrapp	CLKCTRL_X0_Y132_N131	GCLK_OR_SMALLER_REGI	Periphery Clock Region 3

Report Legal Locations

In the **Plan** tab, right-click on a Design Element to generate reports that show legal locations for the element. In the **Reports** tab, select the Legal Locations report for the element. Right-click any Device Location Name to place the cell. Right-click any cell to immediately place the cell in the selected locations or report all periphery location of the selected type.

Table 3-13: Report Legal Locations Commands

Command	Description
Right-click > **Report Legal Locations of Selected Element**	Accessible in the **Plan** tab, this command generates the Legal Locations report that shows legal locations for the selected cell in order of suitability for fitting.
Right-click > **Report Cell Locations for Custom Placement**	Accessible in any of the "Unplaced" reports, this command generates the Legal Locations report that shows legal locations for your custom placement.

Figure 3-16: Legal Locations Report

Legal Locations - ALTERA_RESERVED_FITTER_INSERTED_PMA_AUX0(870)

Shows fitter preference of placement locations for cell
ALTERA_RESERVED_FITTER_INSERTED_PMA_AUX0(870)
Type: HSSI PMA AUX

	Device Location Name
1	HSSIPMAAUX_1CB
2	HSSIPMAAUX_1CT
3	HSSIPMAAUX_4CB
4	HSSIPMAAUX_4CT

Report Periphery Locations

Click the following **Reports** tab command to generate reports that show the status of periphery cells in the BluePrint plan.

Table 3-14: Report Periphery Locations Commands

Command	Description
Right-click > **Report Placed Periphery Cells of Selected Type**	Accessible from the **All Periphery Cells** report. This command reports the name, parent (if any), type, and location of the selected placed periphery cells matching the selected type. Right-click any cell to place, unplace, or report connectivity or location information.
Right-click > **Report Unplaced Periphery Cells of Selected Type**	Accessible from the **All Periphery Cells** report. This command reports the name, parent (if any), type, and number of suitable locations for the selected unplaced periphery cells matching the selected type. Right-click any cell to place, unplace, or report connectivity or location information.
Right-click > **Report Periphery Locations of Selected Type**	Reports all locations in the device of the selected type, and whether the location supports merging.

Figure 3-17: Periphery Locations Report

Report Cell Connectivity

Click the following **Reports** tab commands to generate reports showing the connections between all cells in the BluePrint plan.

Table 3-15: Report Cell Connectivity Command

Command	Description
Right-click > **Report Periphery Cell Connectivity**	Right-click any Cell Name in the reports to **Report Periphery Cell Connectivity**. The report lists the source and destination ports and type of connections to the selected cell. Right-click any cell to report all connections to the cell.

Figure 3-18: Cell Connectivity Report

Document Revision History

This document has the following revision history.

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*. • Integration of content into Quartus Prime Handbook. • Added descriptions of new dynamic reports. • Added Package View description. • Added GUI controls reference.
2015.05.04	15.0.0	Second beta release of document on Molson. Added information about the following subjects: • Overview information • Reset Plan command • Legal Assignments list and prompt • Tcl console
2014.12.15	14.1.	First beta release of document on Molson.

FPGA design software that easily integrates into your design flow saves time and improves productivity. The Altera Quartus Prime software provides you with a command-line executable for each step of the FPGA design flow to make the design process customizable and flexible.

The benefits provided by command-line executables include:

- Command-line control over each step of the design flow
- Reduced memory requirements
- Improved performance

  The command-line executables are also completely interchangable with the Quartus Prime GUI, allowing you to use the exact combination of tools that you prefer.

Benefits of Command-Line Executables

The Quartus Prime command-line executables provide control over each step of the design flow. Each executable includes options to control commonly used software settings. Each executable also provides detailed, built-in help describing its function, available options, and settings.

Command-line executables allow for easy integration with scripted design flows. You can easily create scripts with a series of commands. These scripts can be batch-processed, allowing for integration with distributed computing in server farms. These scripting capabilities enhance the ease of integration between the Quartus Prime software and other EDA synthesis, simulation, and verification software.

Command-line executables add flexibility without sacrificing the ease-of-use of the Quartus Prime GUI. You can use the Quartus Prime GUI and command-line executables at different stages in the design flow. For example, you might use the Quartus Prime GUI to edit the floorplan for the design, use the command-line executables to perform place-and-route, and return to the Quartus Prime GUI to perform debugging with the Chip Editor.

Command-line executables reduce the amount of memory required during each step in the design flow. Because each executable targets only one step in the design flow, the executables themselves are relatively compact, both in file size and the amount of memory used during processing. This memory usage reduction improves performance, and is particularly beneficial in design environments where heavy usage of computing resources results in reduced memory availability.

Related Information

Using the Quartus Prime Executables in Shell Scripts

Introductory Example

The following introduction to command-line executables demonstrates how to create a project, fit the design, and generate programming files.

The tutorial design included with the Quartus Prime software is used to demonstrate this functionality. If installed, the tutorial design is found in the <Quartus Prime directory>**/qdesigns/fir_filter** directory.

Before making changes, copy the tutorial directory and type the four commands shown in the introductory example below at a command prompt in the new project directory.

The <Quartus Prime directory>**/quartus/bin** directory must be in your PATH environment variable.

```
quartus_syn filtref --source=filtref.bdf --family="Arria 10"
quartus_fit filtref --part=<part> --pack_register=minimize_area
quartus_asm filtref
quartus_sta filtref
```

The `quartus_syn filtref --source=filtref.bdf --family="Arria 10"` command creates a new Quartus Prime project called **filtref** with **filtref.bdf** as the top-level file. It targets the Arria 10 device family and performs logic synthesis and technology mapping on the design files.

The `quartus_fit filtref --part=<part> --pack_--` `pack_register=minimize_arearegister=minimize_area` command performs fitting on the **filtref** project. This command specifies the device, and directs the Fitter to pack sequential and combinational functions into single logic cells to reduce device resource usage.

The `quartus_asm filtref` command creates programming files for the **filtref** project.

The `quartus_sta filtref` command performs basic timing analysis on the **filtref** project using the Quartus Prime TimeQuest Timing Analyzer, reporting worst-case setup slack, worst-case hold slack, and other measurements.

You can put the four commands from the introductory example into a batch file or script file, and run them. For example, you can create a simple UNIX shell script called **compile.sh**, which includes the code shown in the UNIX shell script example below.

```
#!/bin/sh
PROJECT=filtref
TOP_LEVEL_FILE=filtref.bdf
FAMILY="Arria 10"
PART=<part>
PACKING_OPTION=minimize_area
quartus_syn $PROJECT --source=$TOP_LEVEL_FILE --family=$FAMILY
quartus_fit $PROJECT --part=$PART --pack_register=$PACKING_OPTION
quartus_asm $PROJECT
quartus_sta $PROJECT
```

Edit the script as necessary and compile your project.

Related Information

TimeQuest Timing Analyzer Quick Start Tutorial
For more information about using all of the features of the quartus_sta executable. The TimeQuest Timing Analyzer employs Synopsys Design Constraints to fully analyze the timing of your design.

Command-Line Scripting Help

Help for command-line executables is available through different methods. You can access help built in to the executables with command-line options. You can use the Quartus Prime Command-Line and Tcl API Help browser for an easy graphical view of the help information.

To use the Quartus Prime Command-Line and Tcl API Help browser, type the following command:

```
quartus_sh --qhelp
```

This command starts the Quartus Prime Command-Line and Tcl API Help browser, a viewer for information about the Quartus Prime Command-Line executables and Tcl API.

Use the `-h` option with any of the Quartus Prime Command-Line executables to get a description and list of supported options. Use the `--help=<option name>` option for detailed information about each option.

Figure 4-1: Quartus Prime Command-Line and Tcl API Help Browser
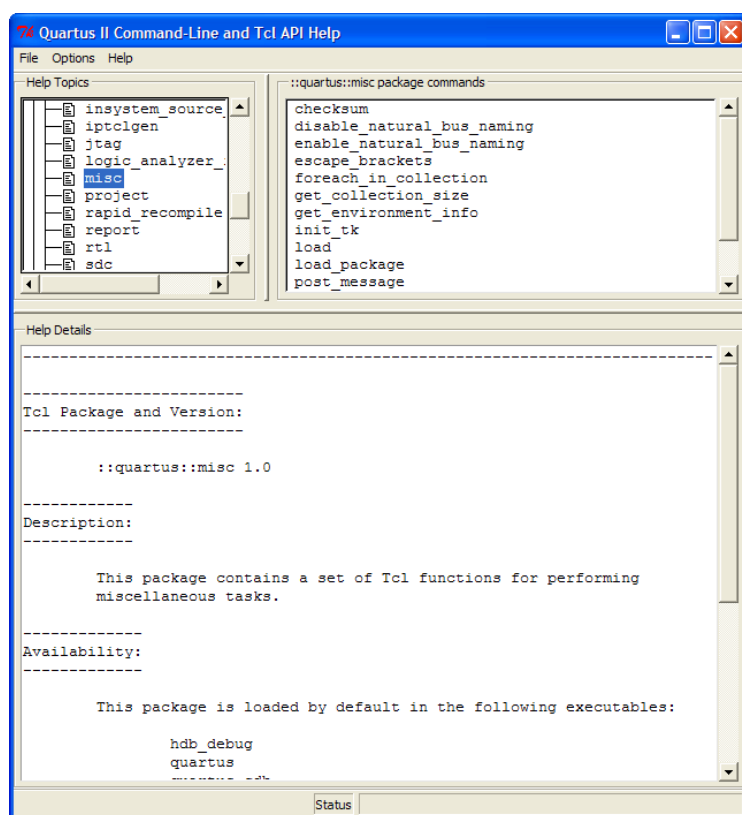
Project Settings with Command-Line Options

Command-line options are provided for many common global project settings and for performing common tasks.

You can use either of two methods to make assignments to an individual entity. If the project exists, open the project in the Quartus Prime GUI, change the assignment, and close the project. The changed assignment is updated in the **.qsf**. Any command-line executables that are run after this update use the

updated assignment. You can also make assignments using the Quartus Prime Tcl scripting API. If you want to completely script the creation of a Quartus Prime project, choose this method.

Related Information

- **Option Precedence** on page 4-4
- **Tcl Scripting** on page 5-1
- **QSF Reference Manual**

Option Precedence

If you use command-line executables, you must be aware of the precedence of various project assignments and how to control the precedence. Assignments for a particular project exist in the Quartus Prime Settings File (**.qsf**) for the project. Before the **.qsf** is updated after assignment changes, the updated assignments are reflected in compiler database files that hold intermediate compilation results.

All command-line options override any conflicting assignments found in the **.qsf** or the compiler database files. There are two command-line options to specify whether the **.qsf** or compiler database files take precedence for any assignments not specified as command-line options.

Any assignment not specified as a command-line option or found in the **.qsf** or compiler database file is set to its default value.

The file precedence command-line options are `--read_settings_files` and `--write_settings_files`.

By default, the `--read_settings_files` and `--write_settings_files` options are turned on. Turning on the `--read_settings_files` option causes a command-line executable to read assignments from the **.qsf** instead of from the compiler database files. Turning on the `--write_settings_files` option causes a command-line executable to update the **.qsf** to reflect any specified options, as happens when you close a project in the Quartus Prime GUI.

If you use command-line executables, be aware of the precedence of various project assignments and how to control the precedence. Assignments for a particular project can exist in three places:

- The **.qsf** for the project
- The result of the last compilation, in the **/db** directory, which reflects the assignments that existed when the project was compiled
- Command-line options

The precedence for reading assignments depends on the value of the `--read_settings_files` option.

Table 4-1: Precedence for Reading Assignments

Option Specified	Precedence for Reading Assignments
`--read_settings_files = on`	Command-line options
(Default)	The **.qsf** for the project
	Project database (db directory, if it exists)
	Quartus Prime software defaults

Option Specified	Precedence for Reading Assignments
`--read_settings_files = off`	Command-line options
	Project database (db directory, if it exists)
	Quartus Prime software defaults

The table lists the locations to which assignments are written, depending on the value of the `--write_settings_files` command-line option.

Table 4-2: Location for Writing Assignments

Option Specified	Location for Writing Assignments
`--write_settings_files = on` (Default)	**.qsf** and compiler database
`--write_settings_files = off`	Compiler database

The example assumes that a project named **fir_filter** exists, and that the analysis and synthesis step has been performed.

```
quartus_fit fir_filter --pack_register=off
quartus_sta fir_filter
mv fir_filter_sta.rpt fir_filter_1_sta.rpt
quartus_fit fir_filter --pack_register=minimize_area     --write_settings_files=off
quartus_sta fir_filter
mv fir_filter_sta.rpt fir_filter_2_sta.rpt
```

The first command, `quartus_fit fir_filter` --pack_register=off, runs the `quartus_fit` executable with no aggressive attempts to reduce device resource usage.

The second command, `quartus_sta fir_filter`, performs basic timing analysis for the results of the previous fit.

The third command uses the UNIX `mv` command to copy the report file output from **quartus_sta** to a file with a new name, so that the results are not overwritten by subsequent timing analysis.

The fourth command runs **quartus_fit** a second time, and directs it to attempt to pack logic into registers to reduce device resource usage. With the `--write_settings_files=off` option, the command-line executable does not update the **.qsf** to reflect the changed register packing setting. Instead, only the compiler database files reflect the changed setting. If the `--write_settings_files=off` option is not specified, the command-line executable updates the **.qsf** to reflect the register packing setting.

The fifth command reruns timing analysis, and the sixth command renames the report file, so that it is not overwritten by subsequent timing anlysis.

Use the options `--read_settings_files=off` and `--write_settings_files=off` (where appropriate) to optimize the way that the Quartus Prime software reads and updates settings files. In the following

example, the `quartus_asm` executable does not read or write settings files because doing so would not change any settings for the project.

```
quartus_syn filtref --source=filtref --part=<part>
quartus_fit filtref --pack_register=off --read_settings_files=off
quartus_asm filtref --read_settings_files=off --write_settings_files=off
```

Compilation with quartus_sh --flow

The figure shows a typical Quartus Prime FPGA design flow using command-line executables.

Figure 4-2: Typical Design Flow
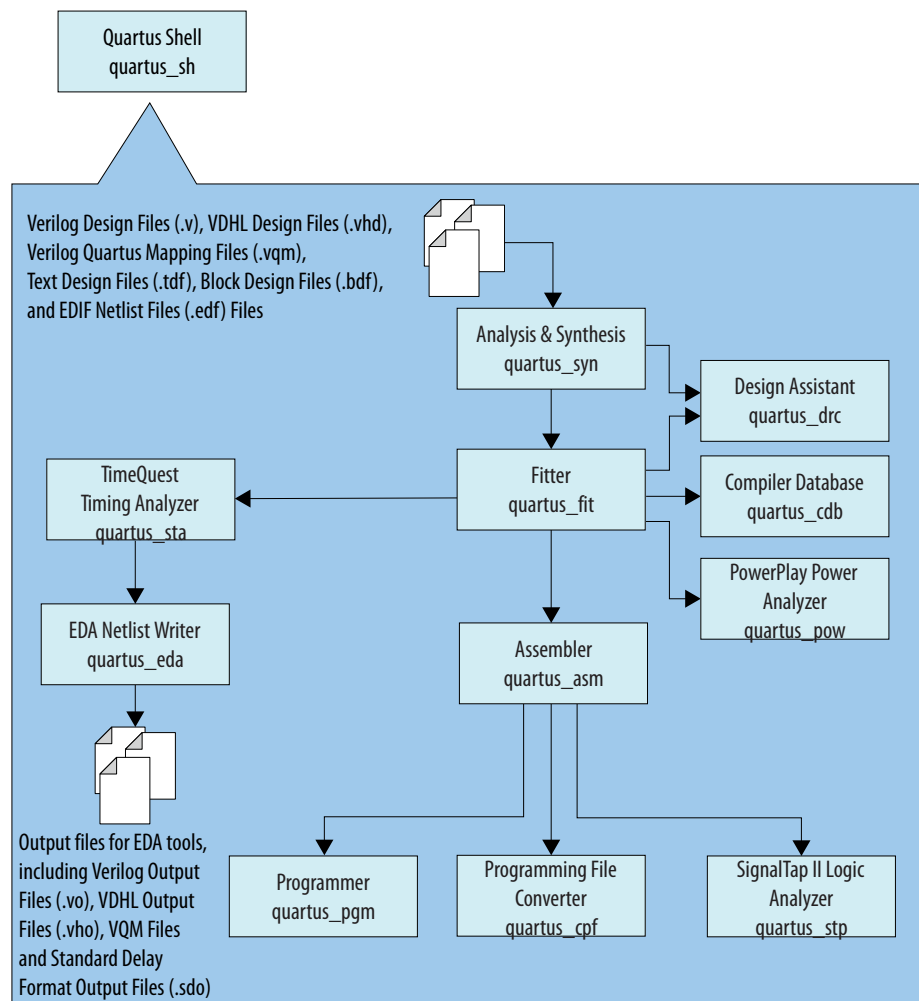
Use the `quartus_sh` executable with the `--flow` option to perform a complete compilation flow with a single command. The `--flow` option supports the smart recompile feature and efficiently sets command-line arguments for each executable in the flow.

The following example runs compilation, timing analysis, and programming file generation with a single command:

```
quartus_sh --flow compile filtref
```

Tip: For information about specialized flows, type `quartus_sh --help=flow` at a command prompt.

Text-Based Report Files

Each command-line executable creates a text report file when it is run. These files report success or failure, and contain information about the processing performed by the executable.

Report file names contain the revision name and the short-form name of the executable that generated the report file, in the format <revision>.<executable>**.rpt**. For example, using the **quartus_fit** executable to place and route a project with the revision name **design_top** generates a report file named **design_top.fit.rpt**. Similarly, using the **quartus_sta** executable to perform timing analysis on a project with the revision name **fir_filter** generates a report file named **fir_filter.sta.rpt**.

As an alternative to parsing text-based report files, you can use the **::quartus::report** Tcl package.

Related Information

- **Tcl Scripting** on page 5-1

Using Command-Line Executables In Scripts

You can use command-line executables in scripts that control other software in addition to the Quartus Prime software. For example, if your design flow uses third-party synthesis or simulation software, and if you can run the other software at a command prompt, you can include those commands with Quartus Prime executables in a single script.

The Quartus Prime command-line executables include options for common global project settings and operations, but you must use a Tcl script or the Quartus Prime GUI to set up a new project and apply individual constraints, such as pin location assignments and timing requirements. Command-line executables are very useful for working with existing projects, for making common global settings, and for performing common operations. For more flexibility in a flow, use a Tcl script, which makes it easier to pass data between different stages of the design flow and have more control during the flow.

For example, a UNIX shell script could run other synthesis software, then place-and-route the design in the Quartus Prime software, then generate output netlists for other simulation software. This script shows a script that synthesizes a design with the Synopsys Synplify software, simulates the design using the Mentor Graphics ModelSim® software, and then compiles the design targeting a Cyclone V device.

```
#!/bin/sh
# Run synthesis first.
# This example assumes you use Synplify software
synplify -batch synthesize.tcl
# If your Quartus Prime project exists already, you can just
# recompile the design.
# You can also use the script described in a later example to
# create a new project from scratch
quartus_sh --flow compile myproject
# Use the quartus_sta executable to do fast and slow-model
# timing analysis
```

```
quartus_sta myproject --model=slow
quartus_sta myproject --model=fast
# Use the quartus_eda executable to write out a gate-level
# Verilog simulation netlist for ModelSim
quartus_eda my_project --simulation --tool=modelsim --format=verilog
# Perform the simulation with the ModelSim software
vlib cycloneV_ver
vlog -work cycloneV_ver /opt/quartusii/eda/sim_lib/cycloneV_atoms.v
vlib work
vlog -work work my_project.vo
vsim -L cycloneV_ver -t 1ps work.my_project
```

Related Information

- **Tcl Scripting** on page 5-1

Common Scripting Examples

You can create scripts including command line executable to control common Quartus Prime processes.

Create a Project and Apply Constraints

The command-line executables include options for common global project settings and commands. To apply constraints such as pin locations and timing assignments, run a Tcl script with the constraints in it. You can write a Tcl constraint file yourself, or generate one for an existing project.

From the Project menu, click **Generate Tcl File for Project**.

The example creates a project with a Tcl script and applies project constraints using the tutorial design files in the Quartus Prime*<installation directory>*/**qdesigns/fir_filter/** directory.

```
project_new filtref -overwrite
# Assign family, device, and top-level file
set_global_assignment -name FAMILY Arria 10
set_global_assignment -name DEVICE <Device>
set_global_assignment -name BDF_FILE filtref.bdf
# Assign pins
set_location_assignment -to clk Pin_28
set_location_assignment -to clkx2 Pin_29
set_location_assignment -to d[0] Pin_139
set_location_assignment -to d[1] Pin_140
# Other assignments could follow
project_close
```

Save the script in a file called **setup_proj.tcl** and type the commands illustrated in the eample at a command prompt to create the design, apply constraints, compile the design, and perform fast-corner and slow-corner timing analysis. Timing analysis results are saved in two files, **filtref_sta_1.rpt** and **filtref_sta_2.rpt**.

```
quartus_sh -t setup_proj.tcl
quartus_map filtref
quartus_fit filtref
quartus_asm filtref
quartus_sta filtref --model=fast --export_settings=off
mv filtref_sta.rpt filtref_sta_1.rpt
quartus_sta filtref --export_settings=off
mv filtref_sta.rpt filtref_sta_2.rpt
```

Type the following commands to create the design, apply constraints, and compile the design, without performing timing analysis:

```
quartus_sh -t setup_proj.tcl
quartus_sh --flow compile filtref
```

The `quartus_sh --flow compile` command performs a full compilation, and is equivalent to clicking the **Start Compilation** button in the toolbar.

Check Design File Syntax

The UNIX shell script example shown in the example assumes that the Quartus Prime software **fir_filter** tutorial project exists in the current directory. You can find the **fir_filter** project in the <Quartus Prime directory>/**qdesigns/fir_filter** directory unless the Quartus Prime software tutorial files are not installed.

The `--analyze_file` option causes the **quartus_syn** executable to perform a syntax check on each file.

The script checks the exit code of the **quartus_map** executable to determine whether there is an error during the syntax check. Files with syntax errors are added to the `FILES_WITH_ERRORS` variable, and when all files are checked, the script prints a message indicating syntax errors.

When options are not specified, the executable uses the project database values. If not specified in the project database, the executable uses the Quartus Prime software default values. For example, the **fir_filter** project is set to target the Cyclone device family, so it is not necessary to specify the `--family` option.

```
#!/bin/sh
FILES_WITH_ERRORS=""
# Iterate over each file with a .bdf or .v extension
for filename in `ls *.bdf *.v`
do
# Perform a syntax check on the specified file
    quartus_syn fir_filter --analyze_file=$filename
  # If the exit code is non-zero, the file has a syntax error
    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
```

Create a Project and Synthesize a Netlist Using Netlist Optimizations

This example creates a new Quartus Prime project with a file **top.edf** as the top-level entity. The `--enable_register_retiming=on` and `--enable_wysiwyg_resynthesis=on` options cause **quartus_map** to optimize the design using gate-level register retiming and technology remapping.

The `--part` option causes **quartus_syn** to target a device. To create the project and synthesize it using the netlist optimizations described above, type the command shown in this example at a command prompt.

```
quartus_syn top --source=top.edf --enable_register_retiming=on
    --enable_wysiwyg_resynthesis=on --part=<part>
```

Archive and Restore Projects

You can archive or restore a Quartus Prime Archive File (**.qar**) with a single command. This makes it easy to take snapshots of projects when you use batch files or shell scripts for compilation and project management.

Use the `--archive` or `--restore` options for `quartus_sh` as appropriate. Type the command shown in the example at a command prompt to archive your project.

```
quartus_sh --archive <project name>
```

The archive file is automatically named *<project name>*.**qar.** If you want to use a different name, type the command with the `-output` option as shown in example the example.

```
quartus_sh --archive <project name> -output <filename>
```

To restore a project archive, type the command shown in the example at a command prompt.

```
quartus_sh --restore <archive name>
```

The command restores the project archive to the current directory and overwrites existing files.

Related Information
Managing Quartus Prime Projects

Update Memory Contents Without Recompiling

You can use two commands to update the contents of memory blocks in your design without recompiling. Use the `quartus_cdb` executable with the `--update_mif` option to update memory contents from **.mif** or **.hexout** files. Then, rerun the assembler with the `quartus_asm` executable to regenerate the **.sof**, **.pof**, and any other programming files.

```
quartus_cdb --update_mif <project name> [--rev=<revision name>]
quartus_asm <project name> [--rev=<revision name>]
```

The example shows the commands for a DOS batch file for this example. With a DOS batch file, you can specify the project name and the revision name once for both commands. To create the DOS batch file, paste the following lines into a file called **update_memory.bat.**

```
quartus_cdb --update_mif %1 --rev=%2
quartus_asm %1 --rev=%2
```

To run the batch file, type the following command at a command prompt:

```
update_memory.bat <project name> <revision name>
```

Create a Compressed Configuration File

You can create a compressed configuration file in two ways. The first way is to run `quartus_cpf` with an option file that turns on compression.

To create an option file that turns on compression, type the following command at a command prompt:

```
quartus_cpf -w <filename>.opt
```

This interactive command guides you through some questions, then creates an option file based on your answers. Use `--option` to cause **quartus_cpf** to use the option file. For example, the following command creates a compressed **.pof** that targets an EPCS64 device:

```
quartus_cpf --convert --option=<filename>.opt --device=EPCS64 <file>.sof <file>.pof
```

Alternatively, you can use the Convert Programming Files utility in the Quartus Prime software GUI to create a Conversion Setup File (**.cof**). Configure any options you want, including compression, then save the conversion setup. Use the following command to run the conversion setup you specified.

```
quartus_cpf --convert <file>.cof
```

Fit a Design as Quickly as Possible

This example assumes that a project called **top** exists in the current directory, and that the name of the top-level entity is **top**. The `--effort=fast` option causes the **quartus_fit** to use the fast fit algorithm to increase compilation speed, possibly at the expense of reduced f_{MAX} performance. The `--one_fit_attempt=on` option restricts the Fitter to only one fitting attempt for the design.

To attempt to fit the project called **top** as quickly as possible, type the command shown at a command prompt.

```
quartus_fit top --effort=fast --one_fit_attempt=on
```

Fit a Design Using Multiple Seeds

This shell script example assumes that the Quartus Prime software tutorial project called **fir_filter** exists in the current directory (defined in the file **fir_filter.qpf**). If the tutorial files are installed on your system, this project exists in the <Quartus Prime directory>/qdesigns<quartus_version_number> /**fir_filter** directory.

Because the top-level entity in the project does not have the same name as the project, you must specify the revision name for the top-level entity with the `--rev` option. The `--seed` option specifies the seeds to use for fitting.

A seed is a parameter that affects the random initial placement of the Quartus Prime Fitter. Varying the seed can result in better performance for some designs.

After each fitting attempt, the script creates new directories for the results of each fitting attempt and copies the complete project to the new directory so that the results are available for viewing and debugging after the script has completed.

```
#!/bin/sh
ERROR_SEEDS=""
quartus_map fir_filter --rev=filtref
# Iterate over a number of seeds
for seed in 1 2 3 4 5
do
echo "Starting fit with seed=$seed"
# Perform a fitting attempt with the specified seed
quartus_fit fir_filter --seed=$seed --rev=filtref
# If the exit-code is non-zero, the fitting attempt was
# successful, so copy the project to a new directory
if [ $? -eq 0 ]
then
        mkdir ../fir_filter-seed_$seed
        mkdir ../fir_filter-seed_$seed/db
        cp * ../fir_filter-seed_$seed
```

```
                cp db/* ../fir_filter-seed_$seed
else
                ERROR_SEEDS="$ERROR_SEEDS $seed"
fi
done
if [ -z "$ERROR_SEEDS" ]
then
echo "Seed sweeping was successful"
exit 0
else
echo "There were errors with the following seed(s)"
echo $ERROR_SEEDS
exit 1
fi


#!/bin/sh
ERROR_SEEDS=""
quartus_syn fir_filter --rev=filtref
# Iterate over a number of seeds
for seed in 1 2 3 4 5
do
echo "Starting fit with seed=$seed"
# Perform a fitting attempt with the specified seed
quartus_fit fir_filter --seed=$seed --rev=filtref
# If the exit-code is non-zero, the fitting attempt was
# successful, so copy the project to a new directory
if [ $? -eq 0 ]
then
                mkdir ../fir_filter-seed_$seed
                mkdir ../fir_filter-seed_$seed/db
                cp * ../fir_filter-seed_$seed
                cp db/* ../fir_filter-seed_$seed/db
else
                ERROR_SEEDS="$ERROR_SEEDS $seed"
fi
done
if [ -z "$ERROR_SEEDS" ]
then
echo "Seed sweeping was successful"
exit 0
else
echo "There were errors with the following seed(s)"
echo $ERROR_SEEDS
exit 1
fi
```

Tip: Use Design Space Explorer II (DSE) included with the Quartus Prime software script (by typing `quartus_dse` at a command prompt) to improve design performance by performing automated seed sweeping.

The QFlow Script

A Tcl/Tk-based graphical interface called QFlow is included with the command-line executables. You can use the QFlow interface to open projects, launch some of the command-line executables, view report files, and make some global project assignments.

The QFlow interface can run the following command-line executables:

- `quartus_syn` (Analysis and Synthesis)
- `quartus_fit` (Fitter)
- `quartus_sta` (TimeQuest timing analyzer)
- `quartus_asm` (Assembler)
- `quartus_eda` (EDA Netlist Writer)

To view floorplans or perform other GUI-intensive tasks, launch the Quartus Prime software.

Start QFlow by typing the following command at a command prompt:

```
quartus_sh -g
```

Tip: The QFlow script is located in the <Quartus Prime directory>/**common/tcl/apps/qflow/** directory.

Document Revision History

Table 4-3: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
2015.05.04	15.0.0	Remove descriptions of makefile support that was removed from software in 14.0.
December 2014	14.1.0	Updated DSE II commands.
June 2014	14.0.0	Updated formatting.
November 2013	13.1.0	Removed information about -silnet `qmegawiz` command
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	Corrected quartus_qpf example usage. Updated examples.
December 2010	10.1.0	Template update. Added section on using a script to regenerate megafunction variations. Removed references to the Classic Timing Analyzer (quartus_tan). Removed Qflow illustration.
July 2010	10.0.0	Updated script examples to use quartus_sta instead of quartus_tan, and other minor updates throughout document.

Date	Version	Changes
November 2009	9.1.0	Updated Table 2–1 to add quartus_jli and quartus_jbcc executables and descriptions, and other minor updates throughout document.
March 2009	9.0.0	No change to content.
November 2008	8.1.0	Added the following sections: • "The MegaWizard Plug-In Manager" on page 2–11 "Command-Line Support" on page 2–12 "Module and Wizard Names" on page 2–13 "Ports and Parameters" on page 2–14 "Invalid Configurations" on page 2–15 "Strategies to Determine Port and Parameter Values" on page 2–15 "Optional Files" on page 2–15 "Parameter File" on page 2–16 "Working Directory" on page 2–17 "Variation File Name" on page 2–17 • "Create a Compressed Configuration File" on page 2–21 • Updated "Option Precedence" on page 2–5 to clarify how to control precedence • Corrected Example 2–5 on page 2–8 • Changed Example 2–1, Example 2–2, Example 2–4, and Example 2–7 to use the EP1C12F256C6 device • Minor editorial updates • Updated entire chapter using 8½" × 11" chapter template
May 2008	8.0.0	• Updated "Referenced Documents" on page 2–20. • Updated references in document.

Related Information

Quartus Handbook Archive

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

Tcl Scripting

You can use Tcl scripts to control the Altera® Quartus Prime software and to perform a wide range of functions, such as compiling a design or scripting common tasks.

For example, use Tcl scripts to perform the following tasks:

- Manage a Quartus Prime project
- Make assignments
- Define design constraints
- Make device assignments
- Compile your design
- Perform timing analysis
- Access reports

Tcl scripts also facilitate project or assignment migration. For example, when designing in different projects with the same prototype or development board, you can write a script to automate reassignment of pin locations in each new project. The Quartus Prime software can also generate a Tcl script based on all the current assignments in the project, which aids in switching assignments to another project.

The Quartus Prime software Tcl commands follow the EDA industry Tcl application programming interface (API) standards for command-line options. This simplifies learning and using Tcl commands. If you encounter an error with a command argument, the Tcl interpreter includes help information showing correct usage.

This chapter includes sample Tcl scripts for automating tasks in the Quartus Prime software. You can modify these example scripts for use with your own designs. You can find more Tcl scripts in the Design Examples section of the Support area on the Altera website.

Related Information

Design Examples page on the Altera website

Tool Command Language

Tcl (pronounced "tickle") stands for Tool Command Language, a popular scripting language that is similar to many shell scripting and high-level programming languages. It provides support for control structures, variables, network socket access, and APIs.

Synopsys, Mentor Graphics®, and Altera software products support the Tcl industry-standard scripting language. Tcl allows you to create custom commands and works seamlessly across most development platforms.

You can create your own procedures by writing scripts combining basic Tcl commands and Quartus Prime API functions. You can then automate your design flow, run the Quartus Prime software in batch mode, or execute the individual Tcl commands interactively in the Quartus Prime Tcl interactive shell.

The Quartus Prime software supports Tcl/Tk version 8.5, supplied by the Tcl DeveloperXchange.

Related Information

- **External References** on page 5-22
  For a list of recommended literature on Tcl.
- **Tcl Scripting Basics** on page 5-17
  For more information on Tcl scripting, or if you are are a Tcl beginner.
- **tcl.activestate.com/**

Quartus Prime Tcl Packages

The Quartus Prime software groups Tcl commands into packages by function.

Table 5-1: Quartus Prime Tcl Packages

Package Name	Package Description
chip_planner	Identify and modify resource usage and routing with the Chip Editor
design	Manipulate project databases, including the assignments database, to enable the creation of instance assignments without modifying the **.qsf** file
device	Get device and family information from the device database
external_memif_ toolkit	Interact with external memory interfaces and debug components
fif	Contains the set of Tcl functions for using the Fault Injection File (FIF) Driver
flow	Compile a project, run command-line executables, and other common flows
insystem_ memory_edit	Read and edit memory contents in Altera devices
insystem_source_ probe	Interact with the In-System Sources and Probes tool in an Altera device
interactive_ synthesis	

Send Feedback

Package Name	Package Description
iptclgen	Generate memory IP
jtag	Control the JTAG chain
logic_analyzer_ interface	Query and modify the Logic Analyzer Interface output pin state
misc	Perform miscellaneous tasks such as enabling natural bus naming, package loading, and message posting
names	
periph	Interact with the BluePrint plans
project	Create and manage projects and revisions, make any project assignments including timing assignments
report	Get information from report tables, create custom reports
rtl	Traverse and query the RTL netlist of your design
rtm	
sdc	Specify constraints and exceptions to the TimeQuest Timing Analyzer
sdc_ext	Altera-specific SDC commands
simulator	Configure and perform simulations
sta	Contain the set of Tcl functions for obtaining advanced information from the TimeQuest Timing Analyzer
stp	Run the SignalTap® II Logic Analyzer
synthesis_report	Contain the set of Tcl functions for the Dynamic Synthesis Report tool
tdc	Obtain information from the TimeQuest Timing Analyzer

By default, only the minimum number of packages loads automatically with each Quartus Prime executable. This keeps the memory requirement for each executable as low as possible. Because the number of packages that the Quartus Prime executable loads is limited, you must load other packages before you can run commands in those packages.

Because different packages are available in different executables, you must run your scripts with executables that include the packages you use in the scripts. For example, if you use commands in the **sdc_ext** package, you must use the `quartus_sta` executable to run the script because the `quartus_sta` executable is the only one with support for the **sdc_ext** package.

The following command prints lists of the packages loaded or available to load for an executable, to the console:

`<executable name> --tcl_eval help`

For example, type the following command to list the packages loaded or available to load by the **quartus_fit** executable:

```
quartus_fit --tcl_eval help
```

Loading Packages

To load a Quartus Prime Tcl package, use the `load_package` command as follows:

```
load_package [-version <version number>] <package name>
```

This command is similar to the `package require` Tcl command, but you can easily alternate between different versions of a Quartus Prime Tcl package with the `load_package` command because of the `-version` option.

Related Information

- **Command-Line Scripting** on page 4-1
  For additional information about these and other Quartus Prime command-line executables.

Quartus Prime Tcl API Help

Access the Quartus Prime Tcl API Help reference by typing the following command at a system command prompt:

```
quartus_sh --qhelp
```

This command runs the Quartus Prime Command-Line and Tcl API help browser, which documents all commands and options in the Quartus Prime Tcl API.

Quartus Prime Tcl help allows easy access to information about the Quartus Prime Tcl commands. To access the help information, type `help` at a Tcl prompt.

Tcl Help Output

Table 5-2: Help Options Available in the Quartus Prime Tcl Environment

Help Command	Description
`help`	To view a list of available Quartus Prime Tcl packages, loaded and not loaded.
`help -tcl`	To view a list of commands used to load Tcl packages and access command-line help.
`help -pkg <package_name> [-version <version number>]`	To view help for a specified Quartus Prime package that includes the list of available Tcl commands. For convenience, you can omit the `::quartus::` package prefix, and type `help -pkg <package name>`. If you do not specify the `-version` option, help for the currently loaded package is displayed by default. If the package for which you want help is not loaded, help for the latest version of the package is displayed by default. Examples: `help -pkg ::quartus::project` `help -pkg project help -pkg project -version 1.0`

Help Command	Description

`<command_name> -h`
 	
or
 	
`<command_name> -help`
	To view short help for a Quartus Prime Tcl command for which the package is loaded.
	
	Examples:
	
	`project_open -h`
	
	`project_open -help`

`package require ::quartus::<package name> [<version>]`	To load a Quartus Prime Tcl package with the specified version. If *<version>* is not specified, the latest version of the package is loaded by default.
	
	Example:
	
	`package require ::quartus::project 1.0`
	
	This command is similar to the `load_package` command.
	
	The advantage of the `load_package` command is that you can alternate freely between different versions of the same package.
	
	Type `load_package <package name> [-version <version number>]` to load a Quartus Prime Tcl package with the specified version. If the `-version` option is not specified, the latest version of the package is loaded by default.
	
	Example:
	
	`load_package ::quartus::project -version 1.0`

`help -cmd <command_name> [-version <version>]`
 	
or
 	
`<command_name> -long_help`
	To view complete help text for a Quartus Prime Tcl command.
	
	If you do not specify the `-version` option, help for the command in the currently loaded package version is displayed by default.
	
	If the package version for which you want help is not loaded, help for the latest version of the package is displayed by default.
	
	Examples:
	
	`project_open -long_help`
	
	`help -cmd project_open`
	
	`help -cmd project_open -version 1.0`

`help -examples`	To view examples of Quartus Prime Tcl usage.
`help -quartus`	To view help on the predefined global Tcl array that contains project information and information about the Quartus Prime executable that is currently running.
`quartus_sh --qhelp`	To launch the Tk viewer for Quartus Prime command-line help and display help for the command-line executables and Tcl API packages.
`help -timequestinfo`	To view help on the predefined global "TimeQuestInfo" Tcl array that contains delay model information and speed grade information of a TimeQuest design that is currently running.

The Tcl API help is also available in Quartus Prime online help. Search for the command or package name to find details about that command or package.

Related Information

- **Command-Line Scripting** on page 4-1
  For more information about the Tk viewer for Quartus Prime command-line help.

Command-Line Options: -t, -s, and --tcl_eval

There are three command-line options you can use with executables that support Tcl.

Table 5-3: Command-Line Options Supporting Tcl Scripting

Command-Line Option	Description
`--script=<script file>` [`<script args>`]	Run the specified Tcl script with optional arguments.
`-t <script file>` [`<script args>`]	Run the specified Tcl script with optional arguments. The `-t` option is the short form of the `--script` option.
`--shell`	Open the executable in the interactive Tcl shell mode.
`-s`	Open the executable in the interactive Tcl shell mode. The `-s` option is the short form of the `--shell` option.
`--tcl_eval <tcl command>`	Evaluate the remaining command-line arguments as Tcl commands. For example, the following command displays help for the project package: `quartus_sh --tcl_eval help -pkg project`

Run a Tcl Script

Running an executable with the `-t` option runs the specified Tcl script. You can also specify arguments to the script. Access the arguments through the `argv` variable, or use a package such as **cmdline**, which supports arguments of the following form:

```
-<argument name> <argument value>
```

The **cmdline** package is included in the <Quartus Prime directory>/**common/tcl/packages** directory.

For example, to run a script called **myscript.tcl** with one argument, `Stratix`, type the following command at a system command prompt:

```
quartus_sh -t myscript.tcl Stratix
```

Related Information

Accessing Command-Line Arguments on page 5-14

Interactive Shell Mode

Running an executable with the `-s` option starts an interactive Tcl shell. For example, to open the Quartus Prime TimeQuest Timing Analyzer executable in interactive shell mode, type the following command:

```
quartus_sta -s
```

Commands you type in the Tcl shell are interpreted when you press Enter. You can run a Tcl script in the interactive shell with the following command:

```
source <script name>
```

If a command is not recognized by the shell, it is assumed to be an external command and executed with the `exec` command.

Evaluate as Tcl

Running an executable with the `--tcl_eval` option causes the executable to immediately evaluate the remaining command-line arguments as Tcl commands. This can be useful if you want to run simple Tcl commands from other scripting languages.

For example, the following command runs the Tcl command that prints out the commands available in the project package.

```
quartus_sh --tcl_eval help -pkg project
```

The Quartus Prime Tcl Console Window

You can run Tcl commands directly in the Quartus Prime Tcl Console window. On the View menu, click **Utility Windows.** By default, the Tcl Console window is docked in the bottom-right corner of the Quartus Prime GUI. All Tcl commands typed in the Tcl Console are interpreted by the Quartus Prime Tcl shell.

Note: Some shell commands such as `cd`, `ls`, and others can be run in the Tcl Console window, with the Tcl `exec` command. However, for best results, run shell commands and Quartus Prime executables from a system command prompt outside of the Quartus Prime software GUI.

Tcl messages appear in the **System** tab (Messages window). Errors and messages written to `stdout` and `stderr` also are shown in the Quartus Prime Tcl Console window.

End-to-End Design Flows

You can use Tcl scripts to control all aspects of the design flow, including controlling other software, when the other software also includes a scripting interface.

Typically, EDA tools include their own script interpreters that extend core language functionality with tool-specific commands. For example, the Quartus Prime Tcl interpreter supports all core Tcl commands, and adds numerous commands specific to the Quartus Prime software. You can include commands in one Tcl script to run another script, which allows you to combine or chain together scripts to control different tools. Because scripts for different tools must be executed with different Tcl interpreters, it is difficult to pass information between the scripts unless one script writes information into a file and another script reads it.

Within the Quartus Prime software, you can perform many different operations in a design flow (such as synthesis, fitting, and timing analysis) from a single script, making it easy to maintain global state information and pass data between the operations. However, there are some limitations on the operations you can perform in a single script due to the various packages supported by each executable.

There are no limitations on running flows from any executable. Flows include operations found in the Start section of the Processing menu in the Quartus Prime GUI, and are also documented as options for

the `execute_flow` Tcl command. If you can make settings in the Quartus Prime software and run a flow to get your desired result, you can make the same settings and run the same flow in a Tcl script.

Creating Projects and Making Assignments

You can create a script that makes all the assignments for an existing project, and then use the script at any time to restore your project settings to a known state. From the Project menu, click **Generate Tcl File for Project** to automatically generate a **.tcl** file containing your assignments. You can source this file to recreate your project, and you can add other commands to this file, such as commands for compiling the design. The file is a good starting point to learn about project management commands and assignment commands.

The following example creates a project, makes assignments, and compiles the project. The example uses the **fir_filter** tutorial design files in the **qdesigns** installation directory. Run this script in the **fir_filter** directory, with the `quartus_sh` executable.

Create and Compile a Project

```
load_package flow
# Create the project and overwrite any settings
# files that exist
project_new fir_filter -revision filtref -overwrite
# Set the device, the name of the top-level BDF,
# and the name of the top level entity
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C6F256C6
set_global_assignment -name BDF_FILE filtref.bdf
set_global_assignment -name TOP_LEVEL_ENTITY filtref
# Add other pin assignments here
set_location_assignment -to clk Pin_G1
# compile the project
execute_flow -compile
project_close
```

Note: The assignments created or modified while a project is open are not committed to the Quartus Prime Settings File (**.qsf**) unless you explicitly call `export_assignments` or `project_close` (unless `-dont_export_assignments` is specified). In some cases, such as when running `execute_flow`, the Quartus Prime software automatically commits the changes.

Related Information

- **Interactive Shell Mode** on page 5-6
- **Constraining Designs** on page 1-1
  For more information on making assignments.
- **QSF Reference Manual**
  For more information on scripting for all Quartus Prime project settings and assignments.

Compiling Designs

You can run the Quartus Prime command-line executables from Tcl scripts. Use the included **flow** package to run various Quartus Prime compilation flows, or run each executable directly.

The flow Package

The **flow** package includes two commands for running Quartus Prime command-line executables, either individually or together in standard compilation sequence. The `execute_module` command allows you to run an individual Quartus Prime command-line executable. The `execute_flow` command allows you to run some or all of the executables in commonly-used combinations. Use the **flow** package instead of system calls to run Quartus Prime executables from scripts or from the Quartus Prime Tcl Console.

Compile All Revisions

You can use a simple Tcl script to compile all revisions in your project. Save the following script in a file called **compile_revisions.tcl** and type the following to run it:

```
quartus_sh -t compile_revisions.tcl <project name>
```

Compile All Revisions

```
load_package flow
project_open [lindex $quartus(args) 0]
set original_revision [get_current_revision]
foreach revision [get_project_revisions] {
    set_current_revision $revision
    execute flow -compile
}
set_current_revision $original_revision
project_close
```

Reporting

You can extract information from the Compilation Report to evaluate results. The Quartus Prime Tcl API provides easy access to report data so you do not have to write scripts to parse the text report files.

If you know the exact report cell or cells you want to access, use the `get_report_panel_data` command and specify the row and column names (or x and y coordinates) and the name of the appropriate report panel. You can often search for data in a report panel. To do this, use a loop that reads the report one row at a time with the `get_report_panel_row` command.

Column headings in report panels are in row 0. If you use a loop that reads the report one row at a time, you can start with row 1 to skip row 0 with column headings. The `get_number_of_rows` command returns the number of rows in the report panel, including the column heading row. Because the number of rows includes the column heading row, continue your loop as long as the loop index is less than the number of rows.

Report panels are hierarchically arranged and each level of hierarchy is denoted by the string "||" in the panel name. For example, the name of the Fitter Settings report panel is `Fitter||Fitter Settings` because it is in the **Fitter** folder. Panels at the highest hierarchy level do not use the "||" string. For example, the Flow Settings report panel is named `Flow Settings`.

The following Tcl code prints a list of all report panel names in your project. You can run this code with any executable that includes support for the report package.

Print All Report Panel Names

```
load_package report
project_open myproject
```

```
load_report
set panel_names [get_report_panel_names]
foreach panel_name $panel_names {
post_message "$panel_name"
}
```

Viewing Report Data in Excel

The Microsoft Excel software can be useful in viewing and manipulating timing analysis results. You can create a Comma Separated Value (**.csv**) file from any Quartus Prime report to open with Excel. The following Tcl code shows a simple way to create a **.csv** file with data from the Fitter panel in a report. You could modify the script to use command-line arguments to pass in the name of the project, report panel, and output file to use. You can run this script example with any executable that supports the report package.

Create .csv Files from Reports

```
load_package report
project_open my-project
load_report
# This is the name of the report panel to save as a CSV file
set panel_name "Fitter||Fitter Settings"
set csv_file "output.csv"
set fh [open $csv_file w]
set num_rows [get_number_of_rows -name $panel_name]
# Go through all the rows in the report file, including the
# row with headings, and write out the comma-separated data
for { set i 0 } { $i < $num_rows } { incr i } {
    set row_data [get_report_panel_row -name $panel_name \
        -row $i]
    puts $fh [join $row_data ","]
}
close $fh
unload_report
```

Timing Analysis

The Quartus Prime TimeQuest Timing Analyzer includes support for industry-standard SDC commands in the **sdc** package. The Quartus Prime software includes comprehensive Tcl APIs and SDC extensions for the TimeQuest Timing Analyzer in the **sta**, and **sdc_ext** packages. The Quartus Prime software also includes a **tdc** package that obtains information from the TimeQuest Timing Analyzer.

Related Information
Quartus Prime TimeQuest Timing Analyzer
For information about how to perform timing analysis with the Quartus Prime TimeQuest Timing Analyzer

Automating Script Execution

You can configure scripts to run automatically at various points during compilation. Use this capability to automatically run scripts that perform custom reporting, make specific assignments, and perform many other tasks.

The following three global assignments control when a script is run automatically:

- `PRE_FLOW_SCRIPT_FILE` —before a flow starts
- `POST_MODULE_SCRIPT_FILE` —after a module finishes
- `POST_FLOW_SCRIPT_FILE` —after a flow finishes

A module is another term for a Quartus Prime executable that performs one step in a flow. For example, two modules are Analysis and Synthesis (**quartus_syn**), and timing analysis (**quartus_sta**).

A flow is a series of modules that the Quartus Prime software runs with predefined options. For example, compiling a design is a flow that typically consists of the following steps (performed by the indicated module):

1. Analysis and synthesis (**quartus_syn**)
2. Fitter (**quartus_fit**)
3. Assembler (**quartus_asm**)
4. Timing Analyzer (**quartus_sta**)

Other flows are described in the help for the `execute_flow` Tcl command. In addition, many commands in the Processing menu of the Quartus Prime GUI correspond to this design flow.

To make an assignment automatically run a script, add an assignment with the following form to the **.qsf** for your project:

```
set_global_assignment -name <assignment name> <executable>:<script name>
```

The Quartus Prime software runs the scripts.

```
<executable> -t <script name> <flow or module name> <project name> <revision name>
```

The first argument passed in the `argv` variable (or `quartus(args)` variable) is the name of the flow or module being executed, depending on the assignment you use. The second argument is the name of the project and the third argument is the name of the revision.

When you use the `POST_MODULE_SCRIPT_FILE` assignment, the specified script is automatically run after every executable in a flow. You can use a string comparison with the module name (the first argument passed in to the script) to isolate script processing to certain modules.

Execution Example

To illustrate how automatic script execution works in a complete flow, assume you have a project called **top** with a current revision called **rev_1**, and you have the following assignments in the **.qsf** for your project.

```
set_global_assignment -name PRE_FLOW_SCRIPT_FILE quartus_sh:first.tcl
set_global_assignment -name POST_MODULE_SCRIPT_FILE quartus_sh:next.tcl
set_global_assignment -name POST_FLOW_SCRIPT_FILE quartus_sh:last.tcl
```

When you compile your project, the `PRE_FLOW_SCRIPT_FILE` assignment causes the following command to be run before compilation begins:

```
quartus_sh -t first.tcl compile top rev_1
```

Next, the Quartus Prime software starts compilation with analysis and synthesis, performed by the **quartus_syn** executable. After the Analysis and Synthesis finishes, the `POST_MODULE_SCRIPT_FILE` assignment causes the following command to run:

```
quartus_sh -t next.tcl quartus_syn top rev_1
```

Then, the Quartus Prime software continues compilation with the Fitter, performed by the **quartus_fit** executable. After the Fitter finishes, the POST_MODULE_SCRIPT_FILE assignment runs the following command:

```
quartus_sh -t next.tcl quartus_fit top rev_1
```

Corresponding commands are run after the other stages of the compilation. When the compilation is over, the POST_FLOW_SCRIPT_FILE assignment runs the following command:

```
quartus_sh -t last.tcl compile top rev_1
```

Controlling Processing

The POST_MODULE_SCRIPT_FILE assignment causes a script to run after every module. Because the same script is run after every module, you might have to include some conditional statements that restrict processing in your script to certain modules.

For example, if you want a script to run only after timing analysis, use a conditional test like the following example. It checks the flow or module name passed as the first argument to the script and executes code when the module is **quartus_sta**.

Restrict Processing to a Single Module

```
set module [lindex $quartus(args) 0]
if [string match "quartus_sta" $module] {
    # Include commands here that are run
    # after timing analysis
    # Use the post-message command to display
    # messages
    post_message "Running after timing analysis"
}
```

Displaying Messages

Because of the way the Quartus Prime software runs the scripts automatically, you must use the post_message command to display messages, instead of the puts command. This requirement applies only to scripts that are run by the three assignments listed in "Automating Script Execution".

Related Information

- **The post_message Command** on page 5-14
  For more information about this command.
- **Automating Script Execution** on page 5-10
  For more information on the three scripts capable of scripting-message automation.

Other Scripting Features

The Quartus Prime Tcl API includes other general-purpose commands and features described in this section.

Natural Bus Naming

The Quartus Prime software supports natural bus naming. Natural bus naming allows you to use square brackets to specify bus indexes in HDL without including escape characters to prevent Tcl from interpreting the square brackets as containing commands. For example, one signal in a bus named

address can be identified as `address[0]` instead of `address\[0\]`. You can take advantage of natural bus naming when making assignments.

```
set_location_assignment -to address[10] Pin_M20
```

The Quartus Prime software defaults to natural bus naming. You can turn off natural bus naming with the `disable_natural_bus_naming` command. For more information about natural bus naming, type the following at a Quartus Prime Tcl prompt:

```
enable_natural_bus_naming -h
```

Short Option Names

You can use short versions of command options, as long as they are unambiguous. For example, the `project_open` command supports two options: `-current_revision` and `-revision`.

You can use any of the following abbreviations of the `-revision` option:

- `-r`
- `-re`
- `-rev`
- `-revi`
- `-revis`
- `-revisio`

You can use an option as short as `-r` because in the case of the `project_open` command no other option starts with the letter r. However, the `report_timing` command includes the options `-recovery` and `-removal`. You cannot use `-r` or `-re` to shorten either of those options, because the abbreviation would not be unique to only one option.

Collection Commands

Some Quartus Prime Tcl functions return very large sets of data that would be inefficient as Tcl lists. These data structures are referred to as collections. The Quartus Prime Tcl API uses a collection ID to access the collection.

There are two Quartus Prime Tcl commands for working with collections, `foreach_in_collection` and `get_collection_size`. Use the `set` command to assign a collection ID to a variable.

Related Information

foreach_in_collection
For information about which Quartus Prime Tcl commands return collection IDs.

The foreach_in_collection Command

The `foreach_in_collection` command is similar to the `foreach` Tcl command. Use it to iterate through all elements in a collection. The following example prints all instance assignments in an open project.

foreach_in_collection Example

```
set all_instance_assignments [get_all_instance_assignments -name *]
foreach_in_collection asgn $all_instance_assignments {
    # Information about each assignment is
    # returned in a list. For information
    # about the list elements, refer to Help
    # for the get-all-instance-assignments command.
```

```
        set to [lindex $asgn 2]
        set name [lindex $asgn 3]
        set value [lindex $asgn 4]
        puts "Assignment to $to: $name = $value"
    }
```

The get_collection_size Command

Use the `get_collection_size` command to get the number of elements in a collection. The following example prints the number of global assignments in an open project.

get_collection_size Example

```
    set all_global_assignments [get_all_global_assignments -name *]
    set num_global_assignments [get_collection_size $all_global_assignments]
    puts "There are $num_global_assignments global assignments in your project"
```

The post_message Command

To print messages that are formatted like Quartus Prime software messages, use the `post_message` command. Messages printed by the `post_message` command appear in the **System** tab of the Messages window in the Quartus Prime GUI, and are written to standard at when scripts are run. Arguments for the `post_message` command include an optional message type and a required message string.

The message type can be one of the following:

- `info` (default)
- `extra_info`
- `warning`
- `critical_warning`
- `error`

If you do not specify a type, Quartus Prime software defaults to `info`.

With the Quartus Prime software in Windows, you can color code messages displayed at the system command prompt with the `post_message` command. Add the following line to your **quartus2.ini** file:

```
    DISPLAY_COMMAND_LINE_MESSAGES_IN_COLOR = on
```

The following example shows how to use the `post_message` command.

```
    post_message -type warning "Design has gated clocks"
```

Accessing Command-Line Arguments

Many Tcl scripts are designed to accept command-line arguments, such as the name of a project or revision. The global variable `quartus(args)` is a list of the arguments typed on the command-line following the name of the Tcl script. The following Tcl example prints all of the arguments in the `quartus(args)` variable.

Simple Command-Line Argument Access

```
    set i 0
    foreach arg $quartus(args) {
        puts "The value at index $i is $arg"
```

```
        incr i
    }
```

If you copy the script in the previous example to a file named **print_args.tcl**, it displays the following output when you type the following at a command prompt.

Passing Command-Line Arguments to Scripts

```
quartus_sh -t print_args.tcl my_project 100MHz
The value at index 0 is my_project
The value at index 1 is 100MHz
```

The cmdline Package

You can use the **cmdline** package included with the Quartus Prime software for more robust and self-documenting command-line argument passing. The **cmdline** package supports command-line arguments with the form *-<option><value>* .

cmdline Package

```
package require cmdline
variable ::argv0 $::quartus(args)
set options {
    { "project.arg" "" "Project name" }
    { "frequency.arg" "" "Frequency" }
}
set usage "You need to specify options and values"
array set optshash [::cmdline::getoptions ::argv $options $usage]
puts "The project name is $optshash(project)"
puts "The frequency is $optshash(frequency)"
```

If you save those commands in a Tcl script called **print_cmd_args.tcl** you see the following output when you type the following command at a command prompt.

Passing Command-Line Arguments for Scripts

```
quartus_sh -t print_cmd_args.tcl -project my_project -frequency 100MHz
The project name is my_project
The frequency is 100MHz
```

Virtually all Quartus Prime Tcl scripts must open a project. You can open a project, and you can optionally specify a revision name with code like the following example. The example checks whether the specified project exists. If it does, the example opens the current revision, or the revision you specify.

Full-Featured Method to Open Projects

```
package require cmdline
variable ::argv0 $::quartus(args)
set options { \
{ "project.arg" "" "Project Name" } \
{ "revision.arg" "" "Revision Name" } \
}
array set optshash [::cmdline::getoptions ::argv0 $options]
# Ensure the project exists before trying to open it
if {[project_exists $optshash(project)]} {
    if {[string equal "" $optshash(revision)]} {
        # There is no revision name specified, so default
        # to the current revision
        project_open $optshash(project) -current_revision
```

```
    } else {
        # There is a revision name specified, so open the
        # project with that revision
        project_open $optshash(project) -revision \
            $optshash(revision)
    }
} else {
    puts "Project $optshash(project) does not exist"
    exit 1
}
# The rest of your script goes here
```

If you do not require this flexibility or error checking, you can use just the `project_open` command.

Simple Method to Open Projects

```
set proj_name [lindex $argv 0]
project_open $proj_name
```

The quartus() Array

The scripts in the preceding examples parsed command line arguments found in `quartus(args)`. The global `quartus()` Tcl array includes other information about your project and the current Quartus Prime executable that might be useful to your scripts. For information on the other elements of the `quartus()` array, type the following command at a Tcl prompt:

```
help -quartus
```

The Quartus Prime Tcl Shell in Interactive Mode

This section presents how to make project assignments and then compile the finite impulse response (FIR) filter tutorial project with the `quartus_sh` interactive shell. This example assumes that you already have the **fir_filter** tutorial design files in a project directory.

To begin, type the following at the system command prompt to run the interactive Tcl shell:

```
quartus_sh -s
```

Create a new project called **fir_filter**, with a revision called **filtref** by typing the following command at a Tcl prompt:

```
project_new -revision filtref fir_filter
```

Note: If the project file and project name are the same, the Quartus Prime software gives the revision the same name as the project.

Because the revision named **filtref** matches the top-level file, all design files are automatically picked up from the hierarchy tree.

Next, set a global assignment for the device with the following command:

```
set_global_assignment -name family <device family name>
```

To learn more about assignment names that you can use with the `-name` option, refer to Quartus Prime Help.

Note: For assignment values that contain spaces, enclose the value in quotation marks.

To compile a design, use the `::quartus::flow` package, which properly exports the new project assignments and compiles the design with the proper sequence of the command-line executables. First, load the package:

```
load_package flow
```

It returns the following:

```
1.0
```

To perform a full compilation of the FIR filter design, use the `execute_flow` command with the `-compile` option:

```
exectue_flow -compile
```

This command compiles the FIR filter tutorial project, exporting the project assignments and running `quartus_syn`, `quartus_fit`, `quartus_asm`, and `quartus_sta`. This sequence of events is the same as selecting **Start Compilation** from the Processing menu in the Quartus Prime GUI.

When you are finished with a project, close it with the `project_close` command.

To exit the interactive Tcl shell, type `exit` at a Tcl prompt.

The tclsh Shell

On the UNIX and Linux operating systems, the tclsh shell included with the Quartus Prime software is initialized with a minimal `PATH` environment variable. As a result, system commands might not be available within the tclsh shell because certain directories are not in the `PATH` environment variable.

To include other directories in the path searched by the tclsh shell, set the `QUARTUS_INIT_PATH` environment variable before running the tclsh shell. Directories in the `QUARTUS_INIT_PATH` environment variable are searched by the tclsh shell when you execute a system command.

Tcl Scripting Basics

The core Tcl commands support variables, control structures, and procedures. Additionally, there are commands for accessing the file system and network sockets, and running other programs. You can create platform-independent graphical interfaces with the Tk widget set.

Tcl commands are executed immediately as they are typed in an interactive Tcl shell. You can also create scripts (including the examples in this chapter) in files and run them with the Quartus Prime executables or with the tclsh shell.

Hello World Example

The following shows the basic "Hello world" example in Tcl:

```
puts "Hello world"
```

Use double quotation marks to group the words `hello` and `world` as one argument. Double quotation marks allow substitutions to occur in the group. Substitutions can be simple variable substitutions, or the result of running a nested command. Use curly braces `{}` for grouping when you want to prevent substitutions.

Variables

Assign a value to a variable with the `set` command. You do not have to declare a variable before using it. Tcl variable names are case-sensitive.

```
set a 1
```

To access the contents of a variable, use a dollar sign ("`$`") before the variable name. The following example prints `"Hello world"` in a different way.

```
set a Hello
set b world
puts "$a $b"
```

Substitutions

Tcl performs three types of substitution:

- Variable value substitution
- Nested command substitution
- Backslash substitution

Variable Value Substitution

Variable value substitution, refers to accessing the value stored in a variable with a dollar sign ("`$`") before the variable name.

Nested Command Substitution

Nested command substitution refers to how the Tcl interpreter evaluates Tcl code in square brackets. The Tcl interpreter evaluates nested commands, starting with the innermost nested command, and commands nested at the same level from left to right. Each nested command result is substituted in the outer command.

```
set a [string length foo]
```

Backslash Substitution

Backslash substitution allows you to quote reserved characters in Tcl, such as dollar signs ("`$`") and braces ("`[ ]`"). You can also specify other special ASCII characters like tabs and new lines with backslash substitutions. The backslash character is the Tcl line continuation character, used when a Tcl command wraps to more than one line.

```
set this_is_a_long_variable_name [string length "Hello \
    world."]
```

Arithmetic

Use the `expr` command to perform arithmetic calculations. Use curly braces ("`{ }`") to group the arguments of this command for greater efficiency and numeric precision.

```
set a 5
set b [expr { $a + sqrt(2) }]
```

Tcl also supports boolean operators such as `&&` (AND), `||` (OR), `!` (NOT), and comparison operators such as `<` (less than), `>` (greater than), and `==` (equal to).

Lists

A Tcl list is a series of values. Supported list operations include creating lists, appending lists, extracting list elements, computing the length of a list, sorting a list, and more.

```
set a { 1 2 3 }
```

You can use the `lindex` command to extract information at a specific index in a list. Indexes are zero-based. You can use the index `end` to specify the last element in the list, or the index `end-<n>` to count from the end of the list. For example to print the second element (at index 1) in the list stored in `a` use the following code.

```
puts [lindex $a 1]
```

The `llength` command returns the length of a list.

```
puts [llength $a]
```

The `lappend` command appends elements to a list. If a list does not already exist, the list you specify is created. The list variable name is not specified with a dollar sign ("$").

```
lappend a 4 5 6
```

Arrays

Arrays are similar to lists except that they use a string-based index. Tcl arrays are implemented as hash tables. You can create arrays by setting each element individually or with the `array set` command.

To set an element with an index of `Mon` to a value of `Monday` in an array called `days`, use the following command:

```
set days(Mon) Monday
```

The `array set` command requires a list of index/value pairs. This example sets the array called days:

```
array set days { Sun Sunday Mon Monday Tue Tuesday \
    Wed Wednesday Thu Thursday Fri Friday Sat Saturday }


set day_abbreviation Mon
puts $days($day_abbreviation)
```

Use the `array names` command to get a list of all the indexes in a particular array. The index values are not returned in any specified order. The following example is one way to iterate over all the values in an array.

```
foreach day [array names days] {
    puts "The abbreviation $day corresponds to the day \
name $days($day)"
}
```

Arrays are a very flexible way of storing information in a Tcl script and are a good way to build complex data structures.

Control Structures

Tcl supports common control structures, including if-then-else conditions and `for`, `foreach`, and `while` loops. The position of the curly braces as shown in the following examples ensures the control structure commands are executed efficiently and correctly. The following example prints whether the value of variable `a` positive, negative, or zero.

If-Then-Else Structure

```
if { $a > 0 } {
    puts "The value is positive"
} elseif { $a < 0 } {
    puts "The value is negative"
} else {
    puts "The value is zero"
}
```

The following example uses a `for` loop to print each element in a list.

For Loop

```
set a { 1 2 3 }
for { set i 0 } { $i < [llength $a] } { incr i } {
    puts "The list element at index $i is [lindex $a $i]"
}
```

The following example uses a `foreach` loop to print each element in a list.

foreach Loop

```
set a { 1 2 3 }
foreach element $a {
    puts "The list element is $element"
}
```

The following example uses a `while` loop to print each element in a list.

while Loop

```
set a { 1 2 3 }
set i 0
while { $i < [llength $a] } {
    puts "The list element at index $i is [lindex $a $i]"
    incr i
}
```

You do not have to use the `expr` command in boolean expressions in control structure commands because they invoke the `expr` command automatically.

Procedures

Use the `proc` command to define a Tcl procedure (known as a subroutine or function in other scripting and programming languages). The scope of variables in a procedure is local to the procedure. If the procedure returns a value, use the `return` command to return the value from the procedure. The following example defines a procedure that multiplies two numbers and returns the result.

Simple Procedure

```
proc multiply { x y } {
    set product [expr { $x * $y }]
    return $product
}
```

The following example shows how to use the `multiply` procedure in your code. You must define a procedure before your script calls it.

Using a Procedure

```
proc multiply { x y } {
    set product [expr { $x * $y }]
    return $product
}
set a 1
set b 2
puts [multiply $a $b]
```

Define procedures near the beginning of a script. If you want to access global variables in a procedure, use the `global` command in each procedure that uses a global variable.

Accessing Global Variables

```
proc print_global_list_element { i } {
    global my_data
    puts "The list element at index $i is [lindex $my_data $i]"
}
set my_data { 1 2 3}
print_global_list_element 0
```

File I/O

Tcl includes commands to read from and write to files. You must open a file before you can read from or write to it, and close it when the read and write operations are done. To open a file, use the `open` command; to close a file, use the `close` command. When you open a file, specify its name and the mode in which to open it. If you do not specify a mode, Tcl defaults to read mode. To write to a file, specify `w` for write mode.

Open a File for Writing

```
set output [open myfile.txt w]
```

Tcl supports other modes, including appending to existing files and reading from and writing to the same file.

The `open` command returns a file handle to use for read or write access. You can use the `puts` command to write to a file by specifying a filehandle.

Write to a File

```
set output [open myfile.txt w]
puts $output "This text is written to the file."
close $output
```

You can read a file one line at a time with the `gets` command. The following example uses the `gets` command to read each line of the file and then prints it out with its line number.

Read from a File

```
set input [open myfile.txt]
set line_num 1
while { [gets $input line] >= 0 } {
    # Process the line of text here
    puts "$line_num: $line"
    incr line_num
}
close $input
```

Syntax and Comments

Arguments to Tcl commands are separated by white space, and Tcl commands are terminated by a newline character or a semicolon. You must use backslashes when a Tcl command extends more than one line.

Tcl uses the hash or pound character (#) to begin comments. The # character must begin a comment. If you prefer to include comments on the same line as a command, be sure to terminate the command with a semicolon before the # character. The following example is a valid line of code that includes a `set` command and a comment.

```
set a 1;# Initializes a
```

Without the semicolon, it would be an invalid command because the `set` command would not terminate until the new line after the comment.

The Tcl interpreter counts curly braces inside comments, which can lead to errors that are difficult to track down. The following example causes an error because of unbalanced curly braces.

```
# if { $x > 0 } {
if { $y > 0 } {
    # code here
}
```

External References

For more information about Tcl, refer to the following sources:

- Brent B. Welch and Ken Jones, and Jeffery Hobbs, *Practical Programming in Tcl and Tk* (Upper Saddle River: Prentice Hall, 2003)
- John Ousterhout and Ken Jones, *Tcl and the Tk Toolkit* (Boston: Addison-Wesley Professional, 2009)
- Mark Harrison and Michael McLennan, *Effective Tcl/Tk Programming: Writing Better Programs in Tcl and Tk* (Boston: Addison-Wesley Professional, 1997)

Related Information

- **Quartus Prime Tcl Examples**
  For Quartus Prime Tcl example scripts
- **tcl.activestate.com**
  Tcl Developer Xchange

Document Revision History

Table 5-4: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	- Changed instances of *Quartus II* to *Quartus Prime*. - Updated the list of Tcl packages in the *Quartus Prime Tcl Packages* section. - Updated the *Quartus Prime Tcl API Help* section: - Updated the Tcl Help Output
June 2014	14.0.0	Updated the format.
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	- Template update - Updated supported version of Tcl in the section "Tool Command Language." - minor editoral changes
May 2011	11.0.0	Minor updates throughout document.
December 2010	10.1.0	Template update Updated to remove tcl packages used by the Classic Timing Analyzer
July 2010	10.0.0	Minor updates throughout document.
November 2009	9.1.0	- Removed LogicLock Plus example. - Added the incremental_compilation, insystem_source_probe, and rtl packages to Table 3-1 and Table 3-2. - Added quartus_map to table 3-2.
March 2009	9.0.0	- Removed the "EDA Tool Assignments" section - Added the section "Compile All Revisions" on page 3–9 - Added the section "Using the tclsh Shell" on page 3–20
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated references.

Related Information

Quartus Handbook Archive

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

Altera FPGAs and CPLDs offer a multitude of configurable options to allow you to implement a custom application-specific circuit on your PCB.

Your Quartus Prime project provides important information specific to your programmable logic design, which you can use in conjunction with the device literature available on Altera's website to ensure that you implement the correct board-level connections in your schematic.

Refer to the **Settings** dialog box options, the Fitter report, and Messages window when creating and reviewing your PCB schematic. The Quartus Prime software also provides the Pin Planner and the SSN Analyzer to assist you during your PCB schematic review process.

Related Information

- **Schematic Review Worksheets**
- **Pin Connection Guidelines**

Reviewing Quartus Prime Software Settings

Review these settings in the Quartus Prime software to help you review your PCB schematic.

The **Device** dialog box in the Quartus Prime software allows you to specify device-specific assignments and settings. You can use the **Device** dialog box to specify general project-wide options, including specific device and pin options, which help you to implement correct board-level connections in your PCB schematic.

The **Device** dialog box provides project-specific device information, including the target device and any migration devices you specify. Using migration devices can impact the number of available user I/O pins and internal resources, as well as require connection of some user I/O pins to power/ground pins to support migration.

If you want to use vertical migration, which allows you to use different devices with the same package, you can specify your list of migration devices in the **Migration Devices** dialog box. The Fitter places the pins in your design based on your targeted migration devices, and allows you to use only I/O pins that are common to all of the migration devices.

If a migration device has pins that are power or ground, but the pins are also user I/O pins on a different device in the migration path, the Fitter ensures that these pins are not used as user I/O pins. You must ensure that these pins are connected to the appropriate plane on the PCB.

ISO 9001:2008 Registered

ALTERA®

If you are migrating from a smaller device with NC (no-connect) pins to a larger device with power or ground pins in the same package, you can safely connect the NC pins to power or ground pins to facilitate successful migration.

Related Information
Migration Devices Dialog Box
For more information about the **Migration Devices** dialog box in the Quartus Prime software

Device and Pins Options Dialog Box Settings

You can set device and pin options and verify important design-specific data in the **Device and Pin Options** dialog box, including options found on the **General**, **Configuration**, **Unused Pin**, **Dual-Purpose Pins**, and **Voltage** pages.

Configuration Settings

The **Configuration** page of the **Device and Pin Options** dialog box specifies the configuration scheme and configuration device for the target device. Use the **Configuration** page settings to verify the configuration scheme with the MSEL pin settings used on your PCB schematic and the I/O voltage of the configuration scheme.

Your specific configuration settings may impact the availability of some dual-purpose I/O pins in user mode.

Related Information
Dual-Purpose Pins Settings on page 6-2

Unused Pin Settings

The **Unused Pin** page specifies the behavior of all unused pins in your design. Use the **Unused Pin** page to ensure that unused pin settings are compatible with your PCB. For example, if you reserve all unused pins as outputs driving ground, you must ensure that you do not connect unused I/O pins to VCC pins on your PCB. Connecting unused I/O pins to VCC pins may result in contention that could lead to higher than expected current draw and possible device overstress.

The **Reserve all unused pins** list shows available unused pin state options for the target device. The default state for each pin is the recommended setting for each device family.

When you reserve a pin as output driving ground, the Fitter connects a ground signal to the output pin internally. You should connect the output pin to the ground plane on your PCB, although you are not required to do so. Connecting the output driving ground to the ground plane is known as creating a virtual ground pin, which helps to minimize simultaneous switching noise (SSN) and ground bounce effects.

Dual-Purpose Pins Settings

The **Dual-Purpose Pins** page specifies how configuration pins should be used after device configuration completes. You can set the function of the dual-purpose pins by selecting a value for a specific pin in the **Dual-purpose pins** list. Pin functions should match your PCB schematic. The available options on the **Dual-Purpose Pins** page may differ depending on the selected configuration mode.

Voltage Settings

The **Voltage** page specifies the default VCCIO I/O bank voltage and the default I/O bank voltage for the pins on the target device. VCCIO I/O bank voltage settings made in the **Voltage** page are overridden by I/O standard assignments made on I/O pins in their respective banks.

Related Information

Reviewing Device Pin-Out Information in the Fitter Report on page 6-3

Error Detection CRC Settings

The **Error Detection CRC** page specifies error detection cyclic redundancy check (CRC) use for the target device. When **Enable error detection CRC** is turned on, the device checks the validity of the programming data in the devices. Any changes made in the data while the device is in operation generates an error.

Turning on the **Enable open drain on CRC error pin** option allows the CRC ERROR pin to be set as an open-drain pin in some devices, which decouples the voltage level of the CRC ERROR pin from VCCIO voltage. You must connect a pull-up resistor to the CRC ERROR pin on your PCB if you turn on this option.

In addition to settings in the **Device** dialog box, you should verify settings in the **Voltage** page of the **Settings** dialog box.

Related Information

Device and Pin Options Dialog Box
For more information about the **Device and Pins Options** dialog box in the Quartus Prime software

Voltage Settings

The **Voltage** page, under **Operating Settings and Conditions** in the **Settings** dialog box, allows you to specify voltage operating conditions for timing and power analyses. Ensure that the settings in the **Voltage** page match the settings in your PCB schematic, especially if the target device includes transceivers.

The **Voltage** page settings requirements differ depending on the settings of the transceiver instances in the design. Refer to the Fitter report for the required settings, and verify that the voltage settings are correctly set up for your PCB schematic.

After verifying your settings in the **Device** and **Settings** dialog boxes, you can verify your device pin-out with the Fitter report.

Related Information

Pin Connection Guidelines
For more information about voltage settings

Reviewing Device Pin-Out Information in the Fitter Report

After you compile your design, you can use the reports in the Resource section of the Fitter report to check your device pin-out in detail.
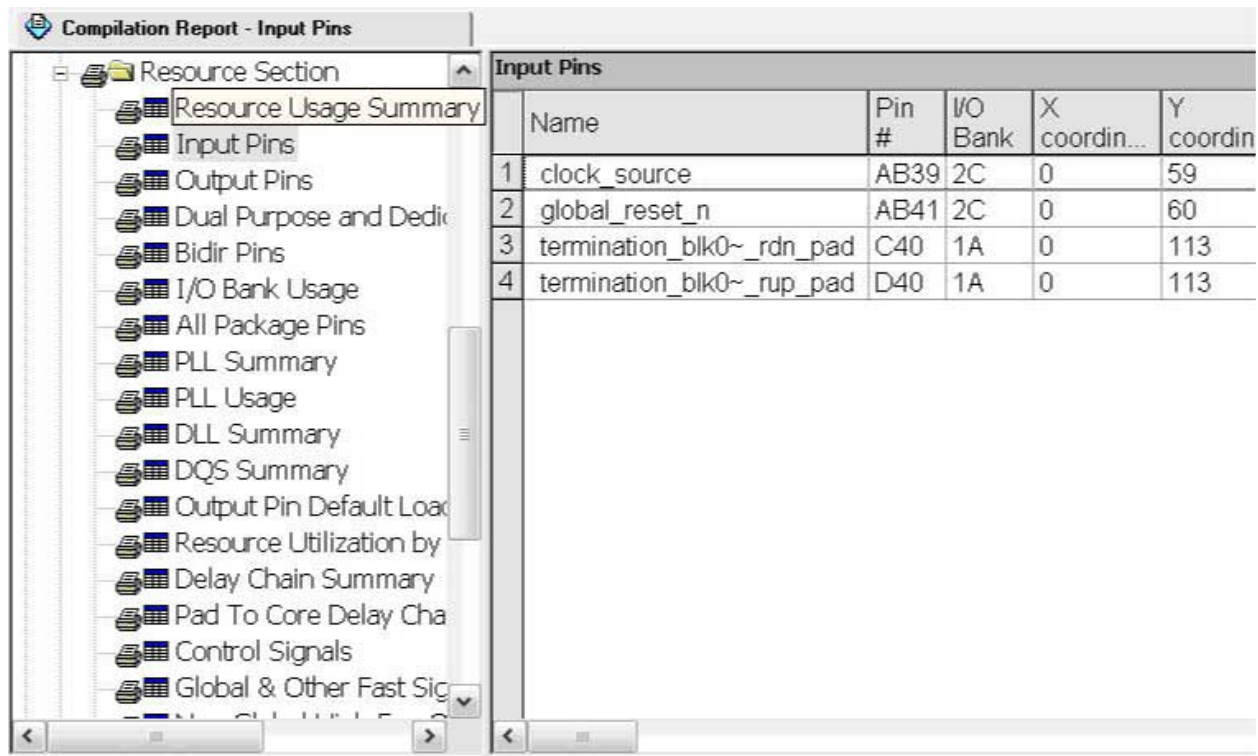
The Input Pins, Output Pins, and Bidirectional Pins reports identify all the user I/O pins in your design and the features enabled for each I/O pin. For example, you can find use of weak internal pull-ups, PCI

clamp diodes, and on-chip termination (OCT) pin assignments in these sections of the Fitter report. You can check the pin assignments reported in the Input Pins, Output Pins, and Bidirectional Pins reports against your PCB schematic to determine whether your PCB requires external components.

These reports also identify whether you made pin assignments or if the Fitter automatically placed the pins. If the Fitter changed your pin assignments, you should make these changes user assignments because the location of pin assignments made by the Fitter may change with subsequent compilations.

Figure 6-1: Resource Section Report

This figure shows the pins the Fitter chose for the OCT external calibration resistor connections (RUP/ RDN) and the name of the associated termination block in the Input Pins report. You should make these types of assignments user assignments.

The I/O Bank Usage report provides a high-level overview of the VCCIO and VREF requirements for your design, based on your I/O assignments. Verify that the requirements in this report match the settings in your PCB schematic. All unused I/O banks, and all banks with I/O pins with undefined I/O standards, default the VCCIO voltage to the voltage defined in the **Voltage** page of the **Device and Pin Options** dialog box.

The All Package Pins report lists all the pins on your device, including unused pins, dedicated pins and power/ground pins. You can use this report to verify pin characteristics, such as the location, name, usage, direction, I/O standard and voltage for each pin with the pin information in your PCB schematic. In particular, you should verify the recommended voltage levels at which you connect unused dedicated inputs and I/O and power pins, especially if you selected a migration device. Use the All Package Pins report to verify that you connected all the device voltage rails to the voltages reported.

Errors commonly reported include connecting the incorrect voltage to the predriver supply (VCCPD) pin in a specific bank, or leaving dedicated clock input pins floating. Unused input pins that should be connected to ground are designated as **GND+** in the **Pin Name/Usage** column in the All Package Pins report.

You can also use the All Package Pins report to check transceiver-specific pin connections and verify that they match the PCB schematic. Unused transceiver pins have the following requirements, based on the pin designation in the Fitter report:

- GXB_GND—Unused GXB receiver or dedicated reference clock pin. This pin must be connected to GXB_GND through a 10k Ohm resistor.
- GXB_NC—Unused GXB transmitter or dedicated clock output pin. This pin must be disconnected.

Some transceiver power supply rails have dual voltage capabilities, such as VCCA_L/R and VCCH_L/R, that depend on the settings you created for the ALTGX parameter editor. Because these user-defined settings overwrite the default settings, you should use the All Package Pins report to verify that these power pins on the device symbol in the PCB schematics are connected to the voltage required by the transceiver. An incorrect connection may cause the transceiver to function not as expected.

If your design includes a memory interface, the DQS Summary report provides an overview of each DQ pin group. You can use this report to quickly confirm that the correct DQ/DQS pins are grouped together.

Finally, the Fitter Device Options report summarizes some of the settings made in the **Device and Pin Options** dialog box. Verify that these settings match your PCB schematics.

Reviewing Compilation Error and Warning Messages

If your project does not compile without error or warning messages, you should resolve the issues identified by the Compiler before signing off on your pin-out or PCB schematic. Error messages often indicate illegal or unsupported use of the device resources and IP.

Additionally, you should cross-reference fitting and timing analysis warnings with the design implementation. Timing may be constrained due to nonideal pin placement. You should investigate if you can reassign pins to different locations to prevent fitting and timing analysis warnings. Ensure that you review each warning and consider its potential impact on the design.

Using Additional Quartus Prime Software Features

You can generate IBIS files, which contain models specific to your design and selected I/O standards and options, with the Quartus Prime software.

Because board-level simulation is important to verify, you should check for potential signal integrity issues. You can turn on the **Board-Level Signal Integrity** feature in the **EDA Tool Settings** page of the **Settings** dialog box.

Additionally, using advanced I/O timing allows you to enter physical PCB information to accurately model the load seen by an output pin. This feature facilitates accurate I/O timing analysis.

Related Information

- **Signal Integrity Analysis with Third-Party Tools** on page 14-1
  For more information about signal integrity analysis in the Quartus Prime software

- **I/O Management** on page 2-1
  For more information about advanced I/O timing

Using Additional Quartus Prime Software Tools

Use the Pin Planner and the SSN Analyzer to assist you with reviewing your PCB schematics.

Pin Planner

The Quartus Prime Pin Planner helps you visualize, plan, and assign device I/O pins in a graphical view of the target device package. You can quickly locate various I/O pins and assign them design elements or other properties to ensure compatibility with your PCB layout.

You can use the Pin Planner to verify the location of clock inputs, and whether they have been placed on dedicated clock input pins, which is recommended when your design uses PLLs.

You can also use the Pin Planner to verify the placement of dedicated SERDES pins. SERDES receiver inputs can be placed only on DIFFIO_RX pins, while SERDES transmitter outputs can be placed only on DIFFIO_TX pins.

The Pin Planner gives a visual indication of signal-to-signal proximity in the Pad View window, and also provides information about differential pin pair placement, such as the placement of pseudo-differential signals.

Related Information

- **I/O Management** on page 2-1
  For more information about the Pin Planner

SSN Analyzer

The SSN Analyzer supports pin planning by estimating the voltage noise caused by the simultaneous switching of output pins on the device. Because of the importance of the potential SSN performance for a specific I/O placement, you can use the SSN Analyzer to analyze the effects of aggressor I/O signals on a victim I/O pin.

Document Revision History

Table 6-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
June 2014	14.0.0	Template update.
November 2012	12.1.0	Minor update of Pin Planner description for task and report windows.
June 2012	12.0.0	Removed survey link.

Date	Version	Changes
November 2011	10.0.2	Template update.
December 2010	10.0.1	Changed to new document template. No change to content.
July 2010	10.0.0	Initial release.

Related Information

Quartus Handbook Archive

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

QPP5V2 ✉ **Subscribe** 💬 **Send Feedback**

Design Optimization Overview

This chapter introduces features in the Altera Quartus Prime software that you can use to achieve the highest design performance when you design for programmable logic devices (PLDs), especially high density FPGAs.

Physical implementation can be an intimidating and challenging phase of the design process. The Quartus Prime software provides a comprehensive environment for FPGA designs, delivering unmatched performance, efficiency, and ease-of-use.

In a typical design flow, you must synthesize your design with Quartus Prime integrated synthesis or a third-party tool, place and route your design with the Fitter, and use the TimeQuest timing analyzer to ensure your design meets the timing requirements. With the PowerPlay Power Analyzer, you ensure the design's power consumption is within limits.

Initial Compilation: Required Settings

There are basic assignments and settings Altera recommends for your initial compilation. Check the following settings before compiling your design in the Quartus Prime software. Significantly varied compilation results can occur depending on the assignments that you set.

Device Settings

Device assignments determine the timing model that the Quartus Prime software uses during compilation.

Choose the correct speed grade to obtain accurate results and the best optimization. The device size and the package determine the device pin-out and the available resources in the device.

Device Migration Settings

If you anticipate a change to the target device later in the design cycle, either because of changes in your design or other considerations, plan for the change at the beginning of your design cycle.

Whenever you select a target device, you can also list any other compatible devices you can migrate by clicking on the **Migration Devices** button in the **Device** dialog box.

Selecting the migration device and companion device early in the design cycle helps to minimize changes to your design at a later stage.

**ISO
9001:2008
Registered**

I/O Assignments

The I/O standards and drive strengths specified for a design affect I/O timing. Specify I/O assignments so that the Quartus Prime software uses accurate I/O timing delays in timing analysis and Fitter optimizations.

If there is no PCB layout requirement, then you do not need to specify pin locations. If your pin locations are not fixed due to PCB layout requirements, then leave the pin locations unconstrained. If your pin locations are already fixed, then make pin assignments to constrain the compilation appropriately.

Use the Assignment Editor and Pin Planner to assign I/O standards and pin locations.

Related Information

- **Timing Closure and Optimization** on page 9-1
  For more information about recommendations for making pin assignments that can have a large effect on your results in smaller macrocell-based architectures.
- **I/O Management** on page 2-1
  For more information about I/O standards and pin constraints, refer to the appropriate device handbook. For more information about planning and checking I/O assignments.

Timing Requirement Settings

Use your real requirements to get the best results. If you apply more demanding timing requirements than you need, then increased resource usage, higher power utilization, increased compilation time, or all of these may result.

You must use comprehensive timing requirement settings to achieve the best results for the following reasons:

- Correct timing assignments enable the software to work hardest to optimize the performance of the timing-critical parts of your design and make trade-offs for performance. This optimization can also save area or power utilization in non-critical parts of your design.
- If enabled, the Quartus Prime software performs physical synthesis optimizations based on timing requirements.
- Depending on the **Fitter Effort** setting, the Fitter can reduce runtime if your design meets the timing requirements.

The Quartus Prime TimeQuest Timing Analyzer determines if the design implementation meets the timing requirement. The Compilation Report shows whether your design meets the timing requirements, while the timing analysis reporting commands provide detailed information about the timing paths.

To create timing constraints for the TimeQuest analyzer, create a Synopsys Design Constraints File (**.sdc**). You can also enter constraints in the TimeQuest GUI. Use the `write_sdc` command, or the Constraints menu in the TimeQuest analyzer. Click **Write SDC File** to write your constraints to an **.sdc**. You can add an **.sdc** to your project on the **Quartus Prime Settings** page under **Timing Analysis Settings**.

Note: If you already have an **.sdc** in your project, using the `write_sdc` command from the command line or using the **Write SDC File** option from the TimeQuest GUI allows you to create a new **.sdc** that combines the constraints from your current **.sdc** and any new constraints added through the GUI or command window, or overwrites the existing **.sdc** with your newly applied constraints.

Ensure that every clock signal has an accurate clock setting constraint. If clocks arrive from a common oscillator, then they are related. Ensure that you set up all related or derived clocks in the constraints correctly. You must constrain all I/O pins that require I/O timing optimization. Specify both minimum and maximum timing constraints as applicable. If your design contains more than one clock or contains

pins with different I/O requirements, make multiple clock settings and individual I/O assignments instead of using a global constraint.

Make any complex timing assignments required in your design, including false path and multicycle path assignments. Common situations for these types of assignments include reset or static control signals (when the time required for a signal to reach a destination is not important) or paths that have more than one clock cycle available for operation in a design. These assignments enable the Quartus Prime software to make appropriate trade-offs between timing paths and can enable the Compiler to improve timing performance in other parts of your design.

Note: To ensure that you apply constraints or assignments to all design nodes, you can report all unconstrained paths in your design with the **Report Unconstrained Paths** command in the **Task** pane of the Quartus Prime TimeQuest Timing Analyzer or the `report_ucp` Tcl command.

Related Information

- **Timing Closure and Optimization** on page 9-1
  For more information about optimization with physical synthesis.
- **Advanced Settings (Fitter)**
  For more information about reducing runtime by changing Fitter effort.
- **The Quartus Prime TimeQuest Timing Analyzer**
  For more information about timing assignments and timing analysis.
- **Quartus Prime TimeQuest Timing Analyzer Cookbook**
  For more information about timing assignments and timing analysis.

Physical Implementation

Most optimization issues involve preserving previous results, reducing area, reducing critical path delay, reducing power consumption, and reducing runtime.

The Quartus Prime software includes advisors to address each of these issues and helps you optimize your design. Run these advisors during physical implementation for advice about your specific design.

You can reduce the time spent on design iterations by following the recommended design practices for designing with Altera® devices. Design planning is critical for successful design timing implementation and closure.

Related Information
Design Planning with the Quartus Prime Software

Trade-Offs and Limitations

Many optimization goals can conflict with one another, so you might need to resolve conflicting goals. For example, one major trade-off during physical implementation is between resource usage and critical path timing, because certain techniques (such as logic duplication) can improve timing performance at the cost of increased area. Similarly, a change in power requirements can result in area and timing trade-offs, such as if you reduce the number of available high-speed tiles, or if you attempt to shorten high-power nets at the expense of critical path nets.

In addition, system cost and time-to-market considerations can affect the choice of device. For example, a device with a higher speed grade or more clock networks can facilitate timing closure at the expense of higher power consumption and system cost.

Finally, not all designs can be realized in a hardware circuit with limited resources and given constraints. If you encounter resource limitations, timing constraints, or power constraints that cannot be resolved by the Fitter, consider rewriting parts of the HDL code.

Related Information

- **Timing Closure and Optimization** on page 9-1

Reducing Area

By default, the Quartus Prime Fitter might physically spread a design over the entire device to meet the set timing constraints. If you prefer to optimize your design to use the smallest area, you can change this behavior. If you require reduced area, you can enable certain physical synthesis options to modify your netlist to create a more area-efficient implementation, but at the cost of increased runtime and decreased performance.

Related Information

Netlist Optimizations and Physical Synthesis on page 13-1

Timing Closure and Optimization on page 9-1

Recommended HDL Coding Styles

Reducing Critical Path Delay

To meet complex timing requirements involving multiple clocks, routing resources, and area constraints, the Quartus Prime software offers a close interaction between synthesis, timing analysis, floorplan editing, and place-and-route processes.

By default, the Quartus Prime Fitter tries to meet the specified timing requirements and stops trying when the requirements are met. Therefore, using realistic constraints is important to successfully close timing. If you under-constrain your design, you may get sub-optimal results. By contrast, if you over-constrain your design, the Fitter might over-optimize non-critical paths at the expense of true critical paths. In addition, you might incur an increased area penalty. Compilation time may also increase because of excessively tight constraints.

If your resource usage is very high, the Quartus Prime Fitter might have trouble finding a legal placement. In such circumstances, the Fitter automatically modifies some of its settings to try to trade off performance for area.

The Quartus Prime Fitter offers a number of advanced options that can help you improve the performance of your design when you properly set constraints. Use the Timing Optimization Advisor to determine which options are best suited for your design.

In high-density FPGAs, routing accounts for a major part of critical path timing. Because of this, duplicating or retiming logic can allow the Fitter to reduce delay on critical paths. The Quartus Prime software offers push-button netlist optimizations and physical synthesis options that can improve design performance at the expense of considerable increases of compilation time and area. Turn on only those options that help you keep reasonable compilation times and resource usage. Alternately, you can modify your HDL to manually duplicate or adjust the timing logic.

Reducing Power Consumption

The Quartus Prime software has features that help reduce design power consumption. The PowerPlay power optimization options control the power-driven compilation settings for Synthesis and the Fitter.

Related Information

- **Power Optimization** on page 10-1

Reducing Runtime

Many Fitter settings influence compilation time. Most of the default settings in the Quartus Prime software are set for reduced compilation time. You can modify these settings based on your project requirements.

The Quartus Prime software supports parallel compilation in computers with multiple processors. This can reduce compilation times by up to 15% while giving the identical result as serial compilation.

Using Quartus Prime Tools

The following sections describe several Quartus Prime tools that you can use to help optimize your design.

Design Analysis

The Quartus Prime software provides tools that help with a visual representation of your design. You can use the RTL Viewer to see a schematic representation of your design before synthesis and place-and-route.The Technology Map Viewer provides a schematic representation of the design implementation in the selected device architecture after synthesis and place-and-route. It can also include timing information.

Advisors

The Quartus Prime software includes several advisors to help you optimize your design and reduce compilation time.

You can complete your design faster by following the recommendations in the following advisor. These advisors give recommendations based on your project settings and your design constraints:

- Resource Optimization Advisor
- Timing Optimization Advisor
- Power Optimization Advisor
- Compilation Time Advisor
- Pin Optimization Advisor
- Arria 10 to Stratix 10 Migration Advisor

Design Space Explorer II

Use Design Space Explorer II (DSE) to find optimal settings in the Quartus Prime software.

DSE II automatically tries different combinations of netlist optimizations and advanced Quartus Prime software compiler settings, and reports the best settings for your design, based on your chosen primary optimization goal. You can try different seeds with DSE II if you are fairly close to meeting your timing or area requirements and find one seed that meets timing or area requirements. Finally, DSE II can run compilations on a remote compute farm, which shortens the timing closure process.

Related Information

About Design Space Explorer II

Send Feedback

Document Revision History

Table 7-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
2014.12.15	14.1.0	• Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. • Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Minor changes for HardCopy.
May 2013	13.0.0	Added the information about initial compilation requirements. This section was moved from the Area Optimization chapter of the Quartus Prime Handbook. Minor updates to delineate division of Timing and Area optimization chapters.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.3	Template update.
December 2010	10.0.2	Changed to new document template. No change to content.
August 2010	10.0.1	Corrected link
July 2010	10.0.0	Initial release. Chapter based on topics and text in Section III of volume 2.

Related Information

Quartus Handbook Archive

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

Reducing Compilation Time

The Analysis and Synthesis and Fitter modules consume the majority of time in a compilation. The Quartus Prime software offers several features and techniques to help reduce compilation time.

The Analysis and Synthesis module includes physical synthesis optimizations performed during synthesis, if you have turned on physical synthesis optimizations. The Fitter includes two steps, placement and routing, and also includes physical synthesis if you turned on the physical synthesis option with **Normal** or **Extra** effort levels. The **Flow Elapsed Time** section of the Compilation Report shows the duration of the Analysis and Synthesis and Fitter modules. The Fitter Messages report in the **Fitter** section of the Compilation Report displays the elapsed time for placement and routing processes.

Placement is the process of finding optimum locations for the logic in your design. Placement includes Quartus Prime pre-Fitter operations, which place dedicated logic such as clocks, PLLs, and transceiver blocks. Routing is the process of connecting the nets between the logic in your design. Finding better placement for the logic in your design requires more compilation time. Good logic placement allows you to more easily meet your timing requirements and makes your design easier to route.

```
Info: Fitter placement operations ending: elapsed time =
<days:hours:minutes:seconds>
Info: Fitter routing operations ending: elapsed time = <days:hours:minutes:seconds>
```

The Quartus Prime software displays info messages while the Fitter is running (including Placement and Routing). The Message window displays this message every hour to indicate Fitter operations are progressing normally.

```
Info: Placement optimizations have been running for 4 hour(s)
```

Compilation Time Advisor

A Compilation Time Advisor is available to help you to reduce compilation time. Run the Compilation Time Advisor on the Tools menu by pointing to **Advisors** and clicking **Compilation Time Advisor**. You can find all the compilation time optimizing techniques described in this section in the Compilation Time Advisor as well.

ISO 9001:2008 Registered

Strategies to Reduce the Overall Compilation Time

You can use the following strategies to reduce the overall time required to compile your design:

- Parallel compilation (for systems with multiple processor cores)
- Rapid Recompile and Smart Compilation reuse results from a previous compilation to reduce overall compilation time

Using Rapid Recompile

Rapid Recompile automatically reuses previous synthesis, placement, and routing results to reduce subsequent recompilation time and timing variations after making small design changes.

Figure 8-1: Rapid Recompile
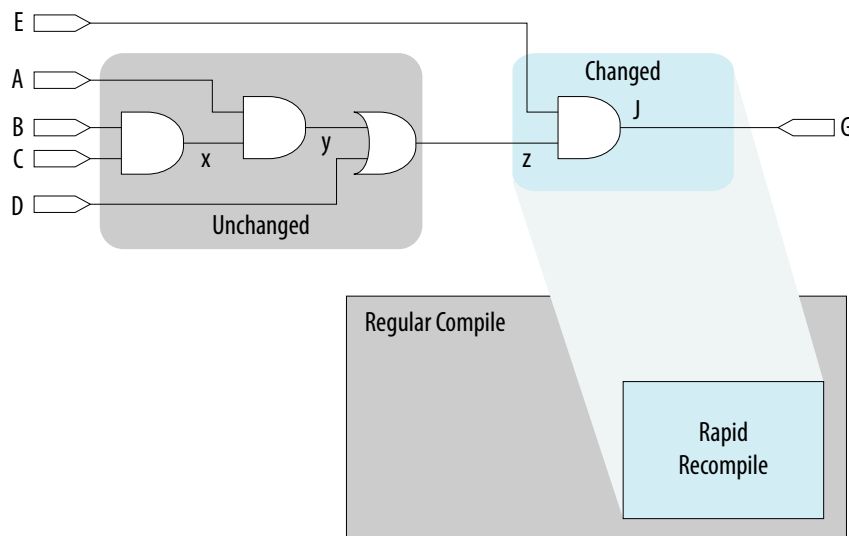
You can use Rapid Recompile to implement HDL-based functional ECO changes that affect a small subset of a large or complex design (less than 5% of total design logic), without full recompilation. Rapid Recompile can achieve up to 4x reduction in compilation time for impacted portions of the design.

To start Rapid Recompilation following an initial compilation, click **Processing** > **Start** > **Start Rapid Recompile**. Rapid Recompile implements the following type of design changes without full recompilation:

- Changes to nodes tapped by the SignalTap II Logic Analyzer
- Changes to combinational logic functions
- Changes to state machine logic (for example, new states, state transition changes)
- Changes to signal or bus latency or addition of pipeline registers
- Changes to coefficients of an adder or multiplier
- Changes register packing behavior of DSP, RAM, or I/O
- Removal of unnecessary logic
- Changes to synthesis directives

The Rapid Recompile Preservation Summary report provides detailed information about the percentage of preserved compilation results.

Figure 8-2: Rapid Recompile Preservation Summary

	Type	Achieved
	Rapid Recompile Preservation Summary	
1	Placement (by node)	33.25 % (2160 / 6497)
2	Routing (by connection)	49.93 % (14165 / 28372)

Using Parallel Compilation with Multiple Processors

The Quartus Prime software can detect the number of processors available on a computer and use multiple processors to reduce compilation time.

You can control the number of processors used during a compilation on a per user basis. The Quartus Prime software can use up to 16 processors to run algorithms in parallel and reduce compilation time. The Quartus Prime software turns on parallel compilation by default to enable the software to detect available multiple processors. You can specify the maximum number of processors that the software can use if you want to reserve some of the available processors for other tasks.

Note: Do not consider processors with Intel Hyper-Threading as more than one processor. If you have a single processor with Intel Hyper-Threading enabled, you should set the number of processors to one. Do not use the Intel Hyper-Threading feature for Quartus Prime compilations, because it can increase run times.

The Quartus Prime software does not necessarily use all the processors that you specify during a given compilation. Additionally, the software never uses more than the specified number of processors, enabling you to work on other tasks on your computer without it becoming slow or less responsive.

You can reduce the compilation time by up to 10% on systems with two processing cores and by up to 20% on systems with four cores. With certain design flows in which timing analysis runs alone, multiple processors can reduce the time required for timing analysis by an average of 10% when using two processors. This reduction can reach an average of 15% when using four processors.

You can also set the number of processors available for Quartus Prime compilation using the following Tcl command in your script:

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS <value>
```

In this case, *<value>* is an integer from 1 to 16.

If you want the Quartus Prime software to detect the number of processors and use all the processors for the compilation, include the following Tcl command in your script:

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS ALL
```

The use of multiple processors does not affect the quality of the fit. For a given Fitter seed on a specific design, the fit is exactly the same, regardless of whether the Quartus Prime software uses one processor or multiple processors. The only difference between compilations using a different number of processors is the compilation time.

The Parallel Compilation report provides detailed information about compilation using multiple processors.

Figure 8-3: Parallel Compilation Report
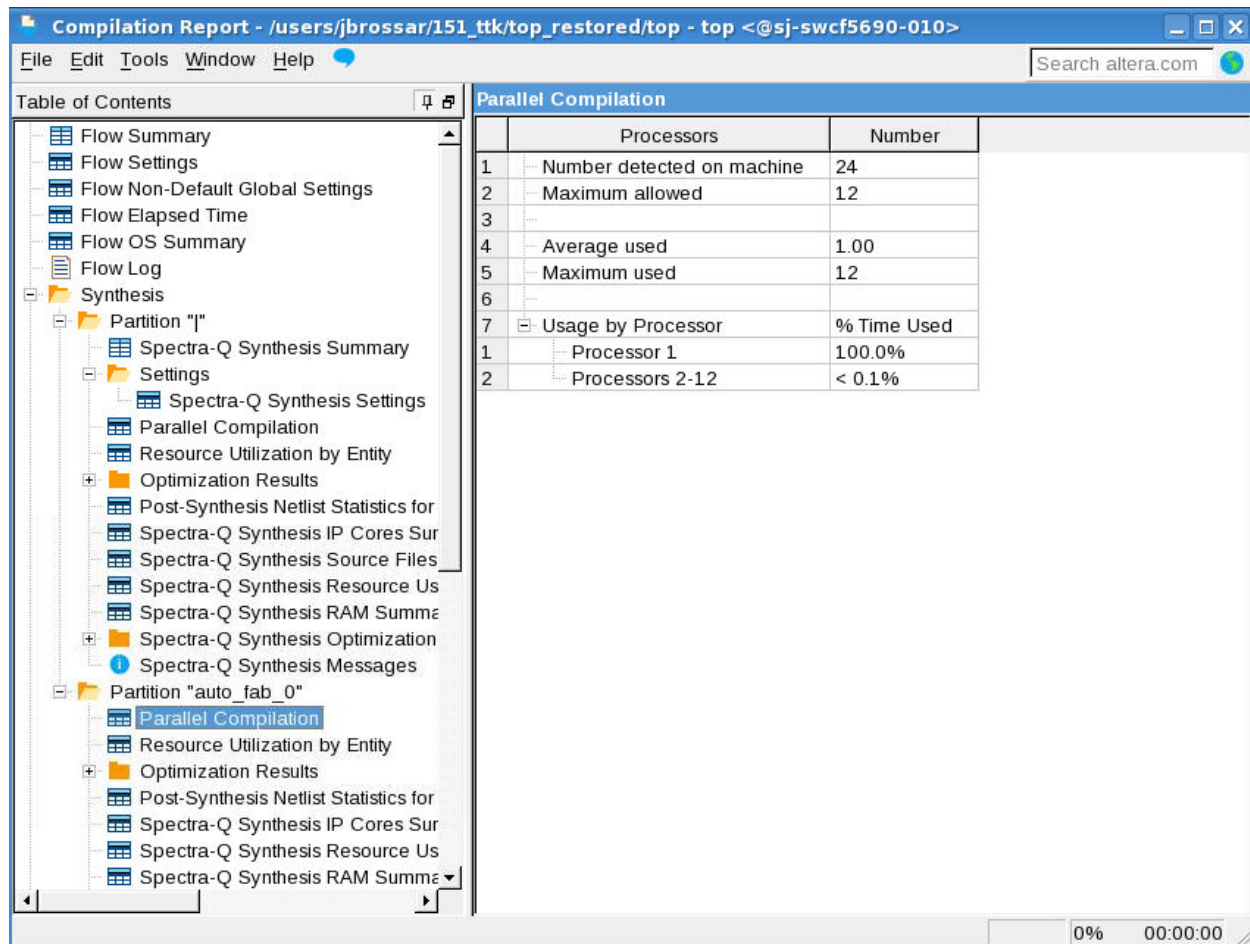
Related Information

- **Processing Page (Options Dialog Box)**
- **Compilation Process Settings Page (Settings Dialog Box)**
  For more information about how to control the number of processors used during compilation for a specific project, refer to Quartus Prime Help.

Reducing Synthesis Time and Synthesis Netlist Optimization Time

You can reduce synthesis time without affecting the Fitter time by reducing your use of netlist optimizations. For tips on reducing synthesis time when using third-party EDA synthesis tools, refer to your synthesis software's documentation.

Settings to Reduce Synthesis Time and Synthesis Netlist Optimization Time

You can use Quartus Prime integrated synthesis to synthesize and optimize HDL designs, and you can use synthesis netlist optimizations to optimize netlists that were synthesized by third-party EDA software. When using Quartus Prime Integrated Synthesis, you can also enable Physical Synthesis Optimization before performing Analysis and Synthesis. Netlist optimizations can cause the Analysis and Synthesis module to take much longer to run. Read the Analysis and Synthesis messages to determine how much

time these optimizations take. The compilation time spent in Analysis and Synthesis is usually short compared to the compilation time spent in the Fitter.

If your design meets your performance requirements without synthesis netlist optimizations, turn off the optimizations to save time. If you require synthesis netlist optimizations to meet performance, you can optimize parts of your design hierarchy separately to reduce the overall time spent in Analysis and Synthesis.

Turn off settings that are not useful. In general, if you carry over compilation settings from a previous project, evaluate all settings and keep only those that you need.

Use Appropriate Coding Style to Reduce Synthesis Time

Your HDL coding style can also affect the synthesis time. For example, if you want to infer RAM blocks from your code, you must follow the guidelines for inferring RAMs. If RAM blocks are not inferred properly, the software implements those blocks as registers.

If you are trying to infer a large memory block, the software consumes more resources in the FPGA. This can cause routing congestion and increasing compilation time significantly. If you see high routing utilizations in certain blocks, it is a good idea to review the code for such blocks.

Related Information

Recommended HDL Coding Styles

For more information about coding guidelines.

Reducing Placement Time

The time required to place a design depends on two factors: the number of ways the logic in your design can be placed in the device and the settings that control the amount of effort required to find a good placement. You can reduce the placement time in two ways:

- Change the settings for the placement algorithm.

Sometimes there is a trade-off between placement time and routing time. Routing time can increase if the placer does not run long enough to find a good placement. When you reduce placement time, ensure that it does not increase routing time and negate the overall time reduction.

Fitter Effort Setting

The highest Fitter effort setting, **Standard Fit**, requires the most runtime, but does not always yield a better result than using the default **Auto Fit**.

For designs with very tight timing requirements, both **Auto Fit** and **Standard Fit** use the maximum effort during optimization. Altera recommends using **Auto Fit** for reducing compilation time. If you are certain that your design has only easy-to-meet timing constraints, you can select **Fast Fit** for an even greater runtime savings.

Placement Effort Multiplier Settings

You can control the amount of time the Fitter spends in placement by reducing with the **Placement Effort Multiplier** option.

Click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)** and specify a value for Placement Effort Multiplier. The default is 1.0. Legal values must be greater than 0 and can be non-integer values. Numbers between 0 and 1 can reduce fitting time, but also can reduce placement quality and design performance.

Physical Synthesis Effort Settings

Physical synthesis options enable you to optimize your post-synthesis netlist and improve your timing performance. These options, which affect placement, can significantly increase compilation time.

If your design meets your performance requirements without physical synthesis options, turn them off to reduce compilation time. For example, if some or all of the physical synthesis algorithm information messages display an improvement of 0 ps, turning off physical synthesis can reduce compilation time.

You also can use the **Physical synthesis effort** setting on the **Advanced Fitter Settings** dialog box to reduce the amount of extra compilation time used by these optimizations.

The **Fast** setting directs the Quartus Prime software to use a lower level of physical synthesis optimization. Compared to the **Normal** physical synthesis effort level, using the **Fast** setting can cause a smaller increase in compilation time. However, the lower level of optimization can result in a smaller increase in design performance.

Reducing Routing Time

The time required to route a design depends on three factors: the device architecture, the placement of your design in the device, and the connectivity between different parts of your design.

The routing time is usually not a significant amount of the compilation time. If your design requires a long time to route, perform one or more of the following actions:

- Check for routing congestion.
- Turn off **Fitter Aggressive Routability Optimization**.

Identifying Routing Congestion in the Chip Planner

To identify areas of routing congestion in your design, open the Chip Planner from the Tools menu.

To view the routing congestion in the Chip Planner, double-click the **Report Routing Utilization** command in the **Tasks** list. Click **Preview** in the **Report Routing Utilization** dialog box to preview the default congestion display. Change the **Routing utilization type** to display congestion for specific resources. The default display uses dark blue for 0% congestion and red for 100%. Adjust the slider for **Threshold percentage** to change the congestion threshold level.

Even if average congestion is not very high, your design may have areas where congestion is very high in a specific type of routing. You can use the Chip Planner to identify areas of high congestion for specific interconnect types. You can change the connections in your design to reduce routing congestion. If the area with routing congestion is in a LogicLock Plus region or between LogicLock Plus regions, change or remove the LogicLock Plus regions and recompile your design. If the routing time remains the same, the time is a characteristic of your design and the placement. If the routing time decreases, consider changing the size, location, or contents of LogicLock Plus regions to reduce congestion and decrease routing time.

Sometimes, routing congestion may be a result of the HDL coding style used in your design. After you identity congested areas using the Chip Planner, review the HDL code for the blocks placed in those areas to determine whether you can reduce interconnect usage by code changes.

The Quartus Prime compilation messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, could be an indication that it might be difficult to fit your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, are likely to have increased chances of not getting a valid fit.

Related Information

Using Incremental Compilation

Analyzing and Optimizing the Design Floorplan on page 12-1
For more information about identifying areas of congested routing using the Chip Planner.

Reducing Static Timing Analysis Time

If you are performing timing-driven synthesis, the Quartus Prime software runs the TimeQuest analyzer during Analysis and Synthesis.

The Quartus Prime Fitter also runs the TimeQuest analyzer during placement and routing. If there are incorrect constraints in the Synopsys Design Constraints File (**.sdc**), the Quartus Prime software may spend unnecessary time processing constraints several times.

- If you do not specify false paths and multicycle paths in your design, the TimeQuest analyzer may analyze paths that are not relevant to your design.
- If you redefine constraints in the **.sdc** files, the TimeQuest analyzer may spend additional time processing them. To avoid this situation, look for indications that Synopsis design constraints are being redefined in the compilation messages, and update the **.sdc** file.
- Ensure that you provide the correct timing constraints to your design, because the software cannot assume design intent, such as which paths to consider as false paths or multicycle paths. When you specify these assignments correctly, the TimeQuest analyzer skips analysis for those paths, and the Fitter does not spend additional time optimizing those paths.

Setting Process Priority

It might be necessary to reduce the computing resources allocated to the compilation at the expense of increased compilation time. It can be convenient to reduce the resource allocation to the compilation with single processor machines if you must run other tasks at the same time.

Related Information
Processing Page (Options Dialog Box)
For more information about setting process priority, refer to Quartus Prime Help.

Document Revision History

Table 8-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
2014.12.15	14.1.0	- Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. - Added information about Rapid Recompile feature.
2014.08.18	14.0a10.0	Added restriction about smart compilation in Arria 10 devices.
June 2014	14.0.0	Updated format.

Date	Version	Changes
May 2013	13.0.0	Removed the "Limit to One Fitting Attempt", "Using Early Timing Estimation", "Final Placement Optimizations", and "Using Rapid Recompile" sections. Updated "Placement Effort Multiplier Settings" section. Updated "Identifying Routing Congestion in the Chip Planner" section. General editorial changes throughout the chapter.
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	• Updated "Using Parallel Compilation with Multiple Processors". • Updated "Identifying Routing Congestion in the Chip Planner". • General editorial changes throughout the chapter.
December 2010	10.1.0	• Template update. • Added details about peak and average interconnect usage. • Added new section "Reducing Static Timing Analysis Time". • Minor changes throughout chapter.
July 2010	10.0.0	Initial release.

Related Information

Quartus Handbook Archive

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

About Timing Closure and Optimization

This manual describes techniques to improve timing performance when designing for Altera® devices.

The application techniques vary between designs. Applying each technique does not always improve results. Settings and options in the Quartus Prime software have default values that provide the best trade-off between compilation time, resource utilization, and timing performance. You can adjust these settings to determine whether other settings provide better results for your design.

Initial Compilation: Optional Fitter Settings

The Fitter offers many optional settings; however, this section focuses on the optional timing-optimization related Fitter settings only, which are the **Optimize Hold Timing**, **Optimize Multi-Corner Timing,** and **Fitter Aggressive Routability Optimization** settings.

Caution: The settings required to optimize different designs could be different. The group of settings that work best for one design may not produce the best result for another design.

Related Information

Advanced Fitter Setting Dialog Box online help
For scripting and device family support information of the **Optimize Hold Timing** and **Optimize Multi-Corner Timing** settings

Optimize Hold Timing

The **Optimize Hold Timing** option directs the Quartus Prime software to optimize minimum delay timing constraints. By default, the Quartus Prime software optimizes hold timing for all paths for designs for supported devices. By default, the Quartus Prime software optimizes hold timing only for I/O paths and minimum t_{PD} paths for older devices.

When you turn on **Optimize Hold Timing** in the **Advanced Fitter Settings** dialog box, the Quartus Prime software adds delay to paths to ensure that your design meets the minimum delay requirements. If you select **I/O Paths and Minimum TPD Paths**, the Fitter works to meet the following criteria:

- Hold times (t_H) from the device input pins to the registers
- Minimum delays from I/O pins to I/O registers or from I/O registers to I/O pins
- Minimum clock-to-out time (t_{CO}) from registers to output pins

If you select **All Paths**, the Fitter also works to meet hold requirements from registers to registers, as highlighted in blue in the figure, in which a derived clock generated with logic causes a hold time problem on another register.

Figure 9-1: Optimize Hold Timing Option Fixing an Internal Hold Time Violation
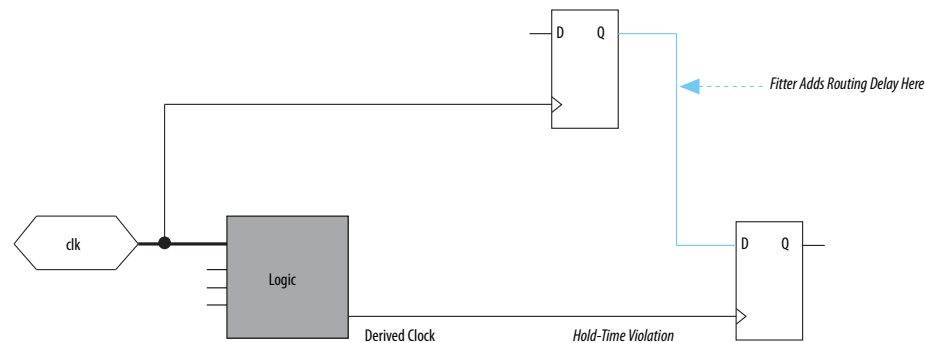
However, if your design still has internal hold time violations between registers, correct the violations by manually adding some delays by instantiating LCELL primitives, or by making changes to your design, such as using a clock enable signal instead of a derived or gated clock.

Related Information
Recommended Design Practices documentation
For design practices that help to eliminate internal hold time violations

Optimize Multi-Corner Timing

Due to process variations and changes in operating conditions, delays on some paths can be significantly smaller than those in the slow corner timing model. This can result in hold time violations on those paths, and in rare cases, additional setup time violations.

Also, because of the small process geometries of newer device families, the slowest circuit performance of designs targeting these devices does not necessarily occur at the highest operating temperature. The temperature at which the circuit is slowest depends on the selected device, the design, and the compilation results. Therefore, the Quartus Prime software provides newer device families with three different timing corners—Slow 85°C corner, Slow 0°C corner, and Fast 0°C corner. For other device families, two timing corners are available—Fast 0°C and Slow 85°C corner.

The **Optimize multi-corner timing** option directs the Fitter to consider all corner timing delays, including both fast-corner timing and slow-corner timing, during optimization to meet timing requirements at all process corners and operating conditions. By default, this option is on, and the Fitter optimizes designs considering multi-corner delays in addition to slow-corner delays, for example, from the fast-corner timing model, which is based on the fastest manufactured device, operating under high-voltage conditions

The **Optimize multi-corner timing** option helps to create a design implementation that is more robust across process, temperature, and voltage variations. Turning on this option increases compilation time by approximately 10%.

When this option is off, the Fitter optimizes designs considering only slow-corner delays from the slow-corner timing model (slowest manufactured device for a given speed grade, operating in low-voltage conditions).

Fitter Aggressive Routability Optimization

The **Fitter Aggressive Routability Optimizations** logic option allows you to specify whether the Fitter aggressively optimizes for routability. Performing aggressive routability optimizations may decrease design speed, but may also reduce routing wire usage and routing time.

This option is useful if routing resources are resulting in no-fit errors, and you want to reduce routing wire use.

The table lists the settings for the **Fitter Aggressive Routability Optimizations** logic option.

Table 9-1: Fitter Aggressive Routability Optimizations Logic Option Settings

Settings	Description
Always	The Fitter always performs aggressive routability optimizations. If you set the Fitter Aggressive Routability Optimizations logic option to Always, reducing wire utilization may affect the performance of your design.
Never	The Fitter never performs aggressive routability optimizations. If improving timing is more important than reducing wire usage, then set this option to Automatically or Never.
Automatically	The Fitter performs aggressive routability optimizations automatically, based on the routability and timing requirements of the design. If improving timing is more important than reducing wire usage, then set this option to Automatically or Never.

Design Analysis

The initial compilation establishes whether the design achieves a successful fit and meets the specified timing requirements. This section describes how to analyze your design results in the Quartus Prime software.

Ignored Timing Constraints

The Quartus Prime software ignores illegal, obsolete, and conflicting constraints.

You can view a list of ignored constraints by clicking **Report Ignored Constraints** in the Reports menu in the TimeQuest GUI or by typing the following command to generate a list of ignored timing constraints:

```
report_sdc -ignored -panel_name "Ignored Constraints"
```

You should analyze any constraints that the Quartus Prime software ignores. If necessary, correct the constraints and recompile your design before proceeding with design optimization.

You can view a list of ignored assignment in the **Ignored Assignment Report** generated by the Fitter.

Related Information

- **Quartus Prime TimeQuest Timing Analyzer documentation**
  For more information about the `report_sdc` command and its options
- **Fitter Summary Reports online help**

I/O Timing (Including t$_{PD}$)

TimeQuest analyzer supports the Synopsys Design Constraints (SDC) format for constraining your design. When using the TimeQuest analyzer for timing analysis, use the `set_input_delay` constraint to specify the data arrival time at an input port with respect to a given clock. For output ports, use the `set_output_delay` command to specify the data arrival time at an output port's receiver with respect to a given clock. You can use the `report_timing` Tcl command to generate the I/O timing reports.

The I/O paths that do not meet the required timing performance are reported as having negative slack and are highlighted in red in the TimeQuest analyzer **Report** pane. In cases where you do not apply an explicit I/O timing constraint to an I/O pin, the Quartus Prime timing analysis software still reports the **Actual** number, which is the timing number that must be met for that timing parameter when the device runs in your system.

Related Information
Quartus Prime TimeQuest Timing Analyzer documentation
Information about how timing numbers are calculated

Register-to-Register Timing

Timing Analysis with the TimeQuest Timing Analyzer

Analyze all valid register-to-register paths by using the appropriate constraints in the TimeQuest analyzer. To view all timing summaries, run the **Report All Summaries** command by double-clicking **Report All Summaries** in the Tasks pane in the TimeQuest analyzer.

If any clock domains have failing paths (highlighted in red in the Report panel), right-click the Clock Name listed in the Clocks Summary panel and go to **Report Timing** to get more details. Your design meets timing requirements when you do not have negative slack on any register-to-register path on any of the clock domains.

When timing requirements are not met, a report on the failed paths (highlighted in red) can uncover more detail.

When you select a path listed in the TimeQuest **Report Timing** pane, the tabs in the corresponding path detail pane show a path summary of source and destination registers and their timing, statistics about the path delay, detailed information about the complete data path with all nodes in the path, and the waveforms of the relevant signals. The **Extra Fitter Information** tab will show a Graphical Data Path of where the offending path lies on the physical device. This can reveal whether the timing failure may be distance related, due to the source and destination node being too close or too far. The Chip Planner can also be used to investigate the physical layout of a failing path in more detail. To locate a selected path in the Chip Planner, right-click a node, point to **Locate**, and select **Locate in Chip Planner**. The Chip Planner appears with the path highlighted. Use this to show fanout, fanin, routing congestion, and region assignments information, and to determine whether those factors might be contributing to the timing critical path. Additionally, if you know that a path is not a valid path, you can set it to be a false path using the shortcut menu.

The **Data Path** tab can also be useful for determining contributions to timing critical paths. The **Data Path** tab shows details of the paths that the clock and data took to get from source to destination nodes, and the time it took on an incremental and cumulative basis. It also provides information about the routing types and elements used, and their locations.

To view the path details of any selected path, click the **Data Path** tab in the path details pane. The **Data Path** tab displays the details of the Data Arrival Path, as well as the Data Required Path.

The **Waveform** tab will show the slack relationship between arrival data and required data. This could be useful for determining how close or far off the path is from meeting timing.

To aid in timing debug, the RTL Viewer or Technology Map Viewer allow you to see schematic representations of your design. These viewers allow you to view a gate-level or technology-mapped representation of your design netlist. By providing a view of the path from source and destination nodes, the viewers can help identify areas in a design that may benefit from reducing the number of logic levels between the nodes. To locate a timing path in one of the viewers, right-click a path in the report, point to **Locate**, and click **Locate in RTL Viewer** or **Locate in Technology Map Viewer**.

Related Information

- **Quartus Prime TimeQuest Timing Analyzer documentation**
  Information about how timing analysis results are calculated
- **Analyzing Designs with Quartus Prime Netlist Viewers documentation**

Tips for Analyzing Failing Paths

When you are analyzing failing paths, examine the reports and waveforms to determine if the correct constraints are being applied, and add timing exceptions as appropriate. A multicycle constraint relaxes setup or hold relationships by the specified number of clock cycles. A false path constraint specifies paths that can be ignored during timing analysis. Both constraints allow the Fitter to work harder on affected paths.

Focus on improving the paths that show the worst slack. The Fitter works hardest on paths with the worst slack. If you fix these paths, the Fitter might be able to improve the other failing timing paths in the design.

Check for particular nodes that appear in many failing paths. These nodes will appear in a timing report panel at the top of the list, along with their minimum slacks. Look for paths that have common source registers, destination registers, or common intermediate combinational nodes. In some cases, the registers might not be identical, but are part of the same bus.

In the timing analysis report panels, clicking on the **From** or **To** column headings can help to sort the paths by the source or destination registers. Clicking first on **From**, then on **To**, uses the registers in the **To** column as the primary sort and the registers in the **From** column as the secondary sort. If you see common nodes, these nodes indicate areas of your design that might be improved through source code changes or Quartus Prime optimization settings. Constraining the placement for just one of the paths might decrease the timing performance for other paths by moving the common node further away in the device.

Related Information
Design Evaluation for Timing Closure on page 9-26

Tips for Analyzing Failing Clock Paths that Cross Clock Domains

When analyzing clock path failures, check whether these paths cross two clock domains. This is the case if the **From Clock** and **To Clock** in the timing analysis report are different.

Figure 9-2: Different Value in From Clock and To Clock Field

Setup Transfers					
	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	clkin	clkin	21	0	0	0
2	clkin	clkout	false path	0	0	0
3	clkout	clkout	31	0	0	0

There can also be paths that involve a different clock in the middle of the path, even if the source and destination register clock are the same.

When you run Report Timing on your design, the report shows the launch clock and latch clock for each failing path. Check whether these failing paths between these clock domains should be analyzed synchronously. If the failing paths are not to be analyzed synchronously, they must be set as false paths. Also check the relationship between the launch clock and latch clock to make sure it is realistic and what you expect from your knowledge of the design. For example, the path can start at a rising edge and end at a falling edge, which reduces the setup relationship by one half clock cycle.

Review the clock skew reported in the Timing Report. A large skew may indicate a problem in your design, such as a gated clock or a problem in the physical layout (for example, a clock using local routing instead of dedicated clock routing). When you have made sure the paths are analyzed synchronously and that there is no large skew on the path, and that the constraints are correct, you can analyze the data path. These steps help you fine tune your constraints for paths across clock domains to ensure you get an accurate timing report.

Check if the PLL phase shift is reducing the setup requirement. You might be able to adjust this using PLL parameters and settings.

Paths that cross clock domains are generally protected with synchronization logic (for example, FIFOs or double-data synchronization registers) to allow asynchronous interaction between the two clock domains. In such cases, you can ignore the timing paths between registers in the two clock domains while running timing analysis, even if the clocks are related.

The Fitter attempts to optimize all failing timing paths. If there are paths that can be ignored for optimization and timing analysis, but the paths do not have constraints that instruct the Fitter to ignore them, the Fitter tries to optimize those paths as well. In some cases, optimizing unnecessary paths can prevent the Fitter from meeting the timing requirements on timing paths that are critical to the design. It is beneficial to specify all paths that can be ignored by setting false path constraints on them, so that the Fitter can put more effort into the paths that must meet their timing requirements instead of optimizing paths that can be ignored.

Related Information

Quartus Prime TimeQuest Timing Analyzer
Details about how to ignore timing paths that cross clock domains

Tips for Analyzing Paths from/to the Source and Destination of Critical Path

When analyzing the failing paths in a design, it is often helpful to get a fuller picture of the many interactions the fitter may be working on around the paths. To understand what may be pulling on a critical path, the following `report_timing` command can be useful.

In the project directory, run the `report_timing` command, shown in the example below, in a **.tcl** file to analyze the nodes in a critical path.

```
set wrst_src <insert_source_of_worst_path_here>
set wrst_dst <insert_destination_of_worst_path_here>
report_timing -setup -npaths 50 -detail path_only -from $wrst_src \
-panel_name "Worst Path||wrst_src -> *"
report_timing -setup -npaths 50 -detail path_only -to $wrst_dst \
-panel_name "Worst Path||* -> wrst_dst"
report_timing -setup -npaths 50 -detail path_only -to $wrst_src \
-panel_name "Worst Path||* -> wrst_src"
report_timing -setup -npaths 50 -detail path_only -from $wrst_dst \
-panel_name "Worst Path||wrst_dst -> *"
```

Copy the node names from the **From Node** and **To Node** columns of the worst path into the first two variables, and then in the TimeQuest timing analyzer, in the Script menu, source the **.tcl** script.

In the resulting timing panel, timing failed paths (highlighted in red) can be located in the Chip Planner, where information such as distance between the nodes and large fanouts can be viewed.

The figure shows a simplified example of what these reports analyzed.

Figure 9-3: Timing Report
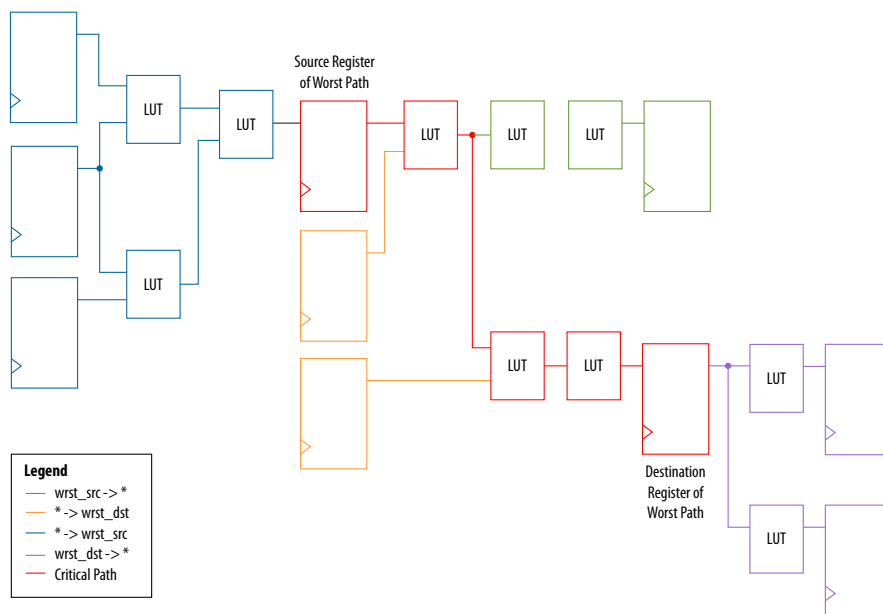
The critical path of the design is in red. The script analyzes the path between the worst source and destination registers. The first `report_timing` command analyzes other path that the source is driving, as shown in green. The second `report_timing` command analyzes the critical path and other path going to the destination, shown in yellow. These commands report everything inside these two endpoints that are pulling them in different directions. The last two `report_timing` commands show everything outside of

Send Feedback

the endpoints pulling them in other directions. If any of these reports have slacks near the critical path, then the Fitter is balancing these paths with the critical path, trying to achieve the best slack. The figure is quite simple compared to the critical path in most designs, but it is easy to see how this can get very complicated quickly.

Tips for Locating Multiple Paths to the Chip Planner

The Chip Planner can be used as a visual aid in locating timing critical paths. To view these paths from timing reports, do the following:

1. Run `report_timing` to show multiple paths.
2. Select multiple rows of the timing report.
3. Right-click, select **Locate Path**, and then click **Chip Planner**.
4. The **Locate History** window in the Chip Planner displays the selected paths and the worst path.
5. Double-click **Locate Paths** to show all paths at once, or select individual paths to view the path in the Chip Planner.

    This will show whether timing failures may be due to large distances between the nodes or large fanouts.

Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles

Many designs have the same critical paths show up after each compile, but some suffer from having critical paths bounce around between different hierarchies, changing with each compile.

This could happen in high speed designs where many register to register paths have very little slack. Different placements can then result in timing failures in the marginal paths. In designs like this, create a **TQ_critical_paths.tcl** script in the project directory. For a given compile, view the critical paths and then write a generic `report_timing` command to capture those paths. For example, if several paths fail in a low-level hierarchy, you can add the following command:

```
report_timing –setup –npaths 50 –detail path_only \
-to "main_system: main_system_inst|app_cpu:cpu|*" \
-panel_name "Critical Paths||s: * -> app_cpu"
```

If there is a specific path, such as a bit of a state-machine going to other `*count_sync*` registers, you can add a command as shown by the following:

```
report_timing –setup –npaths 50 –detail path_only \
-from "main_system: main_system_inst|egress_count_sm:egress_inst|update" \
-to "*count_sync*" –panel_name "Critical Paths||s: egress_sm|update -> count_sync"
```

This file can be sourced in the TimeQuest timing analyzer after every compilation, and new `report_timing` commands can be added as new critical paths appear. This helps you monitor paths that consistently fail and paths that are only marginal, so you can prioritize effectively.

Global Routing Resources

Global routing resources are designed to distribute high fan-out, low-skew signals (such as clocks) without consuming regular routing resources. Depending on the device, these resources can span the entire chip, or some smaller portion, such as a quadrant. The Quartus Prime software attempts to assign signals to global routing resources automatically, but you might be able to make more suitable assignments manually.

For details about the number and types of global routing resources available, refer to the relevant device handbook.

Check the global signal utilization in your design to ensure that the appropriate signals have been placed on the global routing resources. In the Compilation Report, open the Fitter report and click **Resource Section**. Analyze the Global & Other Fast Signals and Non-Global High Fan-out Signals reports to determine whether any changes are required.

You might be able to reduce skew for high fan-out signals by placing them on global routing resources. Conversely, you can reduce the insertion delay of low fan-out signals by removing them from global routing resources. Doing so can improve clock enable timing and control signal recovery/removal timing, but increases clock skew. Use the **Global Signal** setting in the Assignment Editor to control global routing resources.

Optimizing Timing (LUT-Based Devices)

You can use the following guidelines if your design does not meet its timing requirements:

Debugging Timing Failures in the TimeQuest Analyzer

A **Report Timing Closure Recommendations** task is available in the Custom Reports section of the **Tasks** pane of the TimeQuest analyzer. Use this report to get more information and help on the failing paths in your design.

When you run the **Report Timing Closure Recommendations** task, you get specific recommendations about failing paths in your design and changes that you can make to potentially fix the failing paths.

Selecting the **Report Timing Closure Recommendations** task opens the **Report Timing Closure Recommendations** dialog box.

From the **Report Timing Closure Recommendations** dialog box, you can select paths based on the clock domain, filter by nodes on path, and choose the number of paths to analyze.

After running the **Report Timing Closure Recommendations** task in the TimeQuest analyzer, examine the reports in the **Report Timing Closure Recommendations** folder in the **Report** pane of the TimeQuest analyzer GUI. Each recommendation has star symbols (*) associated with it. Recommendations with more stars are more likely to help you close timing on your design.

The reports give you the most probable causes of failure for each path being analyzed. The reports are organized into sections, depending on the type of issues found in the design, such as large clock skew, restricted optimizations, unbalanced logic, skipped optimizations, coding style that has too many levels of logic between registers, or region or partition constraints specific to your project.

You will see recommendations that may help you fix the failing paths. For detailed analysis of the critical paths, run the `report_timing` command on specified paths. In the **Extra Fitter Information** tab of the Path report panel, you will also see detailed Fitter-related information that may help you visualize the issue and take the appropriate action if your constraints cause a specific placement.

Related Information

Report Timing Closure Recommendations Dialog Box online help

Timing Optimization Advisor

While the TimeQuest **Report Timing Closure Recommendations** task gives specific recommendations to fix failing paths, the **Timing Optimization Advisor** gives more general recommendations to improve timing performance for a design.

The **Timing Optimization Advisor** guides you in making settings that optimize your design to meet your timing requirements. To run the Timing Optimization Advisor, on the Tools menu, point to **Advisors** and click **Timing Optimization Advisor**. This advisor describes many of the suggestions made in this section.

When you open the Timing Optimization Advisor after compilation, you can find recommendations to improve the timing performance of your design. Some of the recommendations in these advisors can contradict each other. Altera recommends evaluating these options and choosing the settings that best suit the given requirements.

The example shows the Timing Optimization Advisor after compiling a design that meets its frequency requirements, but requires setting changes to improve the timing.

Figure 9-4: Timing Optimization Advisor
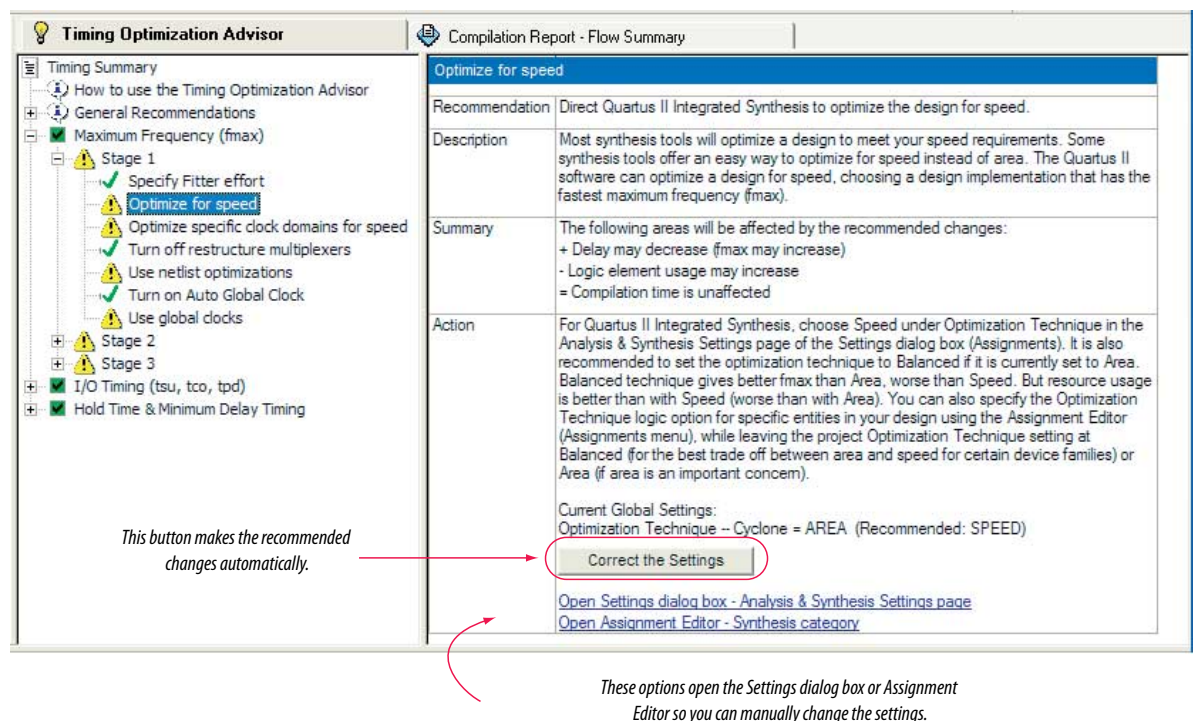
When you expand one of the categories in the Timing Optimization Advisor, such as **Maximum Frequency (fmax)** or **I/O Timing (tsu, tco, tpd)**, the recommendations are divided into stages. The stages show the order in which to apply the recommended settings. The first stage contains the options that are easiest to change, make the least drastic changes to your design optimization, and have the least effect on compilation time. Icons indicate whether each recommended setting has been made in the current project. In the figure, the checkmark icons in the list of recommendations for Stage 1 indicate recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icons indicate general suggestions. For these entries, the advisor

does not report whether these recommendations were followed, but instead explains how you can achieve better performance. For a legend that provides more information for each icon, refer to the "How to use" page in the Timing Optimization Advisor.

There is a link from each recommendation to the appropriate location in the Quartus Prime GUI where you can change the settings. For example, consider the **Synthesis Netlist Optimizations** page of the **Settings** dialog box or the **Global Signals category** in the Assignment Editor. This approach provides the most control over which settings are made and helps you learn about the settings in the software. In some cases, you can also use the **Correct the Settings** button to automatically make the suggested change to global settings.

For some entries in the Timing Optimization Advisor, a button appears that allows you to further analyze your design and gives you more information. The advisor provides a table with the clocks in the design and indicates whether they have been assigned a timing constraint.

I/O Timing Optimization

This stage of design optimization focuses on I/O timing. Ensure that you have made the appropriate assignments described in the "Initial Compilation: Required Settings" section in the **Design Optimization Overview** chapter of the *Quartus Prime Handbook*. You must also ensure that resource utilization is satisfactory before proceeding with I/O timing optimization. The suggestions provided in this section are applicable to all Altera FPGA families and to the MAX II family of CPLDs.

Because changes to the I/O paths affect the internal register-to-register timing, complete this stage before proceeding to the register-to-register timing optimization stage as described in **Register-to-Register Timing Optimization Techniques (LUT-Based Devices)**.

The options presented in this section address how to improve I/O timing, including the setup delay (t_{SU}), hold time (t_H), and clock-to-output (t_{CO}) parameters.

Improving Setup and Clock-to-Output Times Summary

The table lists the recommended order in which to use techniques to reduce t_{SU} and t_{CO} times. "Yes" indicates which timing parameters are affected by each technique. Reducing t_{SU} times increases hold (t_H) times.

Table 9-2: Improving Setup and Clock-to-Output Times

Technique	Affects t_{SU}	Affects t_{CO}
Ensure that the appropriate constraints are set for the failing I/Os (refer to the "Initial Compilation: Required Settings" section in the **Design Optimization Overview**chapter of the *Quartus Prime Handbook.)*	Yes	Yes
Use timing-driven compilation for I/O (**Fast Input, Output, and Output Enable Registers**)	Yes	Yes
Use fast input register (**Programmable Delays**)	Yes	N/A
Use fast output register, fast output enable register, and fast OCT register (**Programmable Delays**)	N/A	Yes
Decrease the value of **Input Delay from Pin to Input Register** or set **Decrease Input Delay to Input Register = ON**	Yes	N/A

Technique	Affects t_{SU}	Affects t_{CO}
Decrease the value of **Input Delay from Pin to Internal Cells** or set **Decrease Input Delay to Internal Cells = ON**	Yes	N/A
Decrease the value of **Delay from Output Register to Output Pin** or set **Increase Delay to Output Pin = OFF** (**Fast Input, Output, and Output Enable Registers**)	N/A	Yes
Increase the value of **Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations** (**Fast Input, Output, and Output Enable Registers**)	Yes	N/A
Use PLLs to shift clock edges (**Use PLLs to Shift Clock Edges**)	Yes	Yes
Use the Fast Regional Clock (**Change How Hold Times are Optimized for MAX II Devices**)	N/A	Yes
For MAX II or MAX V family devices, set **Guarantee I/O Paths Have Zero Hold Time at Fast Corner** to **OFF**, or **When T$_{SU}$ and T$_{PD}$ Constraints Permit** (**Change How Hold Times are Optimized for MAX II Devices**)	Yes	N/A
Increase the value of **Delay to output enable pin** or set **Increase delay to output enable pin** (**Use PLLs to Shift Clock Edges**)	N/A	Yes

Note to table :

1. These options may not apply to all device families.

Timing-Driven Compilation

This option moves registers into I/O elements if required to meet t_{SU} or t_{CO} assignments, duplicating the register if necessary (as in the case in which a register fans out to multiple output locations). This option is turned on by default and is a global setting. The option does not apply to MAX II series devices because they do not contain I/O registers.

The **Optimize IOC Register Placement for Timing** option affects only pins that have a t_{SU} or t_{CO} requirement. Using the I/O register is possible only if the register directly feeds a pin or is fed directly by a pin. This setting does not affect registers with any of the following characteristics:

- Have combinational logic between the register and the pin
- Are part of a carry or cascade chain
- Have an overriding location assignment
- Use the asynchronous load port and the value is not 1 (in device families where the port is available)

Registers with the characteristics listed are optimized using the regular Quartus Prime Fitter optimizations.

Related Information

Optimize IOC Register Placement for Timing Logic Option online help

Fast Input, Output, and Output Enable Registers

Normally, with correct timing assignments, the Fitter already places the I/O registers in the correct I/O cell or in the core, to meet the performance requirement. However, you can place individual registers in I/O cells manually by making fast I/O assignments with the Assignment Editor.

For more information about the **Fast Input Register** option, **Fast Output Register** option, **Fast Output Enable Register** option, and **Fast OCT (on-chip termination) Register** option, refer to Quartus Prime Help.

In MAX II series devices, which have no I/O registers, these assignments lock the register into the LAB adjacent to the I/O pin if there is a pin location assignment for that I/O pin.

If the fast I/O setting is on, the register is always placed in the I/O element. If the fast I/O setting is off, the register is never placed in the I/O element. This is true even if the **Optimize IOC Register Placement for Timing** option is turned on. If there is no fast I/O assignment, the Quartus Prime software determines whether to place registers in I/O elements if the **Optimize IOC Register Placement for Timing** option is turned on.

You can also use the four fast I/O options (**Fast Input Register**, **Fast Output Register**, **Fast Output Enable Register**, and **Fast OCT Register**) to override the location of a register that is in a LogicLock Plus region and force it into an I/O cell. If you apply this assignment to a register that feeds multiple pins, the register is duplicated and placed in all relevant I/O elements. In MAX II series devices, the register is duplicated and placed in each distinct LAB location that is next to an I/O pin with a pin location assignment.

Programmable Delays

You can use various programmable delay options to minimize the t_{SU} and t_{CO} times. For Arria, Cyclone, MAX II, MAX V, and Stratix series devices, the Quartus Prime software automatically adjusts the applicable programmable delays to help meet timing requirements. Programmable delays are advanced options to use only after you compile a project, check the I/O timing, and determine that the timing is unsatisfactory. For detailed information about the effect of these options, refer to the device family handbook or data sheet.

After you have made a programmable delay assignment and compiled the design, you can view the implemented delay values for every delay chain for every I/O pin in the **Delay Chain Summary** section of the Compilation Report.

You can assign programmable delay options to supported nodes with the Assignment Editor. You can also view and modify the delay chain setting for the target device with the Chip Planner and Resource Property Editor. When you use the Resource Property Editor to make changes after performing a full compilation, recompiling the entire design is not necessary; you can save changes directly to the netlist. Because these changes are made directly to the netlist, the changes are not made again automatically when you recompile the design. The change management features allow you to reapply the changes on subsequent compilations.

Although the programmable delays in newer devices are user-controllable, Altera recommends their use for advanced users only. However, the Quartus Prime software might use the programmable delays internally during the Fitter phase.

For details about the programmable delay logic options available for Altera devices, refer to the following Quartus Prime Help topics:

Input Delay from Pin to Input Register logic option

Input Delay from Pin to Internal Cells logic option

Output Enable Pin Delay logic option

Delay from Output Register to Output Pin logic option

Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations logic option

Use PLLs to Shift Clock Edges

Using a PLL typically improves I/O timing automatically. If the timing requirements are still not met, most devices allow the PLL output to be phase shifted to change the I/O timing. Shifting the clock backwards gives a better t_H at the expense of t_{SU}, while shifting it forward gives a better t_{SU} at the expense of t_H. You can use this technique only in devices that offer PLLs with the phase shift option.

Figure 9-5: Shift Clock Edges Forward to Improve t_{SU} at the Expense of t_H
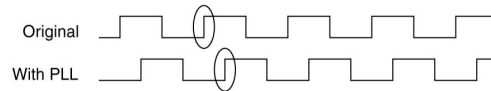
You can achieve the same type of effect in certain devices by using the programmable delay called **Input Delay from Dual Purpose Clock Pin to Fan-Out Destinations**.

Related Information

Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations Logic Option online help

Use Fast Regional Clock Networks and Regional Clocks Networks

Altera devices have a variety of hierarchical clock structures. These include dedicated global clock networks, regional clock networks, fast regional clock networks, and periphery clock networks. The available resources differ between the various Altera device families.

For the number of clocking resources available in your target device, refer to the appropriate device handbook.

In general, fast regional clocks have less delay to I/O elements than regional and global clocks, and are used for high fan-out control signals. Regional clocks provide the lowest clock delay and skew for logic contained in a single quadrant. Placing clocks on these low-skew and low-delay clock nets provides better t_{CO} performance.

Spine Clock Limitations

Global clock networks, regional clock networks, and periphery clock networks have an additional level of clock hierarchy known as spine clocks. Spine clocks drive the final row and column clocks to their registers; thus, the clock to every register in the chip is reached through spine clocks. Spine clocks are not directly user controllable.

If your project has high clock routing demands, due to limitations in the Quartus Prime software, you may see spine clock errors. These errors are often seen with designs using multiple memory interfaces and high-speed serial interface (HSSI) channels (especially PMA Direct mode).

To reduce these spine clock errors, you can constrain your design to better use your regional clock resources using the following techniques:

- If your design does not use LogicLock Plus regions, or if the LogicLock Plus regions are not aligned to your clock region boundaries, create additional LogicLock Plus regions and further constrain your logic.

Note: Register packing, a Fitter optimization option, may ignore LogicLock Plus regions. If this occurs, disable register packing for specific instances through the Quartus Prime Assignment Editor.

- Some periphery features may ignore LogicLock Plus region assignments. When this happens, the global promotion process may not function properly. To ensure that the global promotion process uses the correct locations, assign specific pins to the I/Os using these periphery features.
- By default, some IP MegaCore functions apply a global signal assignment with a value of dual-regional clock. If you constrain your logic to a regional clock region and set the global signal assignment to **Regional** instead of **Dual-Regional**, you can reduce clock resource contention.

Change How Hold Times are Optimized for MAX II Devices

For MAX II devices, you can use the **Guarantee I/O Paths Have Zero Hold Time at Fast Corner** option to control how hold time is optimized by the Quartus Prime software.

Register-to-Register Timing Optimization Techniques (LUT-Based Devices)

The next stage of design optimization is to improve register-to-register (f_{MAX}) timing. The following sections provide available options if the performance requirements are not achieved after compilation.

Coding style affects the performance of your design to a greater extent than other changes in settings. Always evaluate your code and make sure to use synchronous design practices.

Note: When using the TimeQuest analyzer, register-to-register timing optimization is the same as maximizing the slack on the clock domains in your design. You can use the techniques described in this section to improve the slack on different timing paths in your design.

Before optimizing your design, understand the structure of your design as well as the type of logic affected by each optimization. An optimization can decrease performance if the optimization does not benefit your logic structure.

Related Information
Recommended Design Practices documentation
Details about synchronous design practices and coding styles

Optimize Source Code

In many cases, optimizing the design's source code can have a very significant effect on your design performance. In fact, optimizing your source code is typically the most effective technique for improving the quality of your results and is often a better choice than using LogicLock Plus or location assignments.

Be aware of the number of logic levels needed to implement your logic while you are coding. Too many levels of logic between registers could result in critical paths failing timing. Try restructuring the design to use pipelining or more efficient coding techniques. Also, try limiting high fan-out signals in the source code. When possible, duplicate and pipeline control signals. Make sure the duplicate registers are protected by a preserve attribute, to avoid merging during synthesis.

If the critical path in your design involves memory or DSP functions, check whether you have code blocks in your design that describe memory or functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to cause these functions to be placed into high-performance dedicated memory or resources in the target device. When using RAM/DSP blocks, enable the optional input and output registers.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Quartus Prime software, you can check the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including state

encoding for each state machine that was recognized during compilation. If your state machine is not recognized, you might have to change your source code to enable it to be recognized.

Related Information

- **Recommended HDL Coding Styles documentation**
  Coding style guidelines including examples of HDL code for inferring memory, functions, guidelines, and sample HDL code for state machines
- **AN 584: Timing Closure Methodology for Advanced FPGA Designs application note.**

Improving Register-to-Register Timing Summary

The choice of options and settings to improve the timing margin (slack) or to improve register-to-register timing depends on the failing paths in the design. To achieve the results that best approximate your performance requirements, apply the following techniques and compile the design after each step:

1. Ensure that your timing assignments are complete and correct. For details, refer to the "Initial Compilation: Required Settings" section in the **Design Optimization Overview** chapter of the *Quartus Prime Handbook*.
2. Ensure that you have reviewed all warning messages from your initial compilation and check for ignored timing assignments.
3. Apply netlist synthesis optimization options.
4. To optimize for speed, apply the following synthesis options:

   - **Optimize Synthesis for Speed, Not Area**
   - **Flatten the Hierarchy During Synthesis**
   - **Set the Synthesis Effort to High**
   - **Change State Machine Encoding**
   - **Prevent Shift Register Inference**
   - **Use Other Synthesis Options Available in Your Synthesis Tool**

5. To optimize for performance using physical synthesis, apply the following options:

   - Perform physical synthesis for combinational logic
   - Perform automatic asynchronous signal pipelining
   - Perform register duplication
   - Perform register retiming
   - Perform logic to memory mapping

6. Try different Fitter seeds. If there are very few paths that are failing by small negative slack, then you can try with a different seed to see if there is a fit that meets constraints in the Fitter seed noise.

   Note: Omit this step if a large number of critical paths are failing or if the paths are failing badly.

7. To control placement, make LogicLock Plus assignments.
8. Make design source code modifications to fix areas of the design that are still failing timing requirements by significant amounts.
9. Make location assignments, or as a last resort, perform manual placement by back-annotating the design.

   You can use Design Space Explorer II (DSE) to automate the process of running several different compilations with different settings.

   If these techniques do not achieve performance requirements, additional design source code modifications might be required.

Related Information
Design Space Explorer II online help

Physical Synthesis Optimizations

The Quartus Prime software offers physical synthesis optimizations that can help improve the performance of many designs regardless of the synthesis tool used. Physical synthesis optimizations can be applied both during synthesis and during fitting.

Physical synthesis optimizations that occur during the synthesis stage of the Quartus Prime compilation operate either on the output from another EDA synthesis tool or as an intermediate step in Quartus Prime integrated synthesis. These optimizations make changes to the synthesis netlist to improve either area or speed, depending on your selected optimization technique and effort level.

To view and modify the synthesis netlist optimization options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

If you use a third-party EDA synthesis tool and want to determine if the Quartus Prime software can remap the circuit to improve performance, you can use the **Perform WYSIWYG Primitive Resynthesis** option. This option directs the Quartus Prime software to unmap the LEs in an atom netlist to logic gates and then map the gates back to Altera-specific primitives. Using Altera-specific primitives enables the Fitter to remap the circuits using architecture-specific techniques.

The Quartus Prime technology mapper optimizes the design to achieve maximum speed performance, minimum area usage, or balances high performance and minimal logic usage, according to the setting of the **Optimization Technique** option. Set this option to **Speed** or **Balanced**.

The physical synthesis optimizations occur during the Fitter stage of the Quartus Prime compilation. Physical synthesis optimizations make placement-specific changes to the netlist that improve speed performance results for a specific Altera device.

The following physical synthesis optimizations are available during the Fitter stage for improving performance:

- Physical synthesis for combinational logic
- Automatic asynchronous signal pipelining
- Physical synthesis for registers

  - Register duplication
  - Register retiming

Note: If you want the performance gain from physical synthesis only on parts of your design, you can apply the physical synthesis options on specific instances.

To apply physical synthesis assignments for fitting on a per-instance basis, use the Quartus Prime Assignment Editor. The following assignments are available as instance assignments:

- Perform physical synthesis for combinational logic
- Perform register duplication for performance
- Perform register retiming for performance
- Perform automatic asynchronous signal pipelining

Related Information

- Perform WYSIWYG Primitive Resynthesis Logic Option online help

- **Optimization Technique Logic Option online help**

Turn Off Extra-Effort Power Optimization Settings

If PowerPlay power optimization settings are set to **Extra Effort**, your design performance can be affected. If improving timing performance is more important than reducing power use, set the PowerPlay power optimization setting to **Normal**.

Related Information

PowerPlay Power Optimization Logic Option online help

Power Optimization documentation on page 10-1

Optimize Synthesis for Speed, Not Area

The manner in which the design is synthesized has a large impact on design performance. Design performance varies depending on the way the design is coded, the synthesis tool used, and the options specified when synthesizing. Change your synthesis options if a large number of paths are failing or if specific paths are failing badly and have many levels of logic.

Set your device and timing constraints in your synthesis tool. Synthesis tools are timing-driven and optimized to meet specified timing requirements. If you do not specify a target frequency, some synthesis tools optimize for area.

Some synthesis tools offer an easy way to instruct the tool to focus on speed instead of area.

You can also specify this logic option for specific modules in your design with the Assignment Editor while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Area** (if area is an important concern). You can also use the **Speed Optimization Technique for Clock Domains** option in the Assignment Editor to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.

To achieve best performance with push-button compilation, follow the recommendations in the following sections for other synthesis settings. You can use DSE II to experiment with different Quartus Prime synthesis options to optimize your design for the best performance.

Related Information

- **Optimization Technique Logic Option online help**
- **Design Space Explorer II online help**

Flatten the Hierarchy During Synthesis

Synthesis tools typically let you preserve hierarchical boundaries, which can be useful for verification or other purposes. However, the best optimization results generally occur when the synthesis tool optimizes across hierarchical boundaries, because doing so often allows the synthesis tool to perform the most logic minimization, which can improve performance. Whenever possible, flatten your design hierarchy to achieve the best results.

Set the Synthesis Effort to High

Some synthesis tools offer varying synthesis effort levels to trade off compilation time with synthesis results. Set the synthesis effort to **high** to achieve best results when applicable.

Duplicate Logic for Fan-Out Control

Duplicating logic or registers can help improve timing in cases where moving a register in a failing timing path to reduce routing delay creates other failing paths or where there are timing problems due to the fan-out of the registers. Most often, timing failures occur not because of the high fan-out registers, but because of the location of those registers. Duplicating registers, where source and destination registers are physically close, can help improve slack on critical paths.

Many synthesis tools support options or attributes that specify the maximum fan-out of a register. When using Quartus Prime integrated synthesis, you can set the **Maximum Fan-Out** logic option in the Assignment Editor to control the number of destinations for a node so that the fan-out count does not exceed a specified value. You can also use the `maxfan` attribute in your HDL code. The software duplicates the node as required to achieve the specified maximum fan-out.

Logic duplication using **Maximum Fan-Out** assignments normally increases resource utilization and can potentially increase compilation time, depending on the placement and the total resource usage within the selected device. The improvement in timing performance that results because of **Maximum Fan-Out** assignments is very design-specific. This is because when you use the **Maximum Fan-Out** assignment, although the Fitter duplicates the source logic to limit the fan-out, it may not be able to control the destinations that each of the duplicated sources drive. Since the **Maximum Fan-Out** destination does not specify which of the destinations the duplicated source should drive, it is possible that it might still be driving logic located all around the device. To avoid this situation, you could use the **Manual Logic Duplication** logic option.

If you are using **Maximum Fan-Out** assignments, Altera recommends benchmarking your design with and without these assignments to evaluate whether they give the expected improvement in timing performance. Use the assignments only when you get improved results.

You can manually duplicate registers in the Quartus Prime software regardless of the synthesis tool used. To duplicate a register, apply the **Manual Logic Duplication** logic option to the register with the Assignment Editor.

Note: Various Fitter optimizations may cause a small violation to the **Maximum Fan-Out** assignments to improve timing.

Related Information
Manual Logic Duplication Logic Option online help

Prevent Shift Register Inference

In some cases, turning off the inference of shift registers increases performance. Doing so forces the software to use logic cells to implement the shift register instead of implementing the registers in memory blocks using the ALTSHIFT_TAPS IP core. If you implement shift registers in logic cells instead of memory, logic utilization is increased.

Use Other Synthesis Options Available in Your Synthesis Tool

With your synthesis tool, experiment with the following options if they are available:

- Turn on register balancing or retiming
- Turn on register pipelining
- Turn off resource sharing

These options can increase performance, but typically increase the resource utilization of your design.

Fitter Seed

The Fitter seed affects the initial placement configuration of the design. Changing the seed value changes the Fitter results because the fitting results change whenever there is a change in the initial conditions. Each seed value results in a somewhat different fit, and you can experiment with several different seeds to attempt to obtain better fitting results and timing performance.

When there are changes in your design, there is some random variation in performance between compilations. This variation is inherent in placement and routing algorithms—there are too many possibilities to try them all and get the absolute best result, so the initial conditions change the compilation result.

Note: Any design change that directly or indirectly affects the Fitter has the same type of random effect as changing the seed value. This includes any change in source files, **Compiler Settings** or **Timing Analyzer Settings**. The same effect can appear if you use a different computer processor type or different operating system, because different systems can change the way floating point numbers are calculated in the Fitter.

If a change in optimization settings slightly affects the register-to-register timing or number of failing paths, you cannot always be certain that your change caused the improvement or degradation, or whether it could be due to random effects in the Fitter. If your design is still changing, running a seed sweep (compiling your design with multiple seeds) determines whether the average result has improved after an optimization change and whether a setting that increases compilation time has benefits worth the increased time (such as setting the **Physical Synthesis Effort** to **Extra**). The sweep also shows the amount of random variation to expect for your design.

If your design is finalized, you can compile your design with different seeds to obtain one optimal result. However, if you subsequently make any changes to your design, you might need to perform seed sweep again.

On the Assignments menu, select **Compiler Settings** to control the initial placement with the seed. You can use the DSE II to perform a seed sweep easily.

You can use the following Tcl command from a script to specify a Fitter seed:

```
set_global_assignment -name SEED <value>
```

Related Information

Design Space Explorer II online help
Information about compiling your design with different seeds using Design Space Explorer II

Set Maximum Router Timing Optimization Level

To improve routability in designs where the router did not pick up the optimal routing lines, set the **Router Timing Optimization Level** to **Maximum**. This setting determines how aggressively the router tries to meet the timing requirements. Setting this option to **Maximum** can increase design speed slightly at the cost of increased compilation time. Setting this option to **Minimum** can reduce compilation time at the cost of slightly reduced design speed. The default value is **Normal**.

Related Information

Router Timing Optimization Level Logic Option online help

LogicLock Plus Assignments

Using LogicLock Plus assignments to improve timing performance is only recommended for older Altera devices, such as the MAX II family. For other device families, especially for larger devices such as Arria and Stratix series devices, Altera does not recommend using LogicLock Plus assignments to improve timing performance. For these devices, use the LogicLock Plus feature for performance preservation and to floorplan your design.

LogicLock Plus assignments do not always improve the performance of the design. In many cases, you cannot improve upon results from the Fitter by making location assignments. If there are existing LogicLock Plus assignments in your design, remove the assignments if your design methodology permits it. Recompile the design, and then check if the assignments are making the performance worse.

When making LogicLock Plus assignments, it is important to consider how much flexibility to give the Fitter. LogicLock Plus assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher Fitter effort, but reduce the chance of design overconstraint. The following types of LogicLock Plus assignments are available, listed in the order of decreasing flexibility:

- Auto size, floating location regions
- Fixed size, floating location regions
- Fixed size, locked location regions

If you are unsure of how big or where a LogicLock Plus region should go, the **Auto/Floating** options are useful for your first pass. After you determine where a LogicLock Plus region must go, modify the Fixed/Locked regions, as Auto/Floating LogicLock Plus regions can hurt your overall performance. To determine what to put into a LogicLock Plus region, refer to the timing analysis results and analyze the critical paths in the Chip Planner. The register-to-register timing paths in the Timing Analyzer section of the Compilation Report help you recognize patterns.

Related Information

- **Analyzing and Optimizing the Design Floorplan with the Chip Planner** on page 12-1

Hierarchy Assignments

For a design with the hierarchy shown in the figure, which has failing paths in the timing analysis results similar to those shown in the table, mod_A is probably a problem module. In this case, a good strategy to fix the failing paths is to place the mod_A hierarchy block in a LogicLock Plus region so that all the nodes are closer together in the floorplan.

Figure 9-6: Design Hierarchy
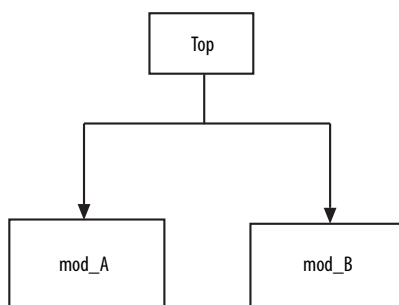
Table 9-3: Failing Paths in a Module Listed in Timing Analysis

From	To
`|mod_A|reg1`	`|mod_A|reg9`
`|mod_A|reg3`	`|mod_A|reg5`
`|mod_A|reg4`	`|mod_A|reg6`
`|mod_A|reg7`	`|mod_A|reg10`
`|mod_A|reg0`	`|mod_A|reg2`

Related Information

- **Analyzing and Optimizing the Design Floorplan with the Chip Planner** on page 12-1

Location Assignments

If a small number of paths are failing to meet their timing requirements, you can use hard location assignments to optimize placement. Location assignments are less flexible for the Quartus Prime Fitter than LogicLock Plus assignments. In some cases, when you are familiar with your design, you can enter location constraints in a way that produces better results.

Note: Improving fitting results, especially for larger devices, such as Arria and Stratix series devices, can be difficult. Location assignments do not always improve the performance of the design. In many cases, you cannot improve upon the results from the Fitter by making location assignments.

Metastability Analysis and Optimization Techniques

Metastability problems can occur when a signal is transferred between circuitry in unrelated or asynchronous clock domains, because the designer cannot guarantee that the signal will meet its setup and hold time requirements. The mean time between failures (MTBF) is an estimate of the average time between instances when metastability could cause a design failure.

You can use the Quartus Prime software to analyze the average MTBF due to metastability when a design synchronizes asynchronous signals and to optimize the design to improve the MTBF. These metastability features are supported only for designs constrained with the TimeQuest analyzer, and for select device families.

If the MTBF of your design is low, refer to the Metastability Optimization section in the Timing Optimization Advisor, which suggests various settings that can help optimize your design in terms of metastability.

This chapter describes how to enable metastability analysis and identify the register synchronization chains in your design, provides details about metastability reports, and provides additional guidelines for managing metastability.

Related Information

- **Understanding Metastability in FPGAs white paper**
- **Managing Metastability with the Quartus Prime Software documentation**

Periphery to Core Register Placement and Routing Optimization

The Periphery to Core Register Placement and Routing Optimization (P2C) option specifies whether the Fitter performs targeted placement and routing optimization on direct connections between periphery logic and registers in the FPGA core. P2C is an optional pre-routing-aware placement optimization stage that enables you to more reliably achieve timing closure.

Note: The **Periphery to Core Register Placement and Routing Optimization** option applies in both directions, periphery to core and core to periphery.

Transfers between external interfaces (for example, high-speed I/O or serial interfaces) and the FPGA often require routing many connections with tight setup and hold timing requirements. When this option is turned on, the Fitter performs P2C placement and routing decisions before those for core placement and routing. This reserves the necessary resources to ensure that your design achieves its timing requirements and avoids routing congestion for transfers with external interfaces.

This option is available as a global assignment, or can be applied to specific instances within your design.

Figure 9-7: Periphery to Core Register Placement and Routing Optimization (P2C) Flow

P2C runs after periphery placement, and generates placement for core registers on corresponding P2C/C2P paths, and core routing to and from these core registers.

Step	Description
User Design	
Synthesis	
Periphery Placement	*Generates periphery placement and routing.*
P2C	*Generates core register placement for periphery interfaces.* *Generates core Routing to/from those registers.*
Core Placement	
Routing	
Design Implementation	
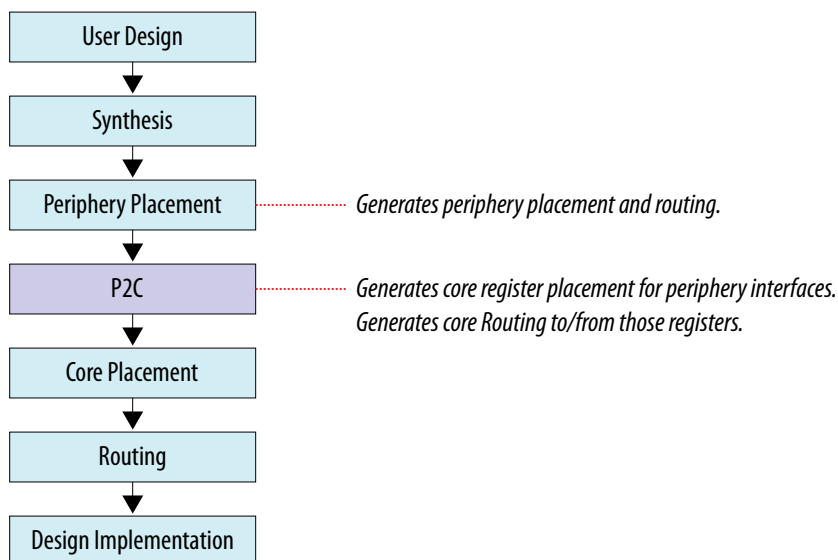
Related Information

- **Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box** on page 9-23
- **Setting Periphery to Core Optimizations in the Assignment Editor** on page 9-24
- **Viewing Periphery to Core Optimizations in the Fitter Report** on page 9-25

Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box

The **Periphery to Core Placement and Routing Optimization** setting specifies whether the Fitter should perform targeted placement and routing optimization on direct connections between periphery logic and registers in the FPGA core.

You can optionally perform periphery to core optimizations by instance with settings in the Assignment Editor.

1. In the Quartus Prime software, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.
2. In the **Advanced Fitter Settings** dialog box, for the **Periphery to Core Placement and Routing Optimization** option, select one of the following options depending on how you want to direct periphery to core optimizations in your design:
    a. Select **Auto** to direct the software to automatically identify transfers with tight timing windows, place the core registers, and route all connections to or from the periphery.
    b. Select **On** to direct the software to globally optimize all transfers between the periphery and core registers, regardless of timing requirements.

    > **Note:** Setting this option to **On** in the **Advanced Fitter Settings** is not recommended. The intended use for this setting is in the Assignment Editor to force optimization for a targeted set of nodes or instance.

    c. Select **Off** to disable periphery to core path optimization in your design.

Related Information

-   **Periphery to Core Register Placement and Routing Optimization** on page 9-23
-   **Setting Periphery to Core Optimizations in the Assignment Editor** on page 9-24

Setting Periphery to Core Optimizations in the Assignment Editor

When you turn on the **Periphery to Core Placement and Routing Optimization** (P2C/C2P) setting in the Assignment Editor, the Quartus Prime software performs periphery to core, or core to periphery optimizations on selected instances in your design.

You can optionally perform periphery to core optimizations by instance with settings in the **Advanced Fitter Settings** dialog box.

1. In the Quartus Prime software, click **Assignments** > **Assignment Editor**.
2. For the selected path, double-click the **Assignment Name** column, and then click the **Periphery to core register placement and routing optimization** option in the drop-down list.
3. In the **To** column, choose either a periphery node or core register node on a P2C/C2P path you want to optimize. Leave the **From** column empty.
    For paths to appear in the Assignments Editor, you must first run Analysis & Synthesis on your design.
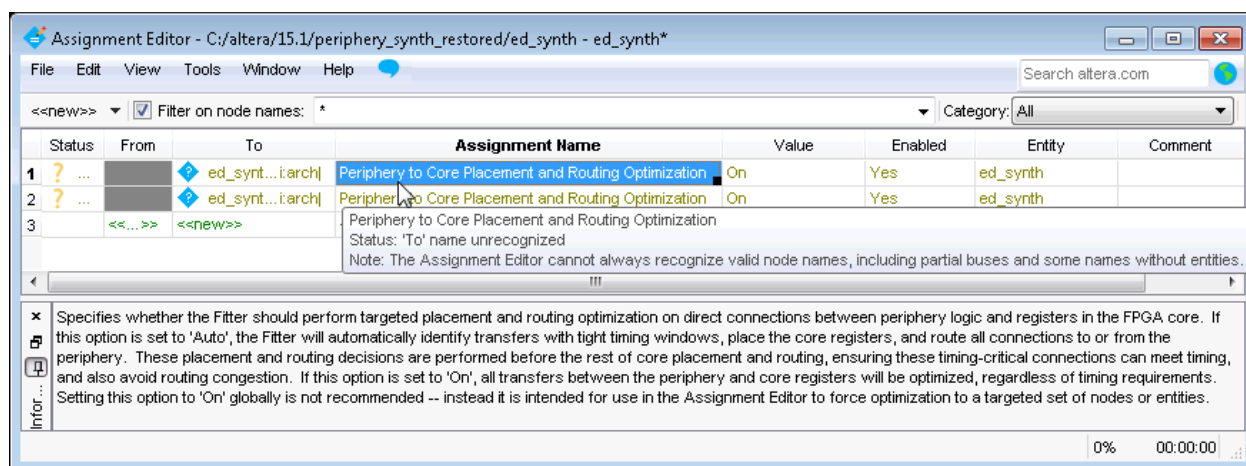
Related Information

- **Periphery to Core Register Placement and Routing Optimization** on page 9-23
- **Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box** on page 9-23

Viewing Periphery to Core Optimizations in the Fitter Report

The Quartus Prime software generates a periphery to core placement and routing optimization summary in the **Fitter (Place & Route)** report after compilation.

1. Compile your Quartus Prime project.
2. In the **Tasks** pane, select **Compilation**.
3. Under **Fitter (Place & Route)**, double-click **View Report**.
4. In the **Fitter** folder, expand the **Place Stage** folder.
5. Double-click **Periphery to Core Transfer Optimization Summary**.

Table 9-4: Fitter Report - Periphery to Core Transfer Optimization (P2C) Summary

From Path	To Path	Status
Node 1	Node 2	**Placed and Routed**—Core register is locked. Periphery to core/core to periphery routing is committed.
Node 3	Node 4	**Placed but not Routed**—Core register is locked. Routing is not committed. This occurs when P2C is not able to optimize all targeted paths within a single group, for example, the same delay/wire requirement, or the same control signals. Partial P2C routing commitments may cause unresolvable routing congestion.

Send Feedback

From Path	To Path	Status
Node 5	Node 6	**Not Optimized**—This occurs when P2C is set to **Auto** and the path is not optimized due to one of the following issues: 1. The delay requirement is impossible to achieve. 2. The minimum delay requirement (for hold timing) is too large. The P2C algorithm cannot efficiently handle cases when many wires need to be added to meet hold timing. 3. P2C encountered unresolvable routing congestion for this particular path.

Periphery to Core Transfer Optimization Summary		
	From	To	Status
1	Periphery to Core Transfer		
2	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[1].lane_gen[3].lane_inst	dut|arch|arch_inst|afi.if_inst|single_port.afi_rdata_valid_regs|sr_out[0]	Placed but Not Routed
3	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[2].lane_gen[2].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[63]	Placed but Not Routed
4	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[1].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[11]	Placed but Not Routed
5	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[7]	Placed but Not Routed
6	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[3]	Placed but Not Routed
7	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[0]	Placed but Not Routed
8	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[79]	Placed but Not Routed
9	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[75]	Placed but Not Routed
10	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[73]	Placed but Not Routed
11	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[2].lane_gen[3].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[71]	Placed but Not Routed
12	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[2].lane_gen[3].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[69]	Placed but Not Routed
13	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[1].lane_gen[3].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[35]	Placed but Not Routed
14	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[0].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[31]	Placed but Not Routed
15	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[3].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[27]	Placed but Not Routed
16	dut|arch|arch_inst|io_tiles_wrap_inst|io_tiles_inst|tile_gen[0].lane_gen[2].lane_inst	dut|arch|arch_inst|afi.if_inst|mem_sp_bidir_data.mem_dq.afi_regs_i|sr_out[23]	Placed but Not Routed

Related Information

Periphery to Core Register Placement and Routing Optimization on page 9-23

Design Evaluation for Timing Closure

Follow the guidelines in this section when you encounter timing failures in a design. The guidelines show you how to evaluate compilation results of a design and how to address some of the problems. While the guideline does not cover specific examples of restructuring RTL to improve design speed, the analysis techniques help you to evaluate changes that may have to be made to RTL to close timing.

Review Compilation Results

Review Messages

After compiling your design, review the messages in each section of the compilation report. Most designs that fail timing start out with other problems that are reported as warning messages during compilation. Determine what causes a warning message, and whether the warning should be fixed or ignored. After reviewing the warning messages, review the informational messages. Take note of anything unexpected,

for example, unconnected ports, ignored constraints, missing files, and assumptions or optimizations that the software made.

Evaluate Physical Synthesis Results

If physical synthesis is enabled, the software can duplicate and retime registers, and modify combinatorial logic during synthesis. After compilation, review the Optimization Results reports in the Analysis & Synthesis section. The reports list the optimizations performed by the physical synthesis optimizations, such as register duplication, retiming, and removal. These reports can be found in the Compilation Report panel.

Figure 9-8: Optimization Results Reports
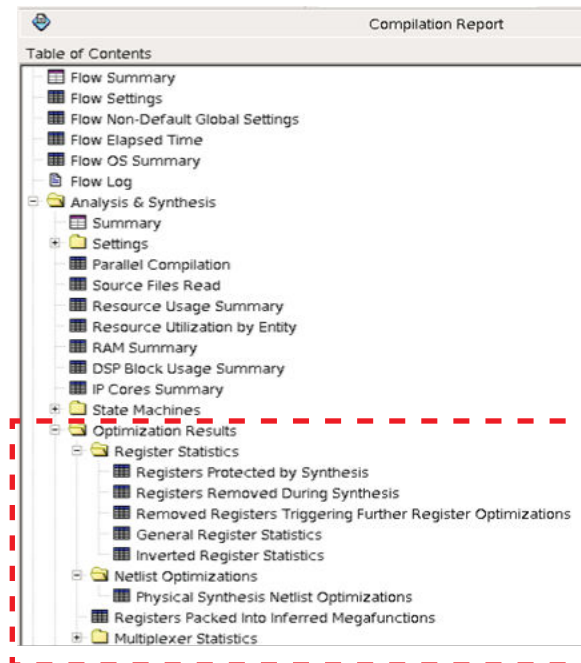
When physical synthesis is enabled, compilation messages include a summary of the physical synthesis algorithms that were run, the performance improvement each algorithm achieved, and the elapsed time. The reported improvement is the sum of the largest improvement estimated to be achievable in each timing-critical clock domain. The values for the slack improvements can vary between compiles because of the random starting point of the compilation algorithms, but the values should be similar. The figure shows an example of the messages.

Figure 9-9: Compilation Messages

Evaluate Fitter Netlist Optimizations

The Fitter can also perform netlist optimizations to the design netlist. Major changes include register packing, duplicating or deleting logic cells, retiming registers, inverting signals, or modifying nodes in a general way such as moving an input from one logic cell to another. These reports can be found in the Netlist Optimizations results of the Fitter section, and they should also be reviewed.

Evaluate Optimization Results

After checking what optimizations were done and how they improved performance, evaluate the runtime it took to get the extra performance. To reduce compilation time, review the physical synthesis and netlist optimizations over a couple of compilations, and edit the RTL to reflect the changes that physical synthesis performed. If a particular set of registers consistently get retimed, edit the RTL to retime the registers the same way. If the changes are made to match what the physical synthesis algorithms did, the physical synthesis options can be turned off to save compile time while getting the same type of performance improvement.

Evaluate Resource Usage

Evaluate a variety of resources used in the design, including global and non-global signal usage, routing utilization, and clustering difficulty.

Global and Non-global Usage

If your design contains a lot of clocks, evaluate global and non-global signals. Determine whether global resources are being used effectively, and if not, consider making changes. These reports can be found in the Resource Section under Fitter in the Compilation Report panel. The figure shows an example of inefficient use of a global clock. The highlighted line has a single fan-out from a global clock. Assigning it to a Regional Clock would make the Global Clock available for another signal. You can ignore signals with an empty value in the **Global Line Name** column as the signal uses dedicated routing, and not a clock buffer.

Figure 9-10: Inefficient Use of a Global Clock

Global & Other Fast Signals			
Location	Fan-Out	Global Resource Used	Global Line Name
FRACTIONALPLL_X98_Y2_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y2_N1	29044	Global Clock	GCLK7
PLLOUTPUTCOUNTER_X98_Y13_N1	253103	Global Clock	GCLK6
FF_X185_Y66_N13	280349	Global Clock	GCLK8
PIN_AE17	4887	Global Clock	GCLK4
FRACTIONALPLL_X98_Y11_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y3_N1	1	Global Clock	GCLK5
PLLOUTPUTCOUNTER_X98_Y1_N1	1691	Regional Clock	RCLK29
PLLOUTPUTCOUNTER_X98_Y8_N1	302	Regional Clock	RCLK23
PLLOUTPUTCOUNTER_X98_Y11_N1	141	Regional Clock	RCLK25
PLLOUTPUTCOUNTER_X98_Y10_N1	17	Regional Clock	RCLK22

The Non-Global High Fan-Out Signals report lists the highest fan-out nodes that are not routed on global signals. Reset and enable signals are at the top of the list. If there is routing congestion in the design, and there are high fan-out non-global nodes in the congested area, consider using global or regional signals to fan-out the nodes, or duplicate the high fan-out registers so that each of the duplicates can have fewer fan-outs. Use the Chip Planner to locate high fan-out nodes, to report routing congestion, and to determine whether the alternatives are viable.

Routing Usage

Review routing usage reported in the **Fitter Resource Usage Summary** report. The figure shows an example of the report.

Figure 9-11: Fitter Resource Usage Summary Report

The average interconnect usage reports the average amount of interconnect that is used, out of what is available on the device. The peak interconnect usage reports the largest amount of interconnect used in the most congested areas. Designs with an average value below 50% typically do not have any problems with routing. Designs with an average between 50-65% may have difficulty routing. Designs with an average over 65% typically have difficulty meeting timing unless the RTL is well designed to tolerate a highly utilized chip. Peak values at or above 90% are likely to have problems with timing closure; a 100% peak value indicates that all routing in an area of the device has been used, so there is a high possibility of degradation in timing performance. The figure shows the **Report Routing Utilization** report.

Figure 9-12: Report Routing Utilization Report
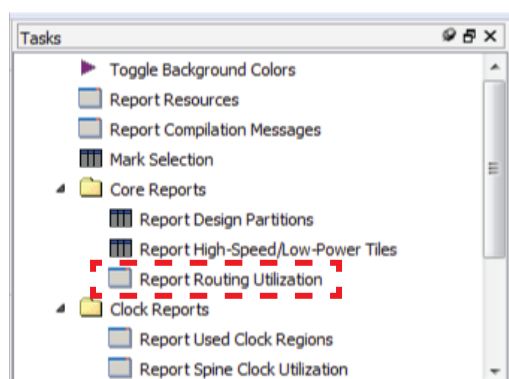
Wires Added for Hold

As part of the fitting process, the router can add wire between register paths to increase delay to meet hold time requirements. During the routing process, the router reports how much extra wire was used to meet hold time requirements. Excessive amounts of added wire can indicate problems with the constraint. Typically it would be caused by incorrect multicycle transfers, particularly between different rate clocks, and between different clock networks. The Fitter reports how much routing delay was added in the **Estimated Delay Added for Hold Timing** report. Specific register paths can be reviewed to view whether a delay was added to meet hold requirements.

Figure 9-13: Estimated Delay Added for Hold Timing Report

An example of an incorrect constraint which can cause the router to add wire for hold requirements is when there is data transfer from 1x to 2x clocks. Assume the design intent is to allow two cycles per transfer. Data can arrive any time in the two destination clock cycles by adding a multicycle setup constraint as shown in the example:

```
set_multicycle_path -from 1x -to 2x -setup -end 2
```

The timing requirement is relaxed by one 2x clock cycle, as shown in the black line in the waveform in the figure.

Figure 9-14: Timing Requirement Relaxed Waveform
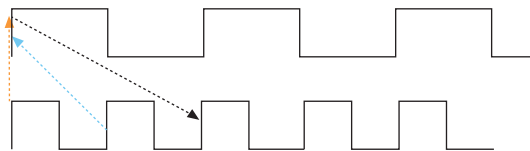
However, the default hold requirement, shown with the dashed blue line, may cause the router to add wire to guarantee that data is delayed by one cycle. To correct the hold requirement, add a multicycle constraint with a hold option.

```
set_multicycle_path -from 1x -to 2x -setup -end 2
set_multicycle_path -from 1x -to 2x -hold -end 1
```

The orange dashed line in the figure above represents the hold relationship, and no extra wire is required to delay the data.

The router can also add wire for hold timing requirements when data is transferred in the same clock domain, but between clock branches that use different buffering. Transferring between clock network types happens more often between the periphery and the core. The figure below shows a case where data is coming into a device, and uses a periphery clock to drive the source register, and a global clock to drive the destination register. A global clock buffer has larger insertion delay than a periphery clock buffer. The clock delay to the destination register is much larger than to the source register, hence extra delay is necessary on the data path to ensure that it meets its hold requirement.

Figure 9-15: Clock Delay
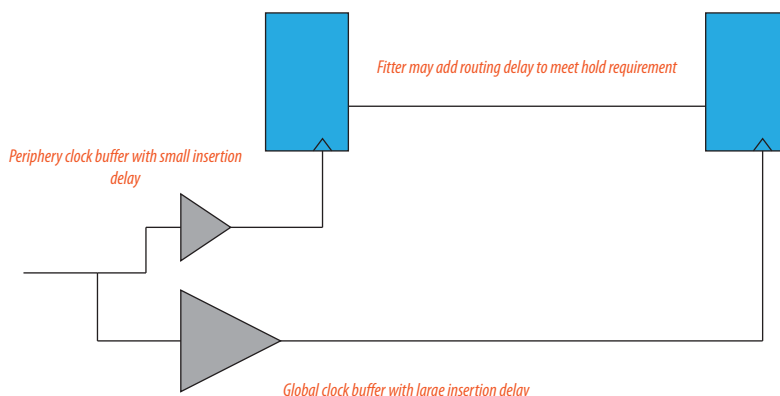
To identify cases where a path has different clock network types, review the path in the TimeQuest timing analyzer, and check nodes along the source and destination clock paths. Also, check the source and destination clock frequencies to see whether they are the same, or multiples, and whether there are multicycle exceptions on the paths. In some cases, cross-domain paths may also be false by intent, so make sure there are false path exceptions on those.

If you suspect that routing is added to fix real hold problems, then disable the **Optimize hold timing** option. Recompile the design and rerun timing analysis to uncover paths that fail hold time.

Figure 9-16: Optimize Hold Timing Option
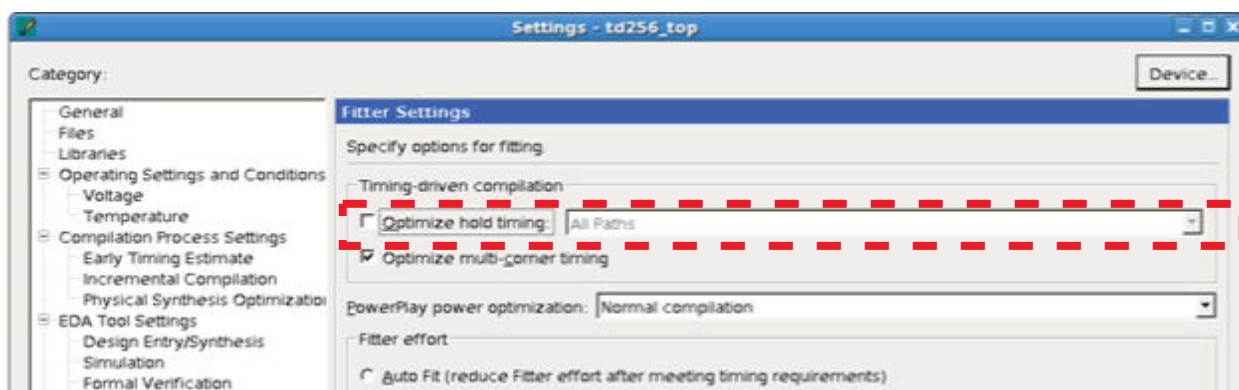
Disabling the **Optimize hold timing** option is a debug step, and should be left enabled (default state) during normal compiles. Wire added for hold is a normal part of timing optimization during routing and is not always a problem.

Evaluate Other Reports and Adjust Settings Accordingly

Difficulty Packing Design

In the Fitter Resource Section, under the **Resource Usage Summary**, review the **Difficulty Packing Design** report. The **Difficulty Packing Design** report details the effort level (low, medium, or high) of the Fitter to fit the design into the device, partition, and LogicLock Plus region. As the effort level of **Difficulty Packing Design** increases, timing closure gets harder. Going from medium to high can result in significant drop in performance or increase in compile time. Consider reducing logic to reduce packing difficulty.

Review Ignored Assignments

The **Compilation Report** includes details of any assignments ignored by the Fitter. Assignments typically get ignored if design names change, but assignments are not updated. Make sure any intended assignments are not being ignored.

Review Non-Default Settings

The reports from Synthesis and Fitter show non-default settings used in a compilation. Review the non-default settings to ensure the design benefits from the change.

Review Floorplan

Use the Chip Planner for reviewing placement. The Chip Planner can be used to locate hierarchical entities, and colors each located entity in the floorplan. Look for logic that seems out of place, based on where you would expect it to be. For example, logic that interfaces with I/Os should be close to the I/Os, and logic that interfaces with an IP or memory must be close to the IP or memory. The figure shows an example of a floorplan with color-coded entities. In the floorplan, the green block is spread apart. Check to see if those paths are failing timing, and if so, what connects to that module that could affect placement. The blue and aqua blocks are spread out and mixed together. Check and see if there are many connections between the two modules that may contribute to this. The pink logic at the bottom should interface with I/Os at the bottom edge.

Figure 9-17: Floorplan with Color-Coded Entities
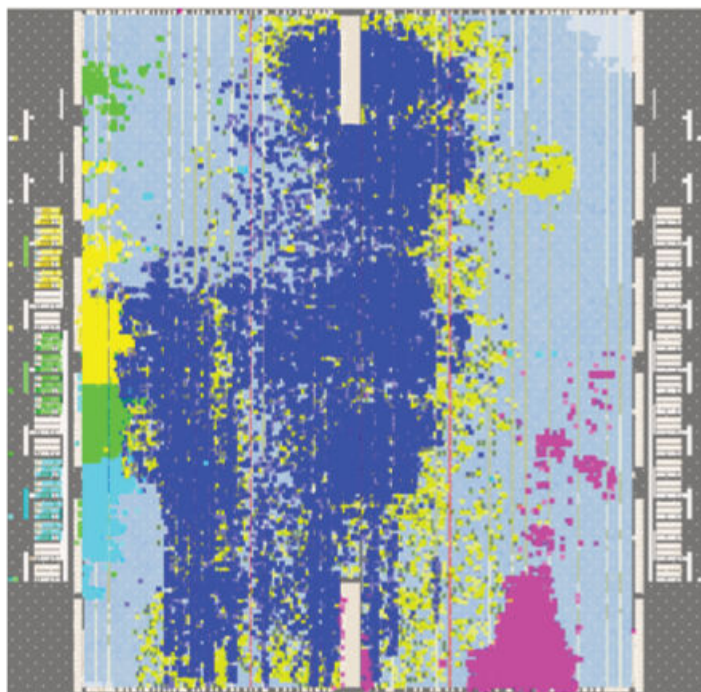
Check fan-in and fan-out of a highlighted module by using the buttons on the task bar shown in the figure below.

Figure 9-18: Fan-in and Fan-Out Buttons

Look for signals that go a long way across the chip and see if they are contributing to timing failures.

Check global signal usage for signals that may affect logic placement. Logic feeding a global buffer may be pulled close to the buffer, away from related logic. High fan-out on non-global resource may pull logic together.

Check for routing congestion. Highly congested areas may cause logic to be spread out, and the design may be difficult to route.

Evaluate Placement and Routing

Review duration of parts of compile time in Fitter messages. If routing takes much more time than placement, then meeting timing may be more difficult than the placer predicted.

Adjust Placement Effort

Increasing the **Placement Effort Multiplier** to improve placement quality may be a good tradeoff at the cost of higher compile time, but the benefit is design dependent. The value should be adjusted after

reviewing and optimizing other settings and RTL. Try an increased value, up to 4, and reset to default if performance or compile time does not improve.

Figure 9-19: Placement Effort Multiplier
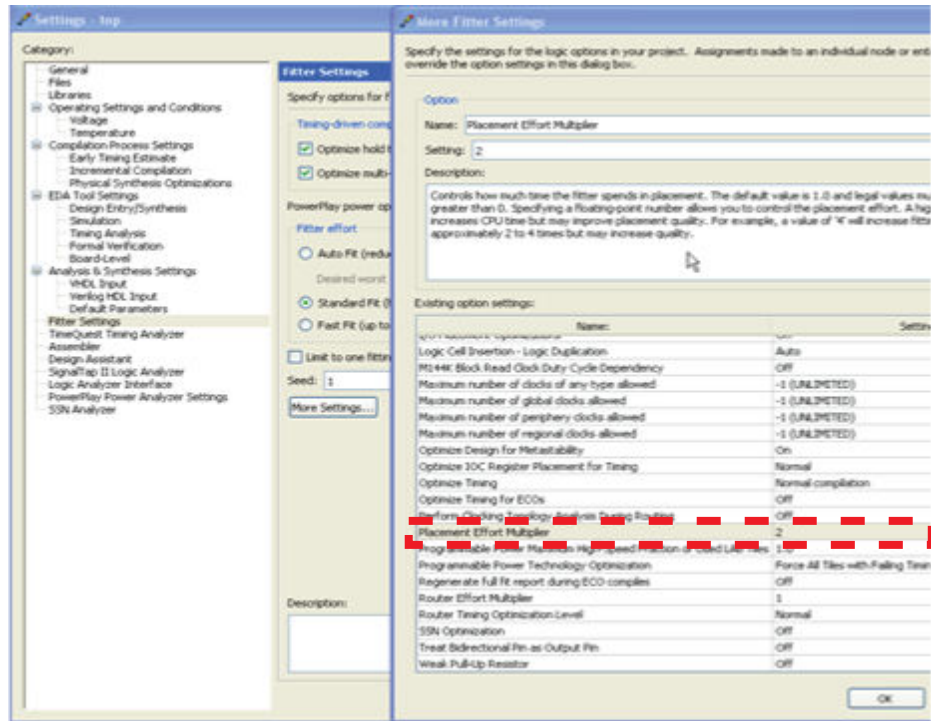
Adjust Fitter Effort

To increase effort, enable the **Standard Fit (highest effort)** option. The default **Auto Fit** option reduces Fitter effort when it estimates timing requirements are met.

Figure 9-20: Fitter Effort
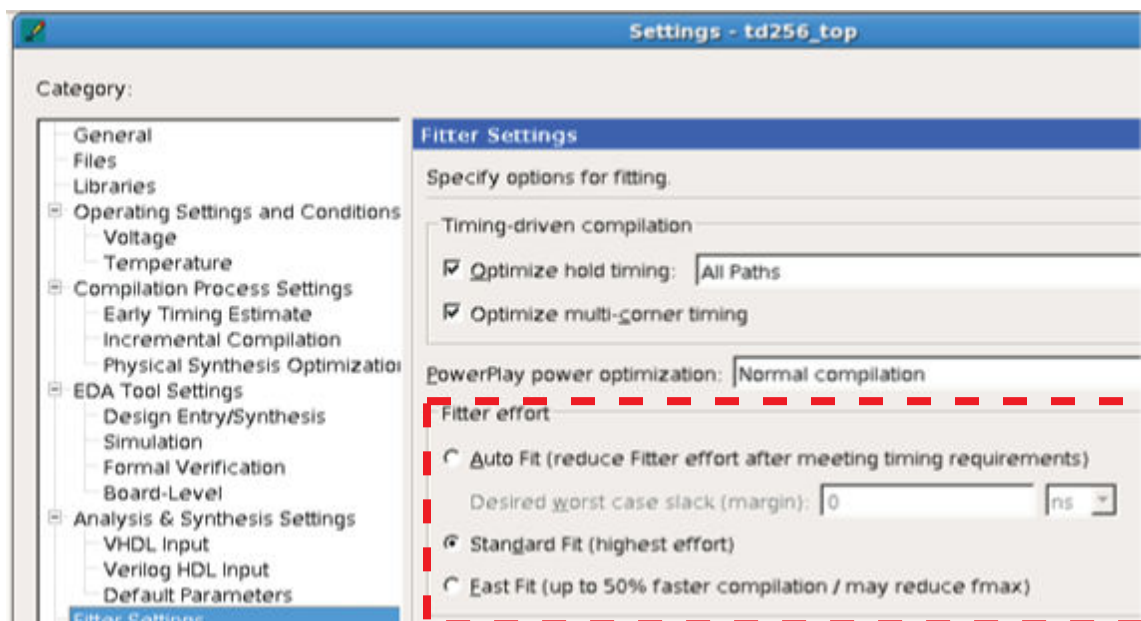
Review Timing Constraints

Ensure that clocks are constrained with the correct frequency requirements. Using the `derive_pll_clocks` assignment keeps generated clock settings updated. TimeQuest can be useful in reviewing SDC constraints. For example, under **Diagnostic** in the Task panel, the **Report Ignored Constraints** report shows any incorrect names in the design, most commonly caused by changes in the design hierarchy. Use the **Report Unconstrained Paths** report to locate unconstrained paths. Add constraints as necessary so that the design can be optimized.

Review Details of Timing Paths

Show Timing Path Routing

Showing routing for a path can help uncover unusual routing delays. In the TimeQuest Tasks panel, enable the **Report panel name** option, and then select **Report Timing**. Then, turn on the **Show routing** option to show routing wires in the path.

Figure 9-21: Show Routing
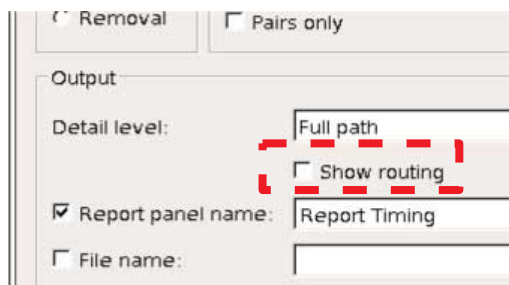
The **Extra Fitter Information** tab shows a miniature floorplan with the path highlighted. The path can also be located in the Chip Planner for viewing routing congestion, and to view whether nodes in a path are placed close together or far apart.

Global Network Buffers

A routing path can be used to identify global network buffers that fail timing. Buffer locations are named according to the network they drive.

- CLK_CTRL_Gn—for Global driver
- CLk_CTRL_Rn—for Regional driver

Buffers to access the global networks are located in the center of each side of the device. The buffering to route a core logic signal on a global signal network will cause insertion delay. Some trade offs to consider for global and non-global routing are source location, insertion delay, fan-out, distance a signal travels, and possible congestion if the signal is demoted to local routing.

Source Location

If the register feeding the global buffer cannot be moved closer, then consider changing either the design logic or the routing type.

Insertion Delay

If a global signal is required, consider adding half a cycle to timing by using a negative-edge triggered register to generate the signal (top figure) and use a multicycle setup constraint (bottom figure).

Figure 9-22: Negative-Edge Triggered Register
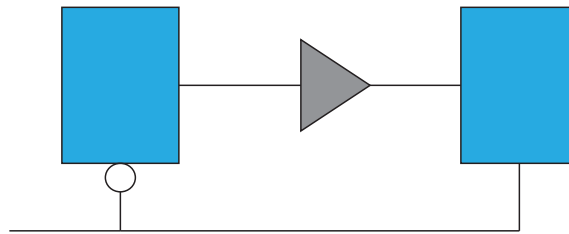
Figure 9-23: Multicycle Setup Constraint

set_multicycle_path −from <generating register> -setup −end 2
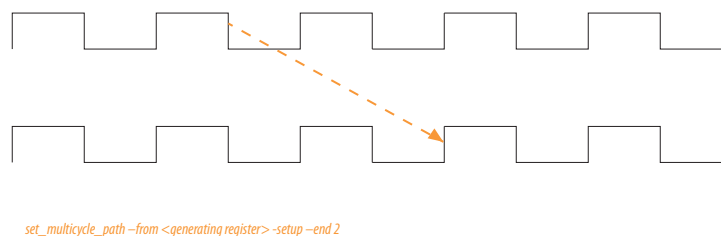
Fan-Out

Nodes with very high fan-out that use local routing tend to pull logic that they drive close to the source node. This can make other paths fail timing. Duplicating registers can help reduce the impact of high fan-out paths. Consider manually duplicating and preserving these registers. Using a MAX_FANOUT assignment

may make arbitrary groups of fan-out nodes, whereas a designer can make more intelligent fan-out groups.

Global Networks

If a signal should use a different type of global signal than it has automatically been assigned, use the Global Signal assignment to control the global signal usage on a per-signal basis. For example, if local routing is desired, set the Global Signal assignment to **OFF**.

Figure 9-24: Global Signal Assignment

To	Assignment Name	Value	Enabled
reg_clk	Global Signal	Off	Yes

Resets and Global Networks

Reset signals are often routed on global networks. Sometimes, the use of a global network causes recovery failures. Consider reviewing the placement of the register that generates the reset and the routing path of the signal.

Suspicious Setup

Suspicious setup failures include paths with very small or very large requirements. One typical cause is math precision error. For example, 10Mhz/3 = 33.33 ns per period. In three cycles, the time would be 99.999 ns vs 100.000 ns. Setting a maximum delay could provide an appropriate setup relationship.

Another cause of failure would be paths that should be false by design intent, such as:

- asynchronous paths that are handled through FIFOs, or
- slow asynchronous paths that rely on handshaking for data that remain available for multiple clock cycles.

To prevent the Fitter from having to meet unnecessarily restrictive timing requirements, consider adding false or multicycle path statements.

Logic Depth

The **Statistics** tab in the TimeQuest path report shows the levels of logic in a path. If the path fails timing and the number of logic levels is high, consider adding pipelining in that part of the design.

Auto Shift Register Replacement

Shift registers or register chains can be converted to RAM during synthesis to save area. However, conversion to RAM often reduces speed. The names of the converted registers will include "altshift_taps".

If paths that fail timing begin or end in shift registers, consider disabling the **Auto Shift Register Replacement** option. Registers that are intended for pipelining should not be converted. For shift registers that are converted to a chain, evaluate area/speed trade off of implementing in RAM or logic cells. If a design is close to full, shift register conversion to RAM may benefit non-critical clock domains by saving area. The settings can be changed globally or on a register or hierarchy basis from the default of **AUTO** to **OFF**.

Clocking Architecture

Review the clock region boundaries in the Chip Planner. You must place registers driven by a regional clock in one quadrant of the chip.

Figure 9-25: Clock Regions
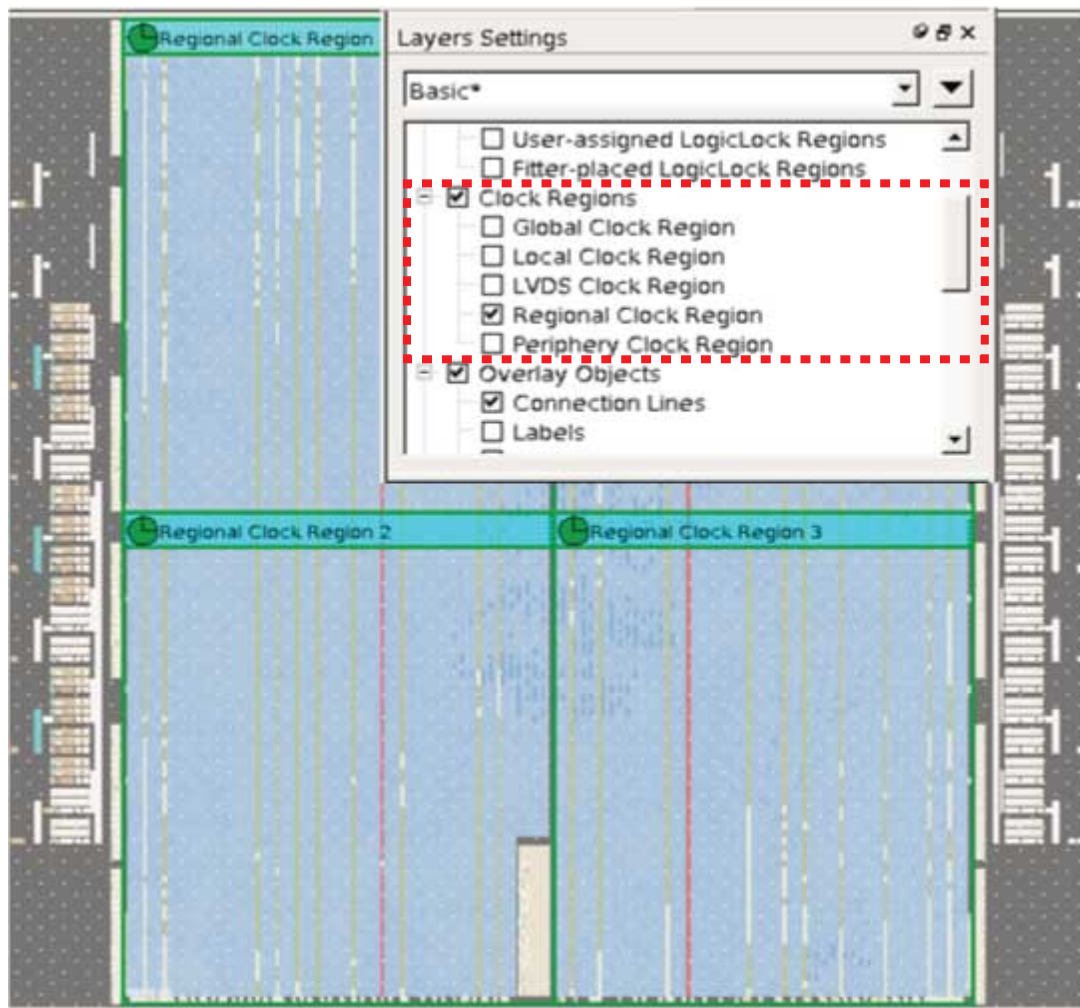
Timing failure can occur when the I/O interface at the top of the device connects to logic driven by a regional clock which is in one quadrant of the device, and placement restrictions force long paths to and from some of the I/Os to logic across quadrants.

Use a different type of clock source to drive the logic - global, which covers the whole device, or dual-regional which covers half the device. Alternatively, you can reduce the frequency of the I/O interface to accommodate the long path delays. You can also redesign the pinout of the device to place all the specified I/Os adjacent to the regional clock quadrant. This issue can happen when register locations are restricted, such as with LogicLock Plus regions, clocking resources, or hard blocks (memories, DSPs, IPs). The Extra Fitter Information tab in the TimeQuest report informs you when placement is restricted for nodes in a path.

Timing Closure Recommendations

The Report Timing Closure Recommendations task in the TimeQuest analyzer analyzes paths and provides specific recommendations based on path characteristics.

Making Adjustments and Recompiling

Look for obvious problems that you can fix with minimal effort. To identify where the Compiler had trouble meeting timing, perform seed sweeping with about five compiles. Doing so shows consistently failing paths. Consider recoding or redesigning that part of the design.

To reach timing closure, a well written RTL can be more effective than changing your compilation settings. Seed sweeping can also be useful if the timing failure is very small, and the design has already been optimized for performance improvements and is close to final release. Additionally, seed sweeping can be used for evaluating changes to compilation settings. Compilation results vary due to the random nature of fitter algorithms. If a compilation setting change produces lower average performance, undo the change.

Sometimes, settings or constraints can cause more problems than they fix. When significant changes to the RTL or design architecture have been made, compile periodically with default settings and without LogicLock Plus regions, and re-evaluate paths that fail timing.

Partitioning often does not help timing closure, and should be done at the beginning of the design process. Adding partitions can increase logic utilization if it prevents cross-boundary optimizations, making timing closure harder and increasing compile times.

Adding LogicLock Plus regions can be an effective part of timing closure, but must be done at the beginning of a design. Adding new LogicLock Plus regions at the end of the design cycle can restrict placement, hence lowering the performance.

Scripting Support

You can run procedures and make settings described in this manual in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> -to <instance name>
```

Note: If the *<value>* field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

Related Information

Tcl Scripting documentation on page 5-1

Quartus Prime Settings Reference File Manual

Initial Compilation Settings

Use the Quartus Prime Settings File (**.qsf**) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

The top table lists the **.qsf** variable name and applicable values for the settings described in the "Initial Compilation: Required Settings" section in the **Design Optimization Overview** chapter in the *Quartus Prime Handbook*. The bottom table lists the advanced compilation settings.

Table 9-5: Initial Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	`OPTIMIZE_IOC_REGISTER_ PLACEMENT_FOR_TIMING`	`ON`, `OFF`	Global
Optimize Hold Timing	`OPTIMIZE_HOLD_TIMING`	`OFF`, `IO PATHS AND MINIMUM TPD PATHS`, `ALL PATHS`	Global

Table 9-6: Advanced Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Router Timing Optimization level	`ROUTER_TIMING_OPTIMIZATION_LEVEL`	`NORMAL`, `MINIMUM`, `MAXIMUM`	Global

Resource Utilization Optimization Techniques (LUT-Based Devices)

The table lists the **.qsf** file variable name and applicable values for the settings described in **Optimizing Timing (LUT-Based Devices)**.

Table 9-7: Resource Utilization Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers **(1)**	`AUTO_PACKED_REGISTERS_<`*device family name*`>`	`OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, AUTO`	Global, Instance
Perform WYSIWYG Primitive Resynthesis	`ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP`	`ON, OFF`	Global, Instance

Setting Name	.qsf File Variable Name	Values	Type
Physical Synthesis for Combinational Logic for Reducing Area	`PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA`	`ON`, `OFF`	Global, Instance
Physical Synthesis for Mapping Logic to Memory	`PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR AREA`	`ON`, `OFF`	Global, Instance
Optimization Technique	*<device family name>*`_OPTIMIZATION_TECHNIQUE`	`AREA`, `SPEED`, `BALANCED`	Global, Instance
Speed Optimization Technique for Clock Domains	`SYNTH_CRITICAL_CLOCK`	`ON`, `OFF`	Instance
State Machine Encoding	`STATE_MACHINE_PROCESSING`	`AUTO`, `ONE-HOT`, `GRAY`, `JOHNSON`, `MINIMAL BITS`, `SEQUENTIAL`, `USER-ENCODE`	Global, Instance
Auto RAM Replacement	`AUTO_RAM_RECOGNITION`	`ON`, `OFF`	Global, Instance
Auto ROM Replacement	`AUTO_ROM_RECOGNITION`	`ON`, `OFF`	Global, Instance
Auto Shift Register Replacement	`AUTO_SHIFT_REGISTER_RECOGNITION`	`ON`, `OFF`	Global, Instance
Auto Block Replacement	`AUTO_DSP_RECOGNITION`	`ON`, `OFF`	Global, Instance
Number of Processors for Parallel Compilation	`NUM_PARALLEL_PROCESSORS`	Integer between `1` and `16` inclusive, or `ALL`	Global

Note to table :

1. Allowed values for this setting depend on the device family that you select.

I/O Timing Optimization Techniques (LUT-Based Devices)

The table lists the **.qsf** file variable name and applicable values for the I/O timing optimization settings.

Table 9-8: I/O Timing Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	`OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING`	ON, OFF	Global
Fast Input Register	`FAST_INPUT_REGISTER`	ON, OFF	Instance
Fast Output Register	`FAST_OUTPUT_REGISTER`	ON, OFF	Instance
Fast Output Enable Register	`FAST_OUTPUT_ENABLE_REGISTER`	ON, OFF	Instance
Fast OCT Register	`FAST_OCT_REGISTER`	ON, OFF	Instance

Register-to-Register Timing Optimization Techniques (LUT-Based Devices)

The table lists the **.qsf** file variable name and applicable values for the settings described in **Register-to-Register Timing Optimization Techniques (LUT-Based Devices)**.

Table 9-9: Register-to-Register Timing Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	`ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP`	ON, OFF	Global, Instance
Perform Physical Synthesis for Combinational Logic	`PHYSICAL_SYNTHESIS_COMBO_LOGIC`	ON, OFF	Global, Instance
Perform Register Duplication	`PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION`	ON, OFF	Global, Instance
Perform Register Retiming	`PHYSICAL_SYNTHESIS_REGISTER_RETIMING`	ON, OFF	Global, Instance
Perform Automatic Asynchronous Signal Pipelining	`PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING`	ON, OFF	Global, Instance
Physical Synthesis Effort	`PHYSICAL_SYNTHESIS_EFFORT`	NORMAL, EXTRA, FAST	Global
Fitter Seed	`SEED`	*<integer>*	Global
Maximum Fan-Out	`MAX_FANOUT`	*<integer>*	Instance
Manual Logic Duplication	`DUPLICATE_ATOM`	*<node name>*	Instance
Optimize Power during Synthesis	`OPTIMIZE_POWER_DURING_SYNTHESIS`	NORMAL, OFF EXTRA_EFFORT	Global

Setting Name	.qsf File Variable Name	Values	Type
Optimize Power during Fitting	`OPTIMIZE_POWER_DURING_FITTING`	`NORMAL, OFF EXTRA_EFFORT`	Global

Document Revision History

Table 9-10: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Added: *Periphery to Core Register Placement and Routing Optimization*. • Changed instances of *Quartus II* to *Quartus Prime*.
2014.12.15	14.1.0	• Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. • Updated DSE II content.
June 2014	14.0.0	• Dita conversion. • Removed content about obsolete devices that are no longer supported in QII software v14.0: Arria GX, Arria II, Cyclone III, Stratix II, Stratix III. • Replaced Megafunction content with IP core content.
November 2013	13.1.0	• Added Design Evaluation for Timing Closure section. • Removed Optimizing Timing (Macrocell-Based CPLDs) section. • Updated Optimize Multi-Corner Timing and Fitter Aggressive Routability Optimization. • Updated Timing Analysis with the TimeQuest Timing Analyzer to show how to access the **Report All Summaries** command. • Updated Ignored Timing Constraints to include a help link to *Fitter Summary Reports* with the **Ignored Assignment Report** information.

Date	Version	Changes
May 2013	13.0.0	• Renamed chapter title from Area and Timing Optimization to Timing Closure and Optimization. • Removed design and area/resources optimization information. • Added the following sections: Fitter Aggressive Routability Optimization. Tips for Analyzing Paths from/to the Source and Destination of Critical Path. Tips for Locating Multiple Paths to the Chip Planner. Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles.
November 2012	12.1.0	• Updated "Initial Compilation: Optional Fitter Settings" on page 13–2, "I/O Assignments" on page 13–2, "Initial Compilation: Optional Fitter Settings" on page 13–2, "Resource Utilization" on page 13–9, "Routing" on page 13–21, and "Resolving Resource Utilization Problems" on page 13–43.
June 2012	12.0.0	• Updated "Optimize Multi-Corner Timing" on page 13–6, "Resource Utilization" on page 13–10, "Timing Analysis with the TimeQuest Timing Analyzer" on page 13–12, "Using the Resource Optimization Advisor" on page 13–15, "Increase Placement Effort Multiplier" on page 13–22, "Increase Router Effort Multiplier" on page 13–22 and "Debugging Timing Failures in the TimeQuest Analyzer" on page 13–24. • Minor text edits throughout the chapter.
November 2011	11.1.0	• Updated the "Timing Requirement Settings", "Standard Fit", "Fast Fit", "Optimize Multi-Corner Timing", "Timing Analysis with the TimeQuest Timing Analyzer", "Debugging Timing Failures in the TimeQuest Analyzer", "LogicLock Plus Assignments", "Tips for Analyzing Failing Clock Paths that Cross Clock Domains", "Flatten the Hierarchy During Synthesis", "Fast Input, Output, and Output Enable Registers", and "Hierarchy Assignments" sections • Updated Table 13–6 • Added the "Spine Clock Limitations" section • Removed the Change State Machine Encoding section from page 19 • Removed Figure 13-5 • Minor text edits throughout the chapter

Date	Version	Changes
May 2011	11.0.0	• Reorganized sections in "Initial Compilation: Optional Fitter Settings" section • Added new information to "Resource Utilization" section • Added new information to "Duplicate Logic for Fan-Out Control" section • Added links to Help • Additional edits and updates throughout chapter
December 2010	10.1.0	• Added links to Help • Updated device support • Added "Debugging Timing Failures in the TimeQuest Analyzer" section • Removed Classic Timing Analyzer references • Other updates throughout chapter
August 2010	10.0.1	Corrected link
July 2010	10.0.0	• Moved Compilation Time Optimization Techniques section to new *Reducing Compilation Time* chapter • Removed references to Timing Closure Floorplan • Moved Smart Compilation Setting and Early Timing Estimation sections to new *Reducing Compilation Time* chapter • Added Other Optimization Resources section • Removed outdated information • Changed references to DSE chapter to Help links • Linked to Help where appropriate • Removed Referenced Documents section

Related Information

Quartus Handbook Archive

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

2015.11.02

Power Optimization

The Quartus Prime software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing your design's total power consumption using power-driven synthesis and power-driven place-and-route.

This chapter describes the power-driven compilation feature and flow in detail, as well as low power design techniques that can further reduce power consumption in your design. The techniques primarily target Arria®, Stratix®, and Cyclone® series of devices. These devices utilize a low-k dielectric material that dramatically reduces dynamic power and improves performance. Arria series, Stratix IV, and Stratix V device families include efficient logic structures called adaptive logic modules (ALMs) that obtain maximum performance while minimizing power consumption. Cyclone device families offer the optimal blend of high performance and low power in a low-cost FPGA.

Altera provides the Quartus Prime PowerPlay Power Analyzer to aid you during the design process by delivering fast and accurate estimations of power consumption. You can minimize power consumption, while taking advantage of the industry's leading FPGA performance, by using the tools and techniques described in this chapter.

Total FPGA power consumption is comprised of I/O power, core static power, and core dynamic power. This chapter focuses on design optimization options and techniques that help reduce core dynamic power and I/O power. In addition to these techniques, there are additional power optimization techniques available for specific devices. These techniques include:

- Programmable Power Technology
- Device Speed Grade Selection

Related Information

- **Literature and Technical Documentation**
  For more information about a device-specific architecture, refer to the device handbook on the Altera website.
- **PowerPlay Power Analysis**
  For more information about the PowerPlay Power Analyzer, refer to volume 3 of the *Quartus Prime Handbook*.
- **AN 514: Power Optimization in Stratix IV FPGAs**
  For more information about power optimization techniques available for Stratix IV devices.

Power Dissipation

You can refine techniques that reduce power consumption in your design by understanding the sources of power dissipation.

The following figure shows the power dissipation of Stratix and Cyclone devices in different designs. All designs were analyzed at a fixed clock rate of 100 MHz and exhibited varied logic resource utilization across available resources.

Figure 10-1: Average Core Dynamic Power Dissipation

Average Core Dynamic Power Dissipation by Block Type in Stratix III Devices at a 12.5% Toggle Rate (1)

- Global Clock Routing 14%
- Routing 30%
- Combinational Logic 16%
- Registered Logic 18%
- DSP Blocks 1% (3)
- Memory 21%

Average Core Dynamic Power Dissipation by Block Type in Cyclone III Devices at a 12.5% Toggle Rate (2)

- Global Clock Routing 16%
- Routing 29%
- Combinational Logic 11%
- Registered Logic 23%
- Multipliers 1% (3)
- Memory 20%

Notes:

1. 103 different designs were used to obtain these results.
2. 96 different designs were used to obtain these results.
3. In designs using DSP blocks, DSPs consumed 5% of core dynamic power.

In Stratix and Cyclone device families, a series of column and row interconnect wires of varying lengths provide signal interconnections between logic array blocks (LABs), memory block structures, and digital signal processing (DSP) blocks or multiplier blocks. These interconnects dissipate the largest component of device power.

FPGA combinational logic is another source of power consumption. The basic building block of logic in the latest Stratix series devices is the ALM, and in Cyclone IV GX devices, it is the logic element (LE).

For more information about ALMs and LEs in Cyclone or Stratix devices, refer to the respective device handbook.

Memory and clock resources are other major consumers of power in FPGAs. Stratix devices feature the TriMatrix memory architecture. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are configurable to support many features. Stratix IV TriMatrix on-chip memory is an enhancement based upon the Stratix II FPGA TriMatrix memory and includes three sizes of memory blocks: MLAB blocks, M9K blocks, and M144K blocks. Stratix IV and Stratix V devices feature Programmable Power Technology, an advanced architecture that enables a smooth trade-off between speed and power. The core of each Stratix IV and Stratix V device is divided into tiles, each of which may be put into a high-speed or low-power mode. The primary benefit of Programmable Power

Technology is to reduce static power, with a secondary benefit being a small reduction in dynamic power. Cyclone IV GX devices have 9-Kbit M9K memory blocks.

Design Space Explorer II

Design Space Explorer II (DSE) is an easy-to-use, self-guided design optimization utility that is included in the Quartus Prime software. DSE II explores and reports optimal Quartus Prime software options for your design, targeting either power optimization, design performance, or area utilization improvements. You can use DSE II to implement the techniques described in this chapter.

Figure 10-2: Design Space Explorer II User Interface
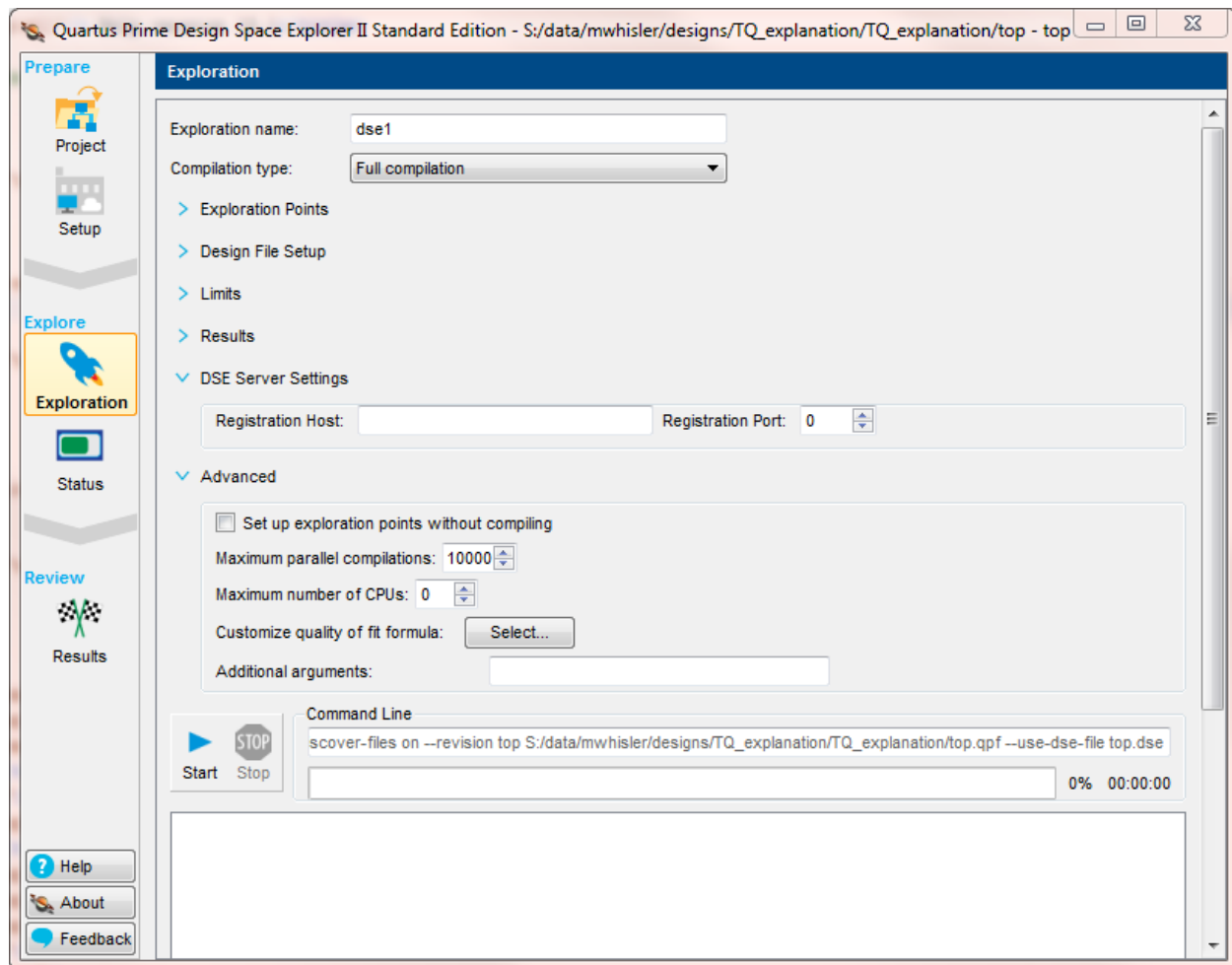
The power optimizations, under **Exploration mode**, target overall design power improvements. These settings focus on applying different options that specifically reduce total design thermal power.

By default, the Quartus Prime PowerPlay Power Analyzer is run for every exploration performed by DSE II when power optimizations are selected. This helps you debug your design and determine trade-offs between power requirements and performance optimization.

DSE II automatically tries different combinations of netlist optimizations and advanced Quartus Prime software compiler settings, and reports the best settings for your design, based on your chosen primary

optimization goal. You can try different seeds with DSE II if you are fairly close to meeting your timing or area requirements and find one seed that meets timing or area requirements. Finally, DSE II can run compilations on a remote compute farm, which shortens the timing closure process

- Name your DSE II session and specify the type of compilation to perform.
- Set **Exploration Points** and specify Exploration mode and the number and types of **Seeds** to use.
- Specify the **Design File Setup** including the use of a specified Quartus Archive File (**.qar**) or create a new one.
- Specify **Limits** to the operation of DSE II.
- Specify the type of **Results** to save.
- When using a remote compute farm, DSE II uses the values in the **DSE Server Settings** box to specify a registration host and network ports to connect.
- Options in the **Advanced** settings allow you to specify options such as:

  - Turn on the option to specify exploration points without compiling.
  - Specify the **Maximum number of parallel compilations** used by DSE II.
  - Specify the **Maximum number of CPUs** that can be used by DSE II.
  - Specify a quality of fit formula.

When you have completed your configuration, you can perform an exploration by clicking **Start**.

Related Information

Launch the Design Space Explorer II

For more information about the DSE II, refer to Quartus Prime Help.

Power-Driven Compilation

The standard Quartus Prime compilation flow consists of Analysis and Synthesis, placement and routing, Assembly, and Timing Analysis. Power-driven compilation takes place at the Analysis and Synthesis and Place-and-Route stages.

Quartus Prime software settings that control power-driven compilation are located in the **PowerPlay power optimization during synthesis** list in the **Advanced Settings (Synthesis)** dialog box, and the **PowerPlay power optimization during fitting** list on the **Advanced Fitter Settings** dialog box. The following sections describes these power optimization options at the Analysis and Synthesis and Fitter levels.

Power-Driven Synthesis

Synthesis netlist optimization occurs during the synthesis stage of the compilation flow. The optimization technique makes changes to the synthesis netlist to optimize your design according to the selection of area, speed, or power optimization. This section describes power optimization techniques at the synthesis level.

To access the PowerPlay Power Optimization During Synthesis option, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis).**

You can apply these settings on a project or entity level.

Table 10-1: Optimize Power During Synthesis Options

Settings	Description
Off	No netlist, placement, or routing optimizations are performed to minimize power.
Normal compilation (Default)	Low compute effort algorithms are applied to minimize power through netlist optimizations as long as they are not expected to reduce design performance.
Extra effort	High compute effort algorithms are applied to minimize power through netlist optimizations. Max performance might be impacted.

The **Normal compilation** setting is turned on by default. This setting performs memory optimization and power-aware logic mapping during synthesis.

Memory blocks can represent a large fraction of total design dynamic power. Minimizing the number of memory blocks accessed during each clock cycle can significantly reduce memory power. Memory optimization involves effective movement of user-defined read/write enable signals to associated read-and-write clock enable signals for all memory types.

A default implementation of a simple dual-port memory block in which write-clock enable signals and read-clock enable signals are connected to V_{CC}, making both read and write memory ports active during each clock cycle.

Figure 10-3: Memory Transformation
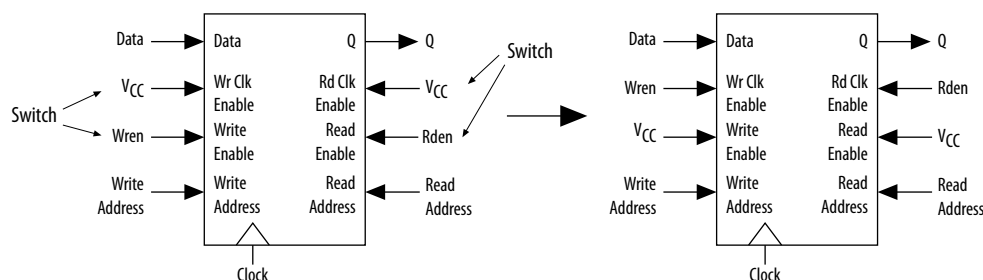
Memory transformation effectively moves the read-enable and write-enable signals to the respective read-clock enable and write-clock enable signals. By using this technique, memory ports are shut down when they are not accessed. This significantly reduces your design's memory power consumption. For Stratix IV and Stratix V devices, the memory transformation takes place at the Fitter level by selecting the **Normal compilation** settings for the power optimization option.

In Cyclone IV GX and Stratix IV devices, the specified read-during-write behavior can significantly impact the power of single-port and bidirectional dual-port RAMs. It is best to set the read-during-write parameter to "Don't care" (at the HDL level), as it allows an optimization whereby the read-enable signal can be set to the inversion of the existing write-enable signal (if one exists). This allows the core of the RAM to shut down (that is, not toggle), which saves a significant amount of power.

The other type of power optimization that takes place with the **Normal compilation** setting is power-aware logic mapping. The power-aware logic mapping reduces power by rearranging the logic during synthesis to eliminate nets with high toggle rates.

The **Extra effort** setting performs the functions of the **Normal compilation** setting and other memory optimizations to further reduce memory power by shutting down memory blocks that are not accessed. This level of memory optimization can require extra logic, which can reduce design performance.

The **Extra effort** setting also performs power-aware memory balancing. Power-aware memory balancing automatically chooses the best memory configuration for your memory implementation and provides optimal power saving by determining the number of memory blocks, decoder, and multiplexer circuits required. If you have not previously specified target-embedded memory blocks for your design's memory functions, the power-aware balancer automatically selects them during memory implementation.

The following figure is an example of a 4k × 4 (4k deep and 4 bits wide) memory implementation in two different configurations using M4K memory blocks available in some Stratix devices.

Figure 10-4: 4K × 4 Memory Implementation Using Multiple M4K Blocks
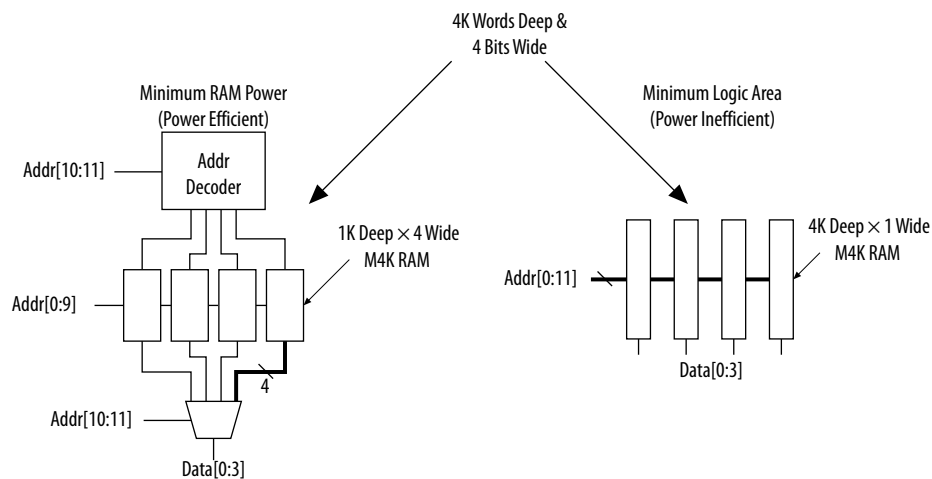
The minimum logic area implementation uses M4K blocks configured as 4k × 1. This implementation is the default in the Quartus Prime software because it has the minimum logic area (0 logic cells) and the highest speed. However, all four M4K blocks are active on each memory access in this implementation, which increases RAM power. The minimum RAM power implementation is created by selecting **Extra effort** in the **PowerPlay power optimization** list. This implementation automatically uses four M4K blocks configured as 1k × 4 for optimal power saving. An address decoder is implemented by the RAM megafunction to select which of the four M4K blocks should be activated on a given cycle, based on the state of the top two user address bits. The RAM megafunction automatically implements a multiplexer to feed the downstream logic by choosing the appropriate M4K output. This implementation reduces RAM power because only one M4K block is active on any cycle, but it requires extra logic cells, costing logic area and potentially impacting design performance.

There is a trade-off between power saved by accessing fewer memories and power consumed by the extra decoder and multiplexor logic. The Quartus Prime software automatically balances the power savings against the costs to choose the lowest power configuration for each logical RAM. The benchmark data shows that the power-driven synthesis can reduce memory power consumption by as much as 60% in Stratix devices.

Memory optimization options can also be controlled by the `Low_Power_Mode` parameter in the **Default Parameters** page of the **Settings** dialog box. The settings for this parameter are **None**, **Auto**, and **ALL**. **None** corresponds to the **Off** setting in the **PowerPlay power optimization** list. **Auto** corresponds to the **Normal compilation** setting and **ALL** corresponds to the **Extra effort** setting, respectively. You can apply PowerPlay power optimization either on a compiler basis or on individual entities. The `Low_Power_Mode`

parameter always takes precedence over the **Optimize Power for Synthesis** option for power optimization on memory.

You can also set the MAXIMUM_DEPTH parameter manually to configure the memory for low power optimization. This technique is the same as the power-aware memory balancer, but it is manual rather than automatic like the **Extra effort** setting in the **PowerPlay power optimization** list. You can set the MAXIMUM_DEPTH parameter for memory modules manually in the megafunction instantiation or in the IP Catalog for power optimization. The MAXIMUM_DEPTH parameter always takes precedence over the **Optimize Power for Synthesis** options for power optimization on memory optimization.

Related Information

[Reducing Memory Power Consumption](#) on page 10-11
For more information about clock enable signals.

Power-Driven Fitter

The **Compiler Settings** page provides access to **PowerPlay power optimization** settings.

You can apply these settings only on a project-wide basis. The **Extra effort** setting for the Fitter requires extensive effort to optimize the design for power and can increase the compilation time.

Table 10-2: Power-Driven Fitter Option

Settings	Description
Off	No netlist, placement, or routing optimizations are performed to minimize power.
Normal compilation (Default)	Low compute effort algorithms are applied to minimize power through placement and routing optimizations as long as they are not expected to reduce design performance.
Extra effort	High compute effort algorithms are applied to minimize power through placement and routing optimizations. Max performance might be impacted.

The **Normal compilation** setting is selected by default and performs DSP optimization by creating power-efficient DSP block configurations for your DSP functions. For Stratix IV and Stratix V devices, this setting, which is based on timing constraints entered for the design, enables the Programmable Power Technology to configure tiles as high-speed mode or low-power mode. Programmable Power Technology is always turned **ON** even when the **OFF** setting is selected for the **PowerPlay power optimization** option. Tiles are the combination of LAB and MLAB pairs (including the adjacent routing associated with LAB and MLAB), which can be configured to operate in high-speed or low-power mode. This level of power optimization does not have any affect on the fitting, timing results, or compile time.

The **Extra effort** setting performs the functions of the **Normal compilation** setting and other place-and-route optimizations during fitting to fully optimize the design for power. The Fitter applies an extra effort to minimize power even after timing requirements have been met by effectively moving the logic closer during placement to localize high-toggling nets, and using routes with low capacitance. However, this effort can increase the compilation time.

The **Extra effort** setting uses a Value Change Dump File (**.vcd**) that guides the Fitter to fully optimize the design for power, based on the signal activity of the design. The best power optimization during fitting results from using the most accurate signal activity information. Signal activities from full post-fit netlist (timing) simulation provide the highest accuracy because all node activities reflect the actual design behavior, provided that supplied input vectors are representative of typical design operation. If you do not have a **.vcd** file, the Quartus Prime software uses assignments, clock assignments, and vectorless estimation values (PowerPlay Power Analyzer Tool settings) to estimate the signal activities. This information is used to optimize your design for power during fitting. The benchmark data shows that the

power-driven Fitter technique can reduce power consumption by as much as 19% in Stratix devices. On average, you can reduce core dynamic power by 16% with the Extra effort synthesis and Extra effort fitting settings, as compared to the **Off** settings in both synthesis and Fitter options for power-driven compilation.

Note: Only the **Extra effort** setting in the **PowerPlay power optimization** list for the Fitter option uses the signal activities (from **.vcd** files) during fitting. The settings made in the **PowerPlay Power Analyzer Settings** page in the **Settings** dialog box are used to calculate the signal activity of your design.

Related Information

- **AN 514: Power Optimization in Stratix IV FPGAs**
  For more information about Stratix IV power optimization.
- **PowerPlay Power Analysis**
  For more information about **.vcd** files and how to create them, refer to the *Quartus Prime Handbook*.

Area-Driven Synthesis

Using area optimization rather than timing or delay optimization during synthesis saves power because you use fewer logic blocks. Using less logic usually means less switching activity. The Quartus Prime integrated synthesis tool provides **Speed**, **Balanced**, or **Area** for the **Optimization Technique** option. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families). The **Speed Optimization Technique** can increase the resource usage of your design if the constraints are too aggressive, and can also result in increased power consumption.

The benchmark data shows that the area-driven technique can reduce power consumption by as much as 31% in Stratix devices and as much as 15% in Cyclone devices.

Gate-Level Register Retiming

You can also use gate-level register retiming to reduce circuit switching activity. Retiming shuffles registers across combinational blocks without changing design functionality. The **Perform gate-level register retiming** option in the Quartus Prime software enables the movement of registers across combinational logic to balance timing, allowing the software to trade off the delay between timing critical and noncritical timing paths.

Retiming uses fewer registers than pipelining. In this example of gate-level register retiming, the 10 ns critical delay is reduced by moving the register relative to the combinational logic, resulting in the reduction of data depth and switching activity.

Figure 10-5: Gate-Level Register Retiming
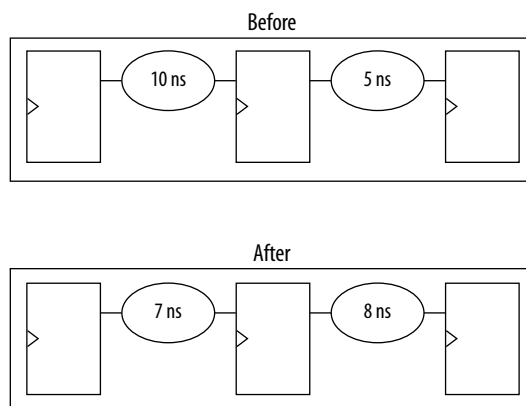
Note: Gate-level register retiming makes changes at the gate level. If you are using an atom netlist from a third-party synthesis tool, you must also select the **Perform WYSIWYG primitive resynthesis** option to undo the atom primitives to gates mapping (so that register retiming can be performed), and then to remap gates to Altera primitives. When using Quartus Prime integrated synthesis, retiming occurs during synthesis before the design is mapped to Altera primitives. The benchmark data shows that the combination of WYSIWYG remapping and gate-level register retiming techniques can reduce power consumption by as much as 6% in Stratix devices and as much as 21% in Cyclone devices.

Related Information

- **Netlist Optimizations and Physical Synthesis** on page 13-1
  For more information about register retiming, refer to the *Quartus Prime Handbook*.

Design Guidelines

Several low-power design techniques can reduce power consumption when applied during FPGA design implementation. This section provides detailed design techniques for Cyclone IV GXdevices that affect overall design power. The results of these techniques might be different from design to design.

Clock Power Management

Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. Actual clock-related power consumption is higher than this because the power consumed by local clock distribution within logic, memory, and DSP or multiplier blocks is included in the power consumption for the respective blocks.

Clock routing power is automatically optimized by the Quartus Prime software, which enables only those portions of the clock network that are required to feed downstream registers. Power can be further reduced by gating clocks when they are not required. It is possible to build clock-gating logic, but this approach is not recommended because it is difficult to generate a glitch free clock in FPGAs using ALMs or LEs.

Cyclone IV, Stratix IV, and Stratix V devices use clock control blocks that include an enable signal. A clock control block is a clock buffer that lets you dynamically enable or disable the clock network and dynamically switch between multiple sources to drive the clock network. You can use the Quartus Prime IP Catalog to create this clock control block with the ALTCLKCTRL megafunction. Cyclone IV,

Stratix IV, and Stratix V devices provide clock control blocks for global clock networks. In addition, Stratix IV, and Stratix V devices have clock control blocks for regional clock networks. The dynamic clock enable feature lets internal logic control the clock network. When a clock network is powered down, all the logic fed by that clock network does not toggle, thereby reducing the overall power consumption of the device. For example, the following shows a 4-input clock control block diagram.

Figure 10-6: Clock Control Block Diagram
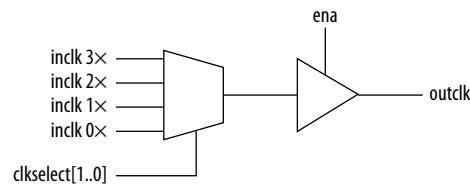
The enable signal is applied to the clock signal before being distributed to global routing. Therefore, the enable signal can either have a significant timing slack (at least as large as the global routing delay) or it can reduce the f_{MAX} of the clock signal.

Another contributor to clock power consumption is the LAB clock that distributes a clock to the registers within a LAB. LAB clock power can be the dominant contributor to overall clock power. For example, in Cyclone devices, each LAB can use two clocks and two clock enable signals, as shown in the following figure. Each LAB's clock signal and clock enable signal are linked. For example, an LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal.

Figure 10-7: LAB-Wide Control Signals
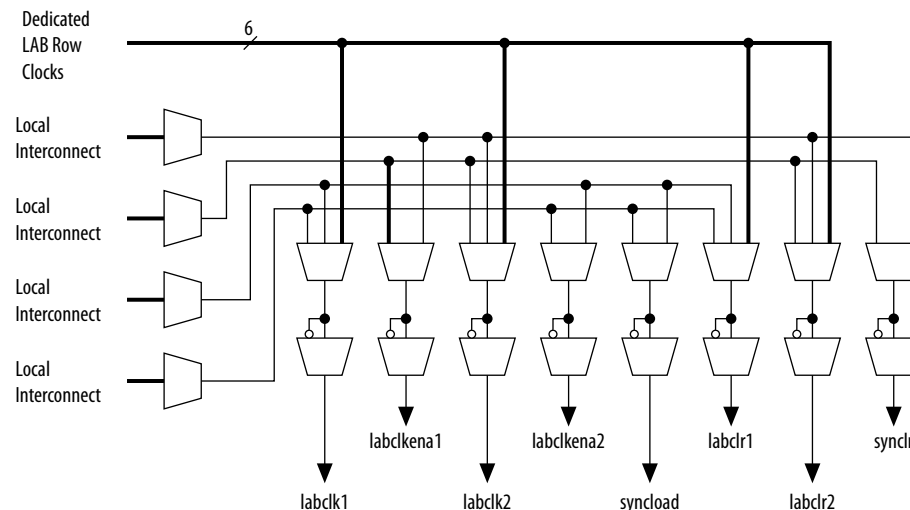
To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus Prime software automatically promotes register-level clock enable signals to the LAB-level. All registers within an LAB that share a common clock and clock enable are controlled by a shared gated clock. To take advantage of these clock enables, use a clock enable construct in the relevant HDL code for the registered logic.

Related Information

Clock Control Block Megafunction User Guide (ALTCLKCTRL)
For more information about using clock control blocks.

LAB-Wide Clock Enable Example

This VHDL code makes use of a LAB-wide clock enable. This clock-gating logic is automatically turned into an LAB-level clock enable signal.

```
IF clk'event AND clock = '1' THEN
    IF logic_is_enabled = '1' THEN
        reg <= value;
    ELSE
        reg <= reg;
    END IF;
END IF;
```

Reducing Memory Power Consumption

The memory blocks in FPGA devices can represent a large fraction of typical core dynamic power. Memory consumes approximately 20% of the core dynamic power in typical some device designs. Memory blocks are unlike most other blocks in the device because most of their power is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines.

Reducing Memory Power Consumption

The memory blocks in FPGA devices can represent a large fraction of typical core dynamic power. Memory consumes approximately 20% of the core dynamic power in typical some device designs. Memory blocks are unlike most other blocks in the device because most of their power is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines.

When a memory block is clocked, there is a sequence of timed events that occur within the block to execute a read or write. The circuitry controlled by the clock consumes the same amount of power regardless of whether or not the address or data has changed from one cycle to the next. Thus, the toggle rate of input data and the address bus have no impact on memory power consumption.

The key to reducing memory power consumption is to reduce the number of memory clocking events. You can achieve this through clock network-wide gating, or on a per-memory basis through use of the clock enable signals on the memory ports.

The logical view of the internal clock of the memory block. Use the appropriate enable signals on the memory to make use of the clock enable signal instead of gating the clock.

Figure 10-8: Memory Clock Enable Signal
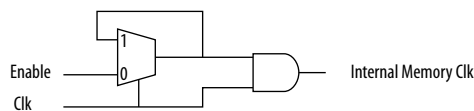
Using the clock enable signal enables the memory only when necessary and shuts it down for the rest of the time, reducing the overall memory power consumption. You can create these enable signals by selecting the **Clock enable signal** option for the appropriate port when generating the memory block function.

Figure 10-9: RAM 2-Port Clock Enable Signal Selectable Option
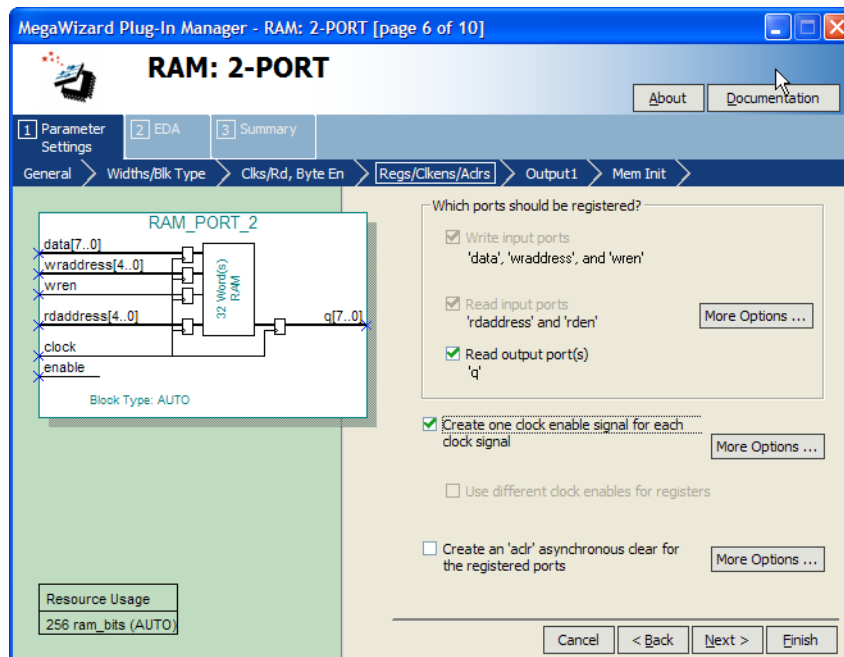
For example, consider a design that contains a 32-bit-wide M4K memory block in ROM mode that is running at 200 MHz. Assuming that the output of this block is only required approximately every four cycles, this memory block will consume 8.45 mW of dynamic power according to the demands of the downstream logic. By adding a small amount of control logic to generate a read clock enable signal for the memory block only on the relevant cycles, the power can be cut 75% to 2.15 mW.

You can also use the MAXIMUM_DEPTH parameter in your memory megafunction to save power in Cyclone IV GX, Stratix IV, and Stratix V devices; however, this approach might increase the number of LEs required to implement the memory and affect design performance.

You can set the MAXIMUM_DEPTH parameter for memory modules manually in the megafunction instantiation. The Quartus Prime software automatically chooses the best design memory configuration for optimal power.

Figure 10-10: RAM 2-Port Maximum Depth Selectable Option
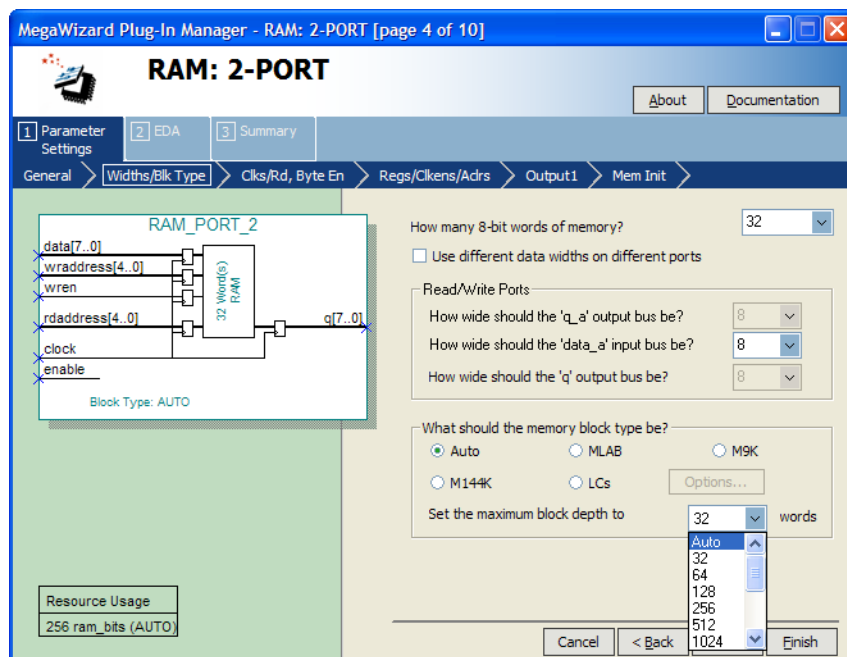
Related Information

- **Power-Driven Compilation** on page 10-4
- **Clock Power Management** on page 10-9
  For more information on clock network-wide gating.

Memory Power Reduction Example

Power usage measurements for a 4K × 36 simple dual-port memory implemented using multiple M4K blocks in a Stratix device. For each implementation, the M4K blocks are configured with a different memory depth.

Table 10-3: 4K × 36 Simple Dual-Port Memory Implemented Using Multiple M4K Blocks

M4K Configuration	Number of M4K Blocks	ALUTs
4K × 1 (Default setting)	36	0
2K × 2	36	40
1K × 4	36	62
512 × 9	32	143
256 × 18	32	302
128 × 36	32	633

Using the `MAXIMUM_DEPTH` parameter can save power. For all implementations, a user-provided read enable signal is present to indicate when read data is required. Using this power-saving technique can reduce power consumption by as much as 60%.

Send Feedback

Figure 10-11: Power Savings Using the MAXIMUM_DEPTH Parameter
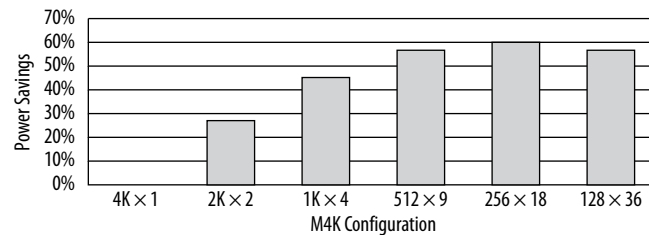
As the memory depth becomes more shallow, memory dynamic power decreases because unaddressed M4K blocks can be shut off using a decoded combination of address bits and the read enable signal. For a 128-deep memory block, power used by the extra LEs starts to outweigh the power gain achieved by using a more shallow memory block depth. The power consumption of the memory blocks and associated LEs depends on the memory configuration.

Note: The SOPC Builder and Qsys system do not offer specific power savings control for on-chip memory block. There is no read enable, write enable, or clock enable that you can enable in the on-chip RAM megafunction to shut down the RAM block in the SOPC Builder and Qsys system.

Pipelining and Retiming

Designs with many glitches consume more power because of faster switching activity. Glitches cause unnecessary and unpredictable temporary logic switches at the output of combinational logic. A glitch usually occurs when there is a mismatch in input signal timing leading to unequal propagation delay.

For example, consider an input change on one input of a 2-input XOR gate from 1 to 0, followed a few moments later by an input change from 0 to 1 on the other input. For a moment, both inputs become 1 (high) during the state transition, resulting in 0 (low) at the output of the XOR gate. Subsequently, when the second input transition takes place, the XOR gate output becomes 1 (high). During signal transition, a glitch is produced before the output becomes stable.

Figure 10-12: XOR Gate Showing Glitch at the Output
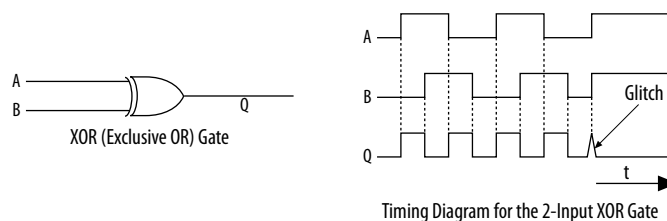
This glitch can propagate to subsequent logic and create unnecessary switching activity, increasing power consumption. Circuits with many XOR functions, such as arithmetic circuits or cyclic redundancy check (CRC) circuits, tend to have many glitches if there are several levels of combinational logic between registers.

Pipelining can reduce design glitches by inserting flipflops into long combinational paths. Flipflops do not allow glitches to propagate through combinational paths. Therefore, a pipelined circuit tends to have less glitching. Pipelining has the additional benefit of generally allowing higher clock speed operations, although it does increase the latency of a circuit (in terms of the number of clock cycles to a first result).

An example where pipelining is applied to break up a long combinational path.

Figure 10-13: Pipelining Example

Non-Pipelined

Combinational
Logic

Long Logic
Depth

Pipelined

Combinational
Logic

Short Logic
Depth

Combinational
Logic

Short Logic
Depth
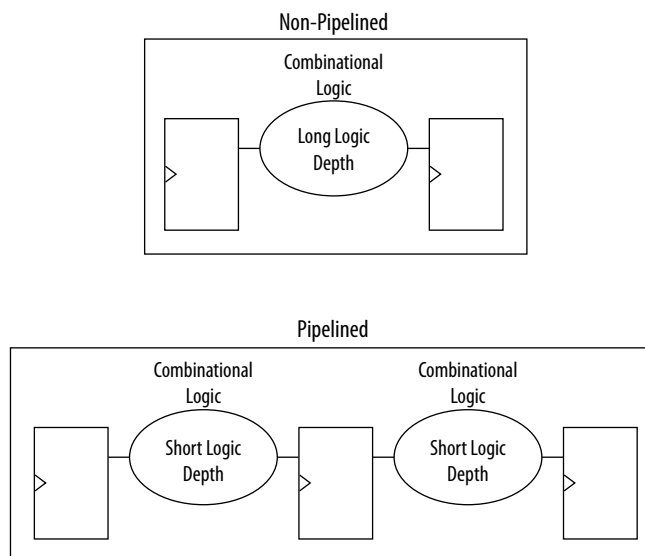
Pipelining is very effective for glitch-prone arithmetic systems because it reduces switching activity, resulting in reduced power dissipation in combinational logic. Additionally, pipelining allows higher-speed operation by reducing logic-level numbers between registers. The disadvantage of this technique is that if there are not many glitches in your design, pipelining can increase power consumption by adding unnecessary registers. Pipelining can also increase resource utilization. The benchmark data shows that pipelining can reduce dynamic power consumption by as much as 30% in Cyclone and Stratix devices.

Architectural Optimization

You can use design-level architectural optimization by taking advantage of specific device architecture features. These features include dedicated memory and DSP or multiplier blocks available in FPGA devices to perform memory or arithmetic-related functions. You can use these blocks in place of LUTs to reduce power consumption. For example, you can build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

The Stratix device family allows you to efficiently target small, medium, and large memories with the TriMatrix memory architecture. Each TriMatrix memory block is optimized for a specific function. M512 memory blocks are more power-efficient than the distributed memory structures in some competing FPGAs. The M4K memory blocks are used to implement buffers for a wide variety of applications, including processor code storage, large look-up table implementation, and large memory applications. The M-RAM blocks are useful in applications where a large volume of data must be stored on-chip. Effective utilization of these memory blocks can have a significant impact on power reduction in your design.

The latest Stratix and Cyclone device families have configurable M9K memory blocks that provide various memory functions such as RAM, FIFO buffers, and ROM.

Related Information

- **Area and Timing Optimization** on page 9-1
  For more information about using DSP and memory blocks efficiently, refer to the *Quartus Prime Handbook*.

I/O Power Guidelines

Nonterminated I/O standards such as LVTTL and LVCMOS have a rail-to-rail output swing. The voltage difference between logic-high and logic-low signals at the output pin is equal to the V_{CCIO} supply voltage. If the capacitive loading at the output pin is known, the dynamic power consumed in the I/O buffer can be calculated.

$P = 0.5 \times F \times C \times V^2$

In this equation, F is the output transition frequency and C is the total load capacitance being switched. V is equal to V_{CCIO} supply voltage. Because of the quadratic dependence on V_{CCIO}, lower voltage standards consume significantly less dynamic power.

Transistor-to-transistor logic (TTL) I/O buffers consume very little static power. As a result, the total power consumed by a LVTTL or LVCMOS output is highly dependent on load and switching frequency.

When using resistively terminated I/O standards like SSTL and HSTL, the output load voltage swings by a small amount around some bias point. The same dynamic power equation is used, where V is the actual load voltage swing. Because this is much smaller than V_{CCIO}, dynamic power is lower than for nonterminated I/O under similar conditions. These resistively terminated I/O standards dissipate significant static (frequency-independent) power, because the I/O buffer is constantly driving current into the resistive termination network. However, the lower dynamic power of these I/O standards means they often have lower total power than LVCMOS or LVTTL for high-frequency applications. Use the lowest drive strength I/O setting that meets your speed and waveform requirements to minimize I/O power when using resistively terminated standards.

You can save a small amount of static power by connecting unused I/O banks to the lowest possible V_{CCIO} voltage of 1.2 V.

For more information about I/O standards, refer to the *Stratix IV Device Handbook*, or the *Cyclone IV GX Handbook* on the Altera website.

When calculating I/O power, the PowerPlay Power Analyzer uses the default capacitive load set for the I/O standard in the **Capacitive Loading** page of the **Device and Pin Options** dialog box. Any other components defined in the board trace model are not taken into account for the power measurement.

For Cyclone IV GX, Stratix IV, and Stratix V, devices, Advanced I/O Timing is always used, which uses the full board trace model.

Related Information

Dynamically Controlled On-Chip Terminations

Stratix IV and Stratix V FPGAs offer dynamic on-chip termination (OCT). Dynamic OCT enables series termination (RS) and parallel termination (RT) to dynamically turn on/off during the data transfer. This feature is especially useful when Stratix IV and Stratix V FPGAs are used with external memory interfaces, such as interfacing with DDR memories.

Compared to conventional termination, dynamic OCT reduces power consumption significantly as it eliminates the constant DC power consumed by parallel termination when transmitting data. Parallel termination is extremely useful for applications that interface with external memories where I/O standards, such as HSTL and SSTL, are used. Parallel termination supports dynamic OCT, which is useful for bidirectional interfaces.

The following is an example of power saving for a DDR3 interface using on-chip parallel termination.

The static current consumed by parallel OCT is equal to the V_{CCIO} voltage divided by 100 W . For DDR3 interfaces that use SSTL-15, the static current is 1.5 V/100 W = 15 mA per pin. Therefore, the static power is 1.5 V ×15 mA = 22.5 mW. For an interface with 72 DQ and 18 DQS pins, the static power is 90 pins × 22.5 mW = 2.025 W. Dynamic parallel OCT disables parallel termination during write operations, so if writing occurs 50% of the time, the power saved by dynamic parallel OCT is 50% × 2.025 W = 1.0125 W.

Related Information

Stratix IV Device I/O Features

For more information about dynamic OCT in Stratix IV devices, refer to the chapter in the *Stratix IV Device Handbook*.

Power Optimization Advisor

The Quartus Prime software includes the Power Optimization Advisor, which provides specific power optimization advice and recommendations based on the current design project settings and assignments. The advisor covers many of the suggestions listed in this chapter. The following example shows how to reduce your design power with the Power Optimization Advisor.

Power Optimization Advisor Example

After compiling your design, run the PowerPlay Power Analyzer to determine your design power and to see where power is dissipated in your design. Based on this information, you can run the Power Optimization Advisor to implement recommendations that can reduce design power.

The Power Optimization Advisor after compiling a design that is not fully optimized for power.

Figure 10-14: Power Optimization Advisor
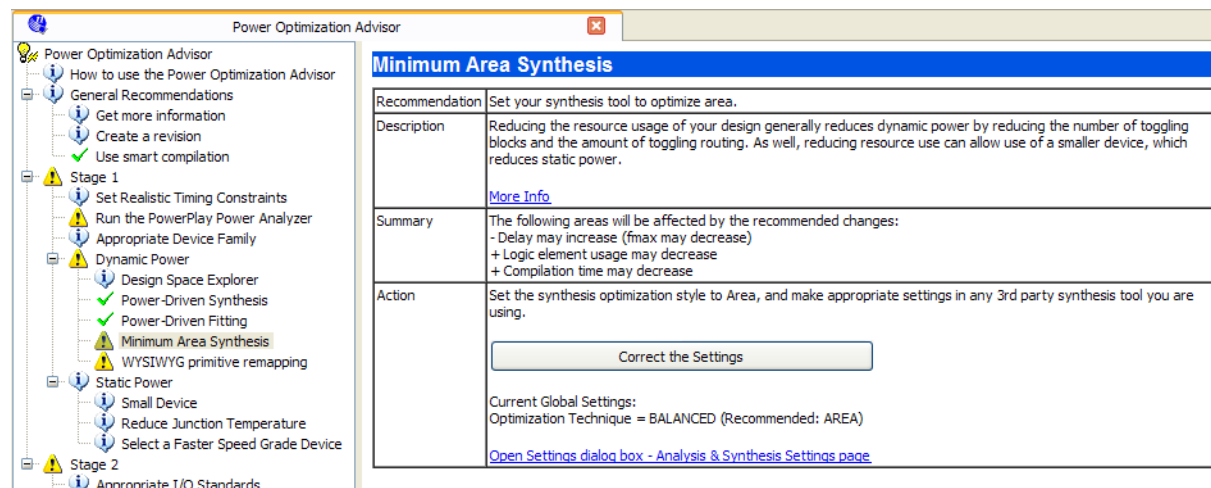
The Power Optimization Advisor shows the recommendations that can reduce power in your design. The recommendations are split into stages to show the order in which you should apply the recommended settings. The first stage shows mostly CAD setting options that are easy to implement and highly effective in reducing design power. An icon indicates whether each recommended setting is made in the current project. The checkmark icons for Stage 1 shows the recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icon shows the general suggestions. Each recommendation includes the description, summary of the effect of the recommendation, and the action required to make the appropriate setting.

There is a link from each recommendation to the appropriate location in the Quartus Prime user interface where you can change the setting. After making the recommended changes, recompile your design. The Power Optimization Advisor indicates with green check marks that the recommendations were implemented successfully. You can use the PowerPlay Power Analyzer to verify your design power results.

Figure 10-15: Implementation of Power Optimization Advisor Recommendations
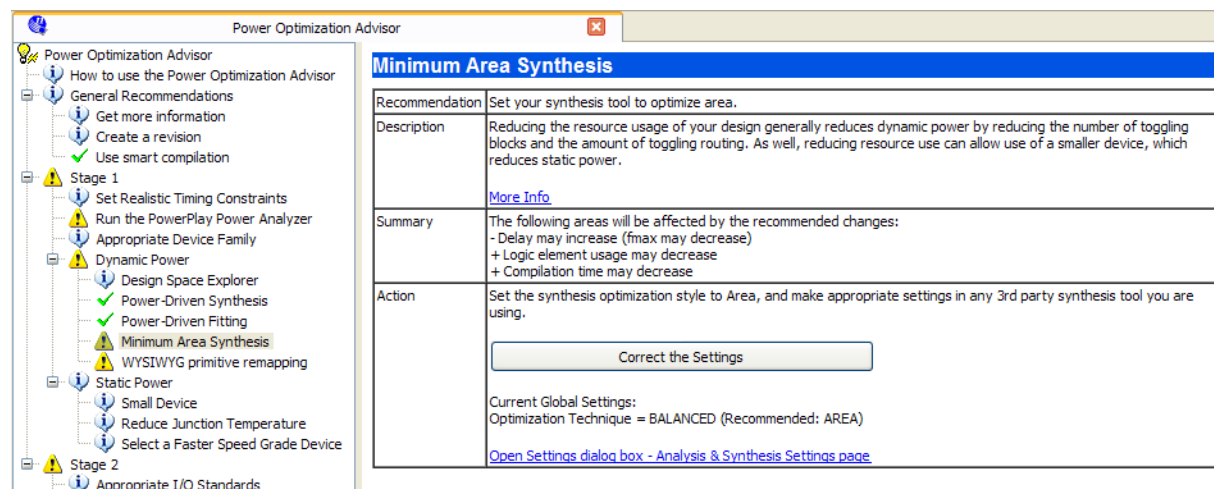
The recommendations listed in Stage 2 generally involve design changes, rather than CAD settings changes as in Stage 1. You can use these recommendations to further reduce your design power consumption. Altera recommends that you implement Stage 1 recommendations first, then the Stage 2 recommendations.

Document Revision History

Table 10-4: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*. • Updated screenshot for DSE II GUI. • Added information about remote hosts for DSE II.
2015.05.04	15.0.0	
2014.12.15	14.1.0	• Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. • Updated DSE II GUI and optimization settings.
2014.06.30	14.0.0	Updated the format.
May 2013	13.0.0	Added a note to "Memory Power Reduction Example" on Qsys and SOPC Builder power savings limitation for on-chip memory block.
June 2012	12.0.0	Removed survey link.

Date	Version	Changes
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	• Was chapter 11 in the 9.1.0 release • Updated Figures 14-2, 14-3, 14-6, 14-18, 14-19, and 14-20 • Updated device support • Minor editorial updates
November 2009	9.1.0	• Updated Figure 11-1 and associated references • Updated device support • Minor editorial update
March 2009	9.0.0	• Was chapter 9 in the 8.1.0 release • Updated for the Quartus Prime software release • Added benchmark results • Removed several sections • Updated Figure 13–1, Figure 13–17, and Figure 13–18
November 2008	8.1.0	• Changed to 8½" × 11" page size • Changed references to altsyncram to RAM • Minor editorial updates
May 2008	8.0.0	• Added support for Stratix IV devices • Updated Table 9–1 and 9–9 • Updated "Architectural Optimization" on page 9–22 • Added "Dynamically-Controlled On-Chip Terminations" on page 9–26 • Updated "Referenced Documents" on page 9–29 • Updated references

Related Information

Quartus Handbook Archive

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

This chapter describes techniques to reduce resource usage when designing for Altera devices.

Resource Utilization

Determining device utilization is important regardless of whether your design achieved a successful fit. If your compilation results in a no-fit error, resource utilization information is important for analyzing the fitting problems in your design.If your fitting is successful, review the resource utilization information to determine whether the future addition of extra logic or other design changes might introduce fitting difficulties. Also, review the resource utilization information to determine if it is impacting timing performance.

To determine resource usage, refer to the **Flow Summary** section of the Compilation Report. This section reports resource utilization, including pins, memory bits, digital signal processing (DSP) blocks, and phase-locked loops (PLLs). **Flow Summary** indicates whether your design exceeds the available device resources. More detailed information is available by viewing the reports under **Resource Section** in the **Fitter** section of the Compilation Report.

Flow Summary shows the overall logic utilization. The Fitter can spread logic throughout the device, which may lead to higher overall utilization.

As the device fills up, the Fitter automatically searches for logic functions with common inputs to place in one ALM. The number of packed registers also increases. Therefore, a design that has high overall utilization might still have space for extra logic if the logic and registers can be packed together more tightly.

The reports under the **Resource Section** in the **Fitter** section of the Compilation Report provide more detailed resource information. The Fitter Resource Usage Summary report breaks down the logic utilization information and provides other resource information, including the number of bits in each type of memory block. This panel also contains a summary of the usage of global clocks, PLLs, DSP blocks, and other device-specific resources.

You can also view reports describing some of the optimizations that occurred during compilation. For example, if you use Quartus Prime integrated synthesis, the reports in the Optimization Results folder in the **Analysis & Synthesis** section include information about registers that integrated synthesis removed during synthesis. Use this report to estimate device resource utilization for a partial design to ensure that registers were not removed due to missing connections with other parts of the design.

If a specific resource usage is reported as less than 100% and a successful fit cannot be achieved, either there are not enough routing resources or some assignments are illegal. In either case, a message appears in the **Processing** tab of the Messages window describing the problem.

If the Fitter finishes unsuccessfully and runs much faster than on similar designs, a resource might be over-utilized or there might be an illegal assignment. If the Quartus Prime software seems to run for an excessively long time compared to runs on similar designs, a legal placement or route probably cannot be found. In the Compilation Report, look for errors and warnings that indicate these types of problems.

You can use the Chip Planner to find areas of the device that have routing congestion on specific types of routing resources. If you find areas with very high congestion, analyze the cause of the congestion. Issues such as high fan-out nets not using global resources, an improperly chosen optimization goal (speed versus area), very restrictive floorplan assignments, or the coding style can cause routing congestion. After you identify the cause, modify the source or settings to reduce routing congestion.

Related Information

Fitter Resources Report
For more information about Fitter Resources Report

Analyzing and Optimizing the Design Floorplan with the Chip Planner on page 12-1
For details about using the Chip Planner tool

Optimizing Resource Utilization (LUT-Based Devices)

The following lists the stages after design analysis:

- Optimize resource utilization—Ensure that you have already set the basic constraints
- I/O timing optimization—Optimize I/O timing after you optimize resource utilization and your design fits in the desired target device
- Register-to-register timing optimization

Related Information

- **Design Optimization Overview** on page 7-1
  Provides information about setting basic constraints
- **Timing Closure and Optimization** on page 9-1
  Provides information about optimizing I/O timing. These tips are valid for all FPGA families and the MAX II family of CPLDs.

Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize resource usage. To run the Resource Optimization Advisor, on the Tools menu, point to **Advisors**, and click **Resource Optimization Advisor**.

The Resource Optimization Advisor provides step-by-step advice about how to optimize resource usage (logic element, memory block, DSP block, I/O, and routing) of your design. Some of the recommenda-

tions in these categories might conflict with each other. Altera recommends evaluating the options and choosing the settings that best suit your requirements.

Related Information

Resource Optimization Advisor Command Tools Menu

For more information about the Resource Optimization Advisor

Resolving Resource Utilization Issues Summary

Resource utilization issues can be divided into the following three categories:

Table 11-1: Resource Utilization Issues

Types of Resource Utilization Issues	Refer to
Issues relating to I/O pin utilization or placement, including dedicated I/O blocks such as PLLs or LVDS transceivers	**I/O Pin Utilization or Placement** on page 11-3
Issues relating to logic utilization or placement, including logic cells containing registers and LUTs as well as dedicated logic, such as memory blocks and DSP blocks	**Logic Utilization or Placement** on page 11-4
Issues relating to routing	**Routing** on page 11-8

Related Information

- **I/O Pin Utilization or Placement** on page 11-3
- **Logic Utilization or Placement** on page 11-4
- **Routing** on page 11-8

I/O Pin Utilization or Placement

Resolve I/O resource problems with these guidelines.

Guideline: Use I/O Assignment Analysis

To help with pin placement, on the Processing menu, point to **Start** and click **Start I/O Assignment Analysis**. The **Start I/O Assignment Analysis** command allows you to check your I/O assignments early in the design process. You can use this command to check the legality of pin assignments before, during, or after compilation of your design. If design files are available, you can use this command to accomplish more thorough legality checks on your design's I/O pins and surrounding logic. These checks include proper reference voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

Common issues with I/O placement relate to the fact that differential standards have specific pin pairings and certain I/O standards might be supported only on certain I/O banks.

If your compilation or I/O assignment analysis results in specific errors relating to I/O pins, follow the recommendations in the error message. Right-click the message in the Messages window and click **Help** to open the Quartus Prime Help topic for this message.

Guideline: Modify Pin Assignments or Choose a Larger Package

If a design that has pin assignments fails to fit, compile the design without the pin assignments to determine whether a fit is possible for the design in the specified device and package. You can use this approach if a Quartus Prime error message indicates fitting problems due to pin assignments.

If the design fits when all pin assignments are ignored or when several pin assignments are ignored or moved, you might have to modify the pin assignments for the design or select a larger package.

If the design fails to fit because insufficient I/Os pins are available, a successful fit can often be obtained by using a larger device package (which can be the same device density) that has more available user I/O pins.

Related Information

- **I/O Management** on page 2-1
  For more information about I/O assignment analysis

Logic Utilization or Placement

Resolve logic resource problems, including logic cells containing registers and LUTs, as well as dedicated logic such as memory blocks and DSP blocks, with these guidelines.

Guideline: Optimize Source Code

If your design does not fit because of logic utilization, then evaluate and modify the design at the source. You can often improve logic significantly by making design-specific changes to your source code. This is typically the most effective technique for improving the quality of your results.

If your design does not fit into available logic elements (LEs) or ALMs, but you have unused memory or DSP blocks, check if you have code blocks in your design that describe memory or DSP functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to allow these functions to be placed into dedicated memory or DSP resources in the target device.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Quartus Prime software, you can check for the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including the state encoding for each state machine that was recognized during compilation. If your state machine is not being recognized, you might have to change your source code to enable it to be recognized.

Related Information

- **Recommended HDL Coding Styles**
  For coding style guidelines, including examples of HDL code for inferring memory and DSP functions and sample HDL code for state machines
- **AN 584: Timing Closure Methodology for Advanced FPGA Designs.**
  For additional HDL coding examples

Guideline: Optimize Synthesis for Area, Not Speed

If your design fails to fit because it uses too much logic, resynthesize the design to improve the area utilization. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Particularly when area utilization of the design is a concern, ensure that you do not over-constrain the timing requirements for the design. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is an important concern, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus Prime integrated synthesis, select **Balanced** or **Area** for the **Optimization Technique**. You can also specify an **Optimization Technique** logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Speed**. You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.

In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization.

Note: In the Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.

The Quartus Prime software provides additional attributes and options that can help improve the quality of your synthesis results.

Related Information

Synthesis
For information about setting the timing requirements and synthesis options in Quartus Prime integrated synthesis and other synthesis tools

Guideline: Restructure Multiplexers

Multiplexers form a large portion of the logic utilization in many FPGA designs. By optimizing your multiplexed logic, you can achieve a more efficient implementation in your Altera device.

Related Information

- **Restructure Multiplexers logic option**
  For more information about the Restructure Multiplexers option
- **Recommended HDL Coding Styles**
  For design guidelines to achieve optimal resource utilization for multiplexer designs

Guideline: Perform WYSIWYG Primitive Resynthesis with Balanced or Area Setting

The **Perform WYSIWYG Primitive Resynthesis** logic option specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the **Optimization Technique** logic option. The **Perform WYSIWYG Primitive Resynthesis** logic option is useful for resynthesizing some or all of the WYSIWYG primitives in your design for better area or performance. However, WYSIWYG primitive resynthesis can be done only when you use third-party synthesis tools.

Note: The **Balanced** setting typically produces utilization results that are very similar to the **Area** setting with better performance results. The **Area** setting can give better results in some cases. Performing WYSIWYG resynthesis for area in this way typically reduces register-to-register timing performance.

Related Information

Perform WYSIWYG Primitive Resynthesis logic option
For information about this logic option

Guideline: Use Register Packing

The **Auto Packed Registers** option implements the functions of two cells into one logic cell by combining the register of one cell in which only the register is used with the LUT of another cell in which only the LUT is used.

Related Information
Auto Packed Registers logic option
For more information about the Auto Packed Registers logic option

Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or LogicLock Plus assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or LogicLock Plus region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or LogicLock Plus region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and LogicLock Plus assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove LogicLock Plus assignments in the Chip Planner, in the LogicLock Plus Regions Window, or on the Assignments menu, click **Remove Assignments.** Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

Related Information

- **Analyzing and Optimizing the Design Floorplan with the Chip Planner** on page 12-1
  For more information about the **Routing Congestion** task in the Chip Planner

Guideline: Flatten the Hierarchy During Synthesis

Synthesis tools typically provide the option of preserving hierarchical boundaries, which can be useful for verification or other purposes. However, the Quartus Prime software optimizes across hierarchical boundaries so as to perform the most logic minimization, which can reduce area in a design with no design partitions.

Guideline: Retarget Memory Blocks

If your design fails to fit because it runs out of device memory resources, your design may require a certain type of memory that the device does not have. For example, a design that requires two M-RAM blocks cannot be targeted to a device with only one M-RAM block. You might be able to obtain a fit by building one of the memories with a different size memory block, such as an M4K memory block.

If the memory block was created with a parameter editor, open the parameter editor and edit the RAM block type so it targets a new memory block size.

ROM and RAM memory blocks can also be inferred from your HDL code, and your synthesis software can place large shift registers into memory blocks by inferring the Shift register (RAM-based) IP core. This inference can be turned off in your synthesis tool to cause the memory or shift registers to be placed in logic instead of in memory blocks. Also, for improved timing performance, you can turn this inference off to prevent registers from being moved into RAM.

Depending on your synthesis tool, you can also set the RAM block type for inferred memory blocks. In Quartus Prime integrated synthesis, set the **ramstyle** attribute to the desired memory type for the inferred

RAM blocks, or set the option to **logic,** to implement the memory block in standard logic instead of a memory block.

Consider the Resource Utilization by Entity report in the report file and determine whether there is an unusually high register count in any of the modules. Some coding styles can prevent the Quartus Prime software from inferring RAM blocks from the source code because of the blocks' architectural implementation, and force the software to implement the logic in flipflops. As an example, a function such as an asynchronous reset on a register bank might make the resistor bank incompatible with the RAM blocks in the device architecture, so that the register bank is implemented in flipflops. It is often possible to move a large register bank into RAM by slight modification of associated logic.

Guideline: Use Physical Synthesis Options to Reduce Area

The physical synthesis options for fitting help you decrease resource usage. When you enable these options, the Quartus Prime software makes placement-specific changes to the netlist that reduce resource utilization for a specific Altera device.

Note: The compilation time might increase considerably when you use physical synthesis options.

With the Quartus Prime software, you can apply physical synthesis options to specific instances, which can reduce the impact on compilation time. Physical synthesis instance assignments allow you to enable physical synthesis algorithms for specific portions of your design.

The following physical synthesis optimizations for fitting are available:

- Physical synthesis for combinational logic
- Map logic into memory

Related Information

Physical Synthesis Optimizations Page Settings Dialog Box

Guideline: Retarget or Balance DSP Blocks

A design might not fit because it requires too many DSP blocks. You can implement all DSP block functions with logic cells, so you can retarget some of the DSP blocks to logic to obtain a fit.

If the DSP function was created with the parameter editor, open the parameter editor and edit the function so it targets logic cells instead of DSP blocks. The Quartus Prime software uses the `DEDICATED_MULTIPLIER_CIRCUITRY` IP core parameter to control the implementation.

DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. You can turn off this inference in your synthesis tool. When you are using Quartus Prime integrated synthesis, you can disable inference by turning off the **Auto DSP Block Replacement** logic option for your entire project. Click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)**. Turn off **Auto DSP Block Replacement**. Alternatively, you can disable the option for a specific block with the Assignment Editor.

The Quartus Prime software also offers the **DSP Block Balancing** logic option, which implements DSP block elements in logic cells or in different DSP block modes. The default **Auto** setting allows DSP block balancing to convert the DSP block slices automatically as appropriate to minimize the area and maximize the speed of the design. You can use other settings for a specific node or entity, or on a project-wide basis, to control how the Quartus Prime software converts DSP functions into logic cells and DSP blocks. Using any value other than **Auto** or **Off** overrides the `DEDICATED_MULTIPLIER_CIRCUITRY` parameter used in IP core variations.

Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

Routing

Resolve routing resource problems with these guidelines.

Guideline: Set Auto Packed Registers to Sparse or Sparse Auto

The **Auto Packed Registers** option reduces LE or ALM count in a design. You can set this option by clicking **Assignment** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

Related Information

Auto Packed Registers logic option

Guideline: Set Fitter Aggressive Routability Optimizations to Always

The **Fitter Aggressive Routability Optimization** option is useful if your design does not fit due to excessive routing wire utilization.

If there is a significant imbalance between placement and routing time (during the first fitting attempt), it might be because of high wire utilization. Turning on the **Fitter Aggressive Routability Optimizations** option can reduce your compilation time.

On average, this option can save up to 6% wire utilization, but can also reduce performance by up to 4%, depending on the device.

Related Information

Fitter Aggressive Routability Optimizations logic option

Guideline: Increase Router Effort Multiplier

The Router Effort Multiplier controls how quickly the router tries to find a valid solution. The default value is 1.0 and legal values must be greater than 0. Numbers higher than 1 help designs that are difficult to route by increasing the routing effort. Numbers closer to 0 (for example, 0.1) can reduce router runtime, but usually reduce routing quality slightly. Experimental evidence shows that a multiplier of 3.0 reduces overall wire usage by approximately 2%. Using a Router Effort Multiplier higher than the default value could be beneficial for designs with complex datapaths with more than five levels of logic. However, congestion in a design is primarily due to placement, and increasing the Router Effort Multiplier does not necessarily reduce congestion.

Note: Any Router Effort Multiplier value greater than 4 only increases by 10% for every additional 1. For example, a value of 10 is actually 4.6.

Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to achieve may not fit the targeted device. Conflicting or difficult-to-achieve constraints can occur when location or LogicLock Plus assignments are too strict and there are not enough routing resources.

In this case, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove all location and LogicLock Plus region assignments from that area. If the local constraints are removed, and the design still does not fit, the design is over-constrained. To correct the problem, remove all location and LogicLock Plus assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove LogicLock Plus assignments in the Chip Planner, in the LogicLock Plus Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

Related Information

- **Analyzing and Optimizing the Design Floorplan with the Chip Planner** on page 12-1
  For more information about the Routing Congestion task in the Chip Planner

Guideline: Optimize Synthesis for Area, Not Speed

In some cases, resynthesizing the design to improve the area utilization can also improve the routability of the design. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Ensure that you do not overconstrain the timing requirements for the design, particularly when the area utilization of the design is a concern. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is important to improve the routing results in your design, some synthesis tools offer an easy way to optimize for area instead of speed. If you are using Quartus Prime integrated synthesis, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)**. For **Optimization Technique**, select **Balanced** or **Area**.

You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance). You can apply the setting to specific modules while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Speed**. You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.

Note: In the Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those obtained with the **Area** setting, with better performance results. The **Area** setting can yield better results in some unusual cases.

In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization, which can improve routability.

Related Information

Synthesis
For information about setting the timing requirements and synthesis options in Quartus Prime integrated synthesis and other synthesis tools

Guideline: Optimize Source Code

If your design does not fit because of routing problems and the methods described in the preceding sections do not sufficiently improve the routability of the design, modify the design at the source to achieve the desired results. You can often improve results significantly by making design-specific changes to your source code, such as duplicating logic or changing the connections between blocks that require significant routing resources.

Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to

the Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> \
-to <instance name>
```

Note: If the *<value>* field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

Related Information

Tcl Scripting on page 5-1
For more information about Tcl scripting

Quartus Prime Settings File Manual
For more information about all settings and constraints in the Quartus Prime software

Command-Line Scripting on page 4-1
For more information about command-line scripting

Initial Compilation Settings

Use the Quartus Prime Settings File (**.qsf**) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 11-2: Advanced Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Placement Effort Multiplier	`PLACEMENT_EFFORT_MULTIPLIER`	Any positive, non-zero value	Global
Router Effort Multiplier	`ROUTER_EFFORT_MULTIPLIER`	Any positive, non-zero value	Global
Router Timing Optimization level	`ROUTER_TIMING_OPTIMIZATION_LEVEL`	`NORMAL`, `MINIMUM`, `MAXIMUM`	Global
Final Placement Optimization	`FINAL_PLACEMENT_OPTIMIZATION`	`ALWAYS`, `AUTOMATICALLY`, `NEVER`	Global

Resource Utilization Optimization Techniques (LUT-Based Devices)

This table lists the **.qsf** file variable name and applicable values for the settings described in **Optimizing Resource Utilization (LUT-Based Devices)** on page 11-2.

Table 11-3: Resource Utilization Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers [1]	`AUTO_PACKED_REGISTERS_ <device family name>`	`OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, AUTO`	Global, Instance
Perform WYSIWYG Primitive Resynthesis	`ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP`	`ON, OFF`	Global, Instance
Physical Synthesis for Combinational Logic for Reducing Area	`PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA`	`ON, OFF`	Global, Instance
Physical Synthesis for Mapping Logic to Memory	`PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_ FOR AREA`	`ON, OFF`	Global, Instance
Optimization Technique	`<device family name>_OPTIMIZATION_ TECHNIQUE`	`AREA, SPEED, BALANCED`	Global, Instance
Speed Optimization Technique for Clock Domains	`SYNTH_CRITICAL_CLOCK`	`ON, OFF`	Instance
State Machine Encoding	`STATE_MACHINE_PROCESSING`	`AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER- ENCODE`	Global, Instance
Auto RAM Replacement	`AUTO_RAM_RECOGNITION`	`ON, OFF`	Global, Instance
Auto ROM Replacement	`AUTO_ROM_RECOGNITION`	`ON, OFF`	Global, Instance
Auto Shift Register Replacement	`AUTO_SHIFT_REGISTER_RECOGNITION`	`ON, OFF`	Global, Instance

[1] Allowed values for this setting depend on the device family that you select.

Setting Name	.qsf File Variable Name	Values	Type
Auto Block Replacement	AUTO_DSP_RECOGNITION	ON, OFF	Global, Instance
Number of Processors for Parallel Compilation	NUM_PARALLEL_PROCESSORS	Integer between 1 and 16 inclusive, or ALL	Global

Document Revision History

Table 11-4: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
2014.12.15	14.1.0	Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.
June 2014	14.0.0	• Removed Cyclone III and Stratix III devices references. • Removed Macrocell-Based CPLDs related information. • Updated template.
May 2013	13.0.0	Initial release.

Analyzing and Optimizing the Design Floorplan with the Chip Planner

As FPGA designs grow larger in density, the ability to analyze the design for performance, routing congestion, and logic placement is critical to meet the design requirements. This chapter discusses how to analyze the design floorplan with the Chip Planner.

Design floorplan analysis helps to close timing and ensure optimal performance in highly complex designs. With analysis capability, the Quartus Prime Chip Planner helps you close timing quickly on your designs. Use the Chip Planner together with LogicLock Plus regions to compile your designs hierarchically, preserving the timing results from individual compilation runs. Use LogicLock Plus regions to improve your productivity.

You can perform design analysis, as well as create and optimize the design floorplan with the Chip Planner. To make I/O assignments, use the Pin Planner.

Related Information

I/O Management on page 2-1
For information about the Pin Planner.

Altera Training
For training courses on the Chip Planner.

Migrating Assignments between Quartus Prime Standard Edition and Quartus Prime Pro Edition

The Quartus Prime Pro Edition software does not support the Quartus Prime Standard Edition LogicLock assignments and vice versa. Therefore, if you are migrating a design from Quartus Prime Pro Edition to Quartus Prime Standard Edition or vice versa, you must convert the LogicLock or LogicLock Plus assignments.

Related Information

Migrating to the Quartus Prime Pro Edition

ALTERA®

Chip Planner Overview

The Chip Planner provides a visual display of chip resources. The Chip Planner shows:

- Logic placement regions
- LogicLock Plus regions
- Relative resource usage
- Detailed routing information

- Fan-in and fan-out connections between nodes
- Timing paths between registers
- Delay estimates for paths
- Routing congestion information

The Chip Planner lets you:

- Make assignment changes with the Chip Planner, such as creating and deleting resource assignments.
- Perform post-compilation changes such as creating, moving, and deleting logic cells and I/O atoms.
- View and create assignments for a design floorplan.
- Perform power and design analyses.
- Change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, and RAM and digital signal processing (DSP) blocks.

Starting the Chip Planner

To start the Chip Planner, choose **Tools** > **Chip Planner**. You can also start the Chip Planner by the following methods:

- Click the Chip Planner icon ◆ on the Quartus Prime software toolbar.
- On the Shortcut menu in the following tools, click **Locate** > **Locate in Chip Planner** to locate:

  - Compilation Report
  - LogicLock Plus Regions window
  - Technology Map Viewer
  - Project Navigator window
  - RTL source code
  - Node Finder
  - Simulation Report
  - RTL Viewer
  - Report Timing panel of the TimeQuest Timing Analyzer

Chip Planner Toolbar

The Chip Planner provides powerful tools for design analysis with a GUI. You can access Chip Planner commands from the View menu and the Shortcut menu, or by clicking the icons on the toolbar.

Chip Planner Presets, Layers, and Editing Modes

The Chip Planner models types of resource objects as unique display layers. It also uses presets—which are predefined sets of layer settings—to control the display of resources.

The Chip Planner provides a set of default presets, and you can create custom presets to customize the display for your particular needs. The **Basic**, **Detailed**, and **Floorplan Editing** presets provided with the Chip Planner are useful for general assignment-related activities.

The Chip Planner editing modes determine the operations that you can perform. The assignment editing mode allows you to make assignment changes that are applied by the Fitter during the next place and route operation.

Locate History Window

As you optimize your design floorplan, you might have to locate a path or node in the Chip Planner many times. The Locate History window lists all the nodes and paths you have displayed using a **Locate in Chip Planner** command, providing easy access to the nodes and paths of interest to you.

If you locate a required path from the **TimeQuest Timing Analyzer Report Timing** pane, the **Locate History** window displays the required clock path. If you locate an arrival path from the **TimeQuest Timing Analyzer Report Timing** pane, the **Locate History** window displays the path from the arrival clock to the arrival data. Double-clicking a node or path in the **Locate History** window displays the selected node or path in the Chip Planner.

Related Information

Layers Settings Dialog Box

LogicLock Plus Regions

LogicLock Plus regions are floorplan location constraints that help you constrain logic in the target device. When you assign entity instances or nodes to a LogicLock Plus region, you direct the Fitter to place those entity instances or nodes within the region during fitting. Your floorplan can contain multiple LogicLock Plus regions.

The Quartus Prime Pro Edition software implements fully hierarchical LogicLock Plus region assignments. A LogicLock Plus region is defined by its height, width, and location. Regions can be rectangular, non-rectangular, or disjoint. You can assign two region types:

- Placement regions—Use these regions to constrain logic to a specific area of the device; the Fitter places the logic in the region you specify. The Fitter may also place other logic in the region unless you have designated the region as reserved.
- Routing regions—If you are using partial reconfiguration, use these regions to constrain routing to a specific area. You use a routing region with a corresponding placement region; the routing region must fully contain the placement region. Additionally, routing regions cannot overlap.

LogicLock Plus regions support the following additional assignments:

- **Reserved**—Prevents the Fitter from placing other logic in the region. You cannot apply the **Reserved** assignment to routing regions.
- **Core-Only**—Excludes pins from a region. Unlike the Quartus Prime Standard Edition software, Quartus Prime Pro Edition region assignments apply to pins by default. If the region is designated as reserved and core only, pins are not reserved from the region.

Note: LogicLock Plus assignments are not compatible with Quartus Prime Standard Edition LogicLock assignments. Additionally, Quartus Prime Pro Edition cannot use LogicLock assignments and Quartus Prime Standard Edition cannot use LogicLock Plus assignments.

Creating LogicLock Plus Regions

You can create LogicLock Plus Regions using several methods, such as with the Chip Planner.

Creating LogicLock Plus Regions with the Project Navigator

After you perform either a full compilation or analysis and elaboration on the design, the Project Navigator displays the hierarchy of the design. If the Project Navigator is not already open, choose **View** > **Utility Windows** > **Project Navigator**. With the design hierarchy fully expanded, right-click on any design entity, and click **Create New LogicLock Plus Region** to create a LogicLock Plus region and assign the entity to the new region. The new region has the same name as the entity.

Creating LogicLock Plus Regions with the LogicLock Plus Regions Window

To create a LogicLock Plus region with the LogicLock Plus Regions window, choose **Assignments** > **LogicLock Plus Regions Window**. In the LogicLock Plus Regions window, click **<<new>>**.

Creating LogicLock Plus Regions with the Chip Planner

To create a LogicLock Plus region in the Chip Planner, choose **View** > **LogicLock Plus Regions** > **Assign LogicLock Plus Region**, then click and drag on the Chip Planner floorplan to create a region of your preferred location and size.

Creating LogicLock Plus Regions with the Assignment Editor

Unlike Quartus Prime Standard Edition, in Quartus Prime Pro Edition you can assign LogicLock Plus regions using the Assignment Editor. To create an assignment, choose **Spectra-Q <placement type>** in the **Assignment Name** field.

Creating Non-Rectangular LogicLock Plus Regions

When you create a floorplan for your design, you may want to create non-rectangular LogicLock Plus regions to exclude certain resources from the LogicLock Plus region.

You might also create a non-rectangular LogicLock Plus region to place certain parts of your design around specific device resources to improve performance.

To create a non-rectangular region with the **Merge LogicLock Plus Region** command, follow these steps:

1. In the Chip Planner, create two or more contiguous or non-contiguous rectangular regions. Do not assign logic to the regions; leave them unassigned.
2. Arrange the regions that you have created into the locations where you want the non-rectangular region.
3. Select all the individual regions that you want to merge by clicking each of them while pressing the Shift key.
4. Right-click the title bar of any of the LogicLock Plus regions that you want to merge, point to **LogicLock Plus Regions**, and then click **Merge LogicLock Plus Region**. The individual regions that you select merge to create a single new region.

   You cannot merge assigned regions. First create and merge the regions, then assign logic to the merged region.

Related Information
Creating LogicLock Plus Regions on page 12-3

Hierarchical Regions

LogicLock Plus regions are fully hierarchical. Parent regions should completely contain child regions and must overlap with child regions. If the parent and child regions do not overlap, the software issues an error. The **Reserved** and **Core-Only** assignments also apply hierarchically.

LogicLock Plus assignments follow the same precedence as other constraints and assignments.

Placing Device Resources into LogicLock Plus Regions

A LogicLock Plus region includes all device resources within its boundaries, including memory and pins. The Quartus Prime Pro Edition software automatically includes pins when you assign in instance to a region. You can manually exclude pins with a core only assignment.

When you add an entity to a region, region has the same name as the entity. You cannot change the region name.

Only one LogicLock Plus region can use a device resource. If a LogicLock Plus region boundary includes part of a device resource, the Quartus Prime software allocates the entire resource to that LogicLock Plus region.

Note: If you want to import multiple instances of a module into a top-level design, you must ensure that the device has two or more locations with exactly the same device resources. (You can determine this from the applicable device specifications.) If the device does not have another area with exactly the same resources, the Quartus Prime software generates a fitting error during compilation of the top-level design.

LogicLock Plus Regions Window

Use the LogicLock Plus Regions window to create LogicLock Plus regions, assign nodes and entities to them, and modify the properties of a LogicLock Plus region. You can modify the size, state, width, height, origin, and whether the region is a reserved or core only region.

Reserved LogicLock Plus Region

The Quartus Prime software honors all entity and node assignments to LogicLock Plus regions. Occasionally, entities and nodes do not occupy an entire region, which leaves some of the region's resources unoccupied.

To increase the region's resource utilization and performance, the Quartus Prime software's default behavior fills the unoccupied resources with other nodes and entities that have not been assigned to another region. You can prevent this behavior by turning on **Reserved** in the LogicLock Plus Regions window. When you turn on this option, your LogicLock Plus region contains only the entities and nodes that you specifically assigned to your LogicLock Plus region.

Additional Quartus Prime LogicLock Plus Design Features

To complement the LogicLock Plus Regions window, the Quartus Prime software has additional features to help you design with LogicLock Plus regions.

Analysis and Synthesis Resource Utilization by Entity

The Compilation Report contains an **Analysis and Synthesis Resource Utilization by Entity** section, which reports resource usage statistics, including entity-level information. You can use this feature to verify that any LogicLock Plus region you manually create contains enough resources to accommodate all the entities you assign to it.

Quartus Prime Revisions Feature

When you evaluate different LogicLock Plus regions in your design, you might want to experiment with different configurations to achieve your desired results. The Quartus Prime Revisions feature allows you to organize the same project with different settings until you find an optimum configuration.

To use the Revisions feature, choose **Project** > **Revisions**. You can create a revision from the current design or any previously created revisions. Each revision can have an associated description. You can use revisions to organize the placement constraints created for your LogicLock Plus regions.

Virtual Pins

A virtual pin is an I/O element that is temporarily mapped to a logic element and not to a pin during compilation. The software implements it as a LUT.

When you apply the **Virtual Pin** assignment to an input pin, the pin no longer appears as an FPGA pin, but is fixed to GND or VCC in the design. The assigned pin is not an open node.

Virtual pins should be used only for I/O elements in lower-level design entities that become nodes when imported to the top-level design. You can create virtual pins by assigning the **Virtual Pin** logic option to an I/O element.

You might use virtual pin assignments when you compile a partial design, because not all the I/Os from a partial design drive chip pins at the top level.

The virtual pin assignment identifies the I/O ports of a design module that are internal nodes in the top-level design. These assignments prevent the number of I/O ports in the lower-level modules from exceeding the total number of available device pins. Every I/O port that you designate as a virtual pin becomes mapped to either a logic cell or an adaptive logic module (ALM), depending on the target device.

Note: The **Virtual Pin** logic option must be assigned to an input or output pin. If you assign this option to a bidirectional pin, tri-state pin, or registered I/O element, Analysis & Synthesis ignores the assignment. If you assign this option to a tri-state pin, the Fitter inserts an I/O buffer to account for the tri-state logic; therefore, the pin cannot be a virtual pin. You can use multiplexer logic instead of a tri-state pin if you want to continue to use the assigned pin as a virtual pin. Do not use tri-state logic except for signals that connect directly to device I/O pins.

In the top-level design, you connect these virtual pins to an internal node of another module. By making assignments to virtual pins, you can place those pins in the same location or region on the device as that of the corresponding internal nodes in the top-level module. You can use the **Virtual Pin** option when compiling a LogicLock Plus module with more pins than the target device allows. The **Virtual Pin** option can enable timing analysis of a design module that more closely matches the performance of the module after you integrate it into the top-level design.

Note: In the Node Finder, you can set **Filter Type** to **Pins: Virtual** to display all assigned virtual pins in the design. Alternatively, to access the Node Finder from the Assignment Editor, double-click the **To** field; when the arrow appears on the right side of the field, click the arrow and select **Node Finder**.

Design Floorplan Analysis in the Chip Planner

The Chip Planner helps you visually analyze the floorplan of your design at any stage of your design cycle. With the Chip Planner, you can view post-compilation placement, connections, and routing paths.

You can also create LogicLock Plus regions and location assignments. The Chip Planner allows you to create new logic cells and I/O atoms and to move existing logic cells and I/O atoms in your design. You

can also see global and regional clock regions within the device, and the connections between I/O atoms, PLLs and the different clock regions.

From the Chip Planner, you can launch the Resource Property Editor that changes the properties and parameters of device resources and modifies connectivity between certain types of device resources. The Change Manager records any changes that you make to your design floorplan so that you can selectively undo changes.

The following sections present Chip Planner floorplan views and design analysis procedures which you can use with any Chip Planner preset, unless a procedure requires a specific preset or editing mode.

Related Information

- **About the Change Manager**
- **About the Resource Property Editor**
  For more information about the Resource Property Editor and the Change Manager, refer to Quartus Prime Help.

Chip Planner Floorplan Views

The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the targeted Altera device. As you zoom in, the level of abstraction decreases, revealing more details about your design.

Bird's Eye View

The Bird's Eye View displays a high-level picture of resource usage for the entire chip and provides a fast and efficient way to navigate between areas of interest in the Chip Planner.

The Bird's Eye View is particularly useful when the parts of your design that you want to view are at opposite ends of the chip and you want to quickly navigate between resource elements without losing your frame of reference.

Properties Window

The Properties Window displays detailed properties of the objects (such as atoms, paths, LogicLock Plus regions, or routing elements) currently selected in the Chip Planner. To display the Properties Window, click **Properties** on the **View** menu in the Chip Planner.

Related Information

- **Displaying Resources and Information**
- **Bird's Eye View**
  For more information about displaying information in the Bird's Eye View.

Viewing Architecture-Specific Design Information

By adjusting the **Layers Settings** in the Chip Planner, you can view the following architecture-specific information related to your design:

- **Device routing resources used by your design**—View how blocks are connected, as well as the signal routing that connects the blocks.
- **LE configuration**—View logic element (LE) configuration in your design. For example, you can view which LE inputs are used; if the LE utilizes the register, the look-up table (LUT), or both; as well as the signal flow through the LE.
- **ALM configuration**—View ALM configuration in your design. For example, you can view which ALM inputs are used, if the ALM utilizes the registers, the upper LUT, the lower LUT, or all of them. You can also view the signal flow through the ALM.
- **I/O configuration**—View device I/O resource usage. For example, you can view which components of the I/O resources are used, if the delay chain settings are enabled, which I/O standards are set, and the signal flow through the I/O.
- **PLL configuration**—View phase-locked loop (PLL) configuration in your design. For example, you can view which control signals of the PLL are used with the settings for your PLL.
- **Timing**—View the delay between the inputs and outputs of FPGA elements. For example, you can analyze the timing of the DATAB input to the COMBOUT output.

In addition, you can modify the following device properties with the Chip Planner:

- LEs and ALMs
- I/O cells
- PLLs
- Registers in RAM and DSP blocks
- Connections between elements
- Placement of elements

For more information about LEs, ALMs, and other resources of an FPGA device, refer to the relevant device handbook.

Viewing Available Clock Networks in the Device

When you select a task with clock region layer preset enabled, you can display the areas of the chip that are driven by global and regional clock networks.

Depending on the clock layers activated in the selected preset, the Chip Planner displays regional and global clock regions in the device, and the connectivity between clock regions, pins, and PLLs. Clock regions appear as rectangular overlay boxes with labels indicating the clock type and index. You can select each clock network region by clicking on the clock region. The clock-shaped icon at the top-left corner indicates that the region represents a clock network region. You can change the color in which the Chip Planner displays clock regions on the **Options** dialog box of the Tools menu.

The **Layers Settings** dialog box lists layers for different clock region types; when the selected device does not contain a given clock region, the option for that category is unavailable in the dialog box. You can customize the Chip Planner's display of clock regions by creating a custom preset with selected clock layers enabled in the **Layers Settings** dialog box.

Viewing Critical Paths

Critical paths are timing paths in your design that have a negative slack. These timing paths can span from device I/Os to internal registers, registers to registers, or from registers to device I/Os.

The slack of a path determines its criticality; slack appears in the timing analysis report. Design analysis for timing closure is a fundamental requirement for optimal performance in highly complex designs. The analytical capability of the Chip Planner helps you close timing on complex designs.

Viewing critical paths in the Chip Planner shows why a specific path is failing. You can see if any modification in the placement can reduce the negative slack. You can display details of a path (to expand/

collapse the path to/from the connections in the path) by clicking **Expand Connections** in the toolbar, or by clicking on the "+/-" on the label.

You can locate failing paths from the timing report in the TimeQuest Timing Analyzer. To locate the critical paths, click the **Report Timing** task from the **Custom Reports** group in the **Tasks** pane of the TimeQuest Timing Analyzer. From the View pane, which lists the failing paths, right-click on any failing path or node, and select **Locate Path**. From the **Locate** dialog box, select **Chip Planner** to see the failing path in the Chip Planner.

Note: To display paths in the floorplan, you must first make timing settings and perform a timing analysis.

Related Information

The Quartus Prime TimeQuest Timing Analyzer

For more information about performing static timing analysis with the TimeQuest Timing Analyzer.

Viewing Routing Congestion

The **Report Routing Utilization** task allows you to determine the percentage of routing resources in use following a compilation. This feature can identify where there is a lack of routing resources, helping you to make design changes to meet routing congestion design requirements.

1. Choose **Tools** > **Chip Planner**.
2. To view the routing congestion in the Chip Planner, double-click the **Report Routing Utilization** command in the **Tasks** list.
3. Click **Preview** in the **Report Routing Utilization** dialog box to preview the default congestion display.
4. Change the **Routing Utilization Type** to display congestion for specific resources.

Note: The default display uses dark blue for 0% congestion (blue indicates zero utilization) and red for 100%. You can adjust the slider for **Threshold percentage** to change the congestion threshold level.

The routing congestion map uses the color and shading of logic resources to indicate relative resource utilization; darker shading represents a greater utilization of routing resources. Areas where routing utilization exceeds the threshold value specified in the **Report Routing Utilization** dialog box appear in red. The congestion map can help you determine whether you can modify the floorplan, or modify the RTL to reduce routing congestion.

To identify a lack of routing resources, it is necessary to investigate each routing interconnect type separately by selecting each interconnect type in turn in the **Routing Utilization Settings** dialog box.

The Compiler messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, could be an indication that it might be difficult to fit your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, are likely to have increased chances of not getting a valid fit.

Viewing I/O Banks

The Chip Planner shows all of the I/O banks of the device. To see the I/O bank map of the device, select **Report All I/O Banks** in the Tasks pane.

Viewing High-Speed Serial Interfaces (HSSI)

The Chip Planner displays a detailed block view of the receiver and transmitter channels of the high-speed serial interfaces. To display the HSSI block view, select **Report HSSI Block Connectivity**.

Generating Fan-In and Fan-Out Connections

The ability to display fan-in and fan-out connections enables you to view the atoms that fan-in to or fan-out from the selected atom. To remove the connections displayed, use the **Clear Unselected Connections** icon ⬉ in the Chip Planner toolbar.

Generating Immediate Fan-In and Fan-Out Connections

The ability to display immediate fan-in and fan-out connections enables you to view the resource that is the immediate fan-in or fan-out connection for the selected atom. For example, if you select a logic resource and choose to view the immediate fan-in for that resource, you can see the routing resource that drives the logic resource. You can generate immediate fan-ins and fan-outs for all logic resources and routing resources. To remove the displayed connections from the screen, click the **Clear Unselected Connections** icon in the toolbar.

Highlight Routing

The **Show Physical Routing** command in the **Locate History** pane enables you to highlight the routing resources used by a selected path or connection.

Figure 12-1: Highlight Routing
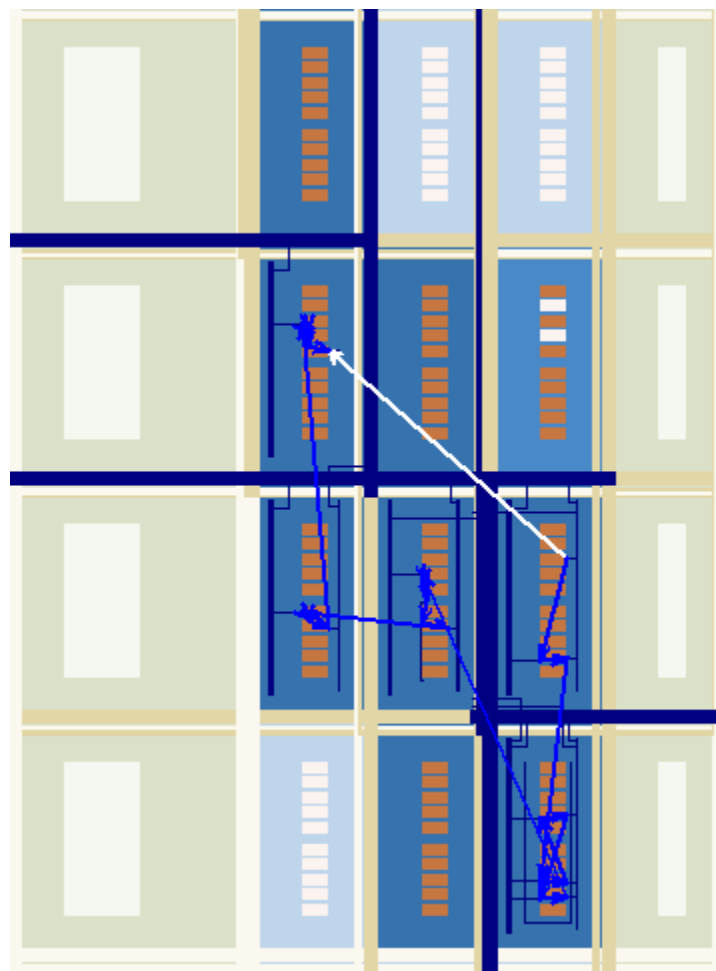
Show Delays

With the **Show Delays** command, you can view timing delays for paths located from TimeQuest Timing Analyzer reports. For example, you can view the delay between two logic resources or between a logic resource and a routing resource.

Figure 12-2: Show Delays Associated in a TimeQuest Timing Analyzer Path
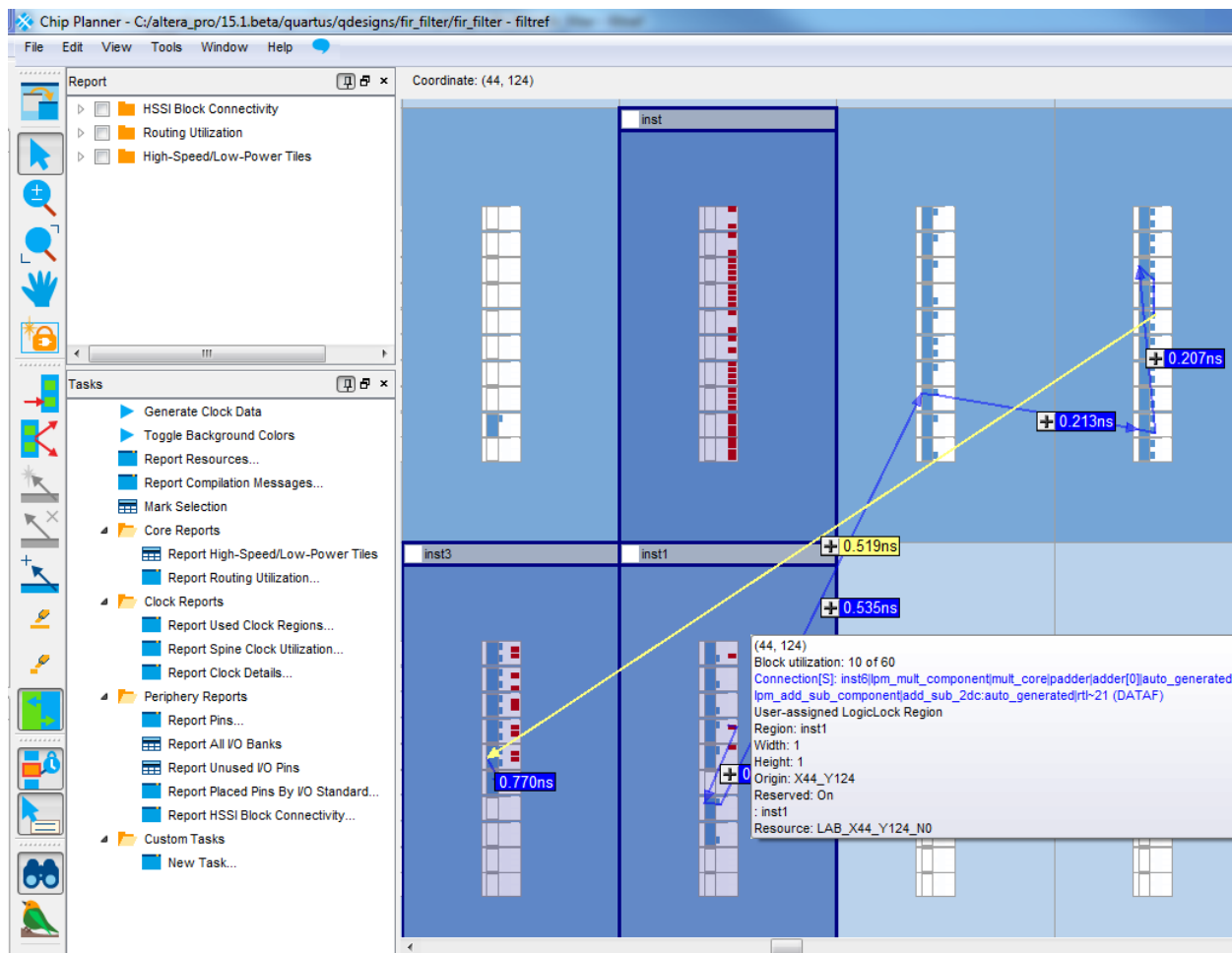
Exploring Paths in the Chip Planner

You can use the Chip Planner to explore paths between logic elements. The following example uses the Chip Planner to traverse paths from the Timing Analysis report.

Locate Path from the Timing Analysis Report to the Chip Planner

To locate a path from the Timing Analysis report to the Chip Planner, perform the following steps:

1. Select the path you want to locate in the Timing Analysis report.
2. Right-click the path and point to **Locate Path** > **Locate in Chip Planner**. The path is displayed with its timing data in the Chip Planner main window and is listed in the Locate History window.
3. To view the routing resources taken for a path you have located in the Chip Planner, use of one the following methods:

- Select the path and then click the **Highlight Routing** icon in the Chip Planner toolbar, or from the View menu, click **Highlight Routing**.
- Right-click the path and choose Expand Connections.

Analyzing Connections for a Path

To determine the connections between items in the Chip Planner, click the **Expand Connections** icon on the toolbar. To add the timing delays for paths you locate from the TimeQuest Timing Analyzer, click the **Show Delays** icon on the toolbar. To see the constituent delays on the selected path, click the "+" sign next to the path delay displayed in the Chip Planner.

Figure 12-3: Path Analysis in the Chip Planner of a Path Located from TimeQuest
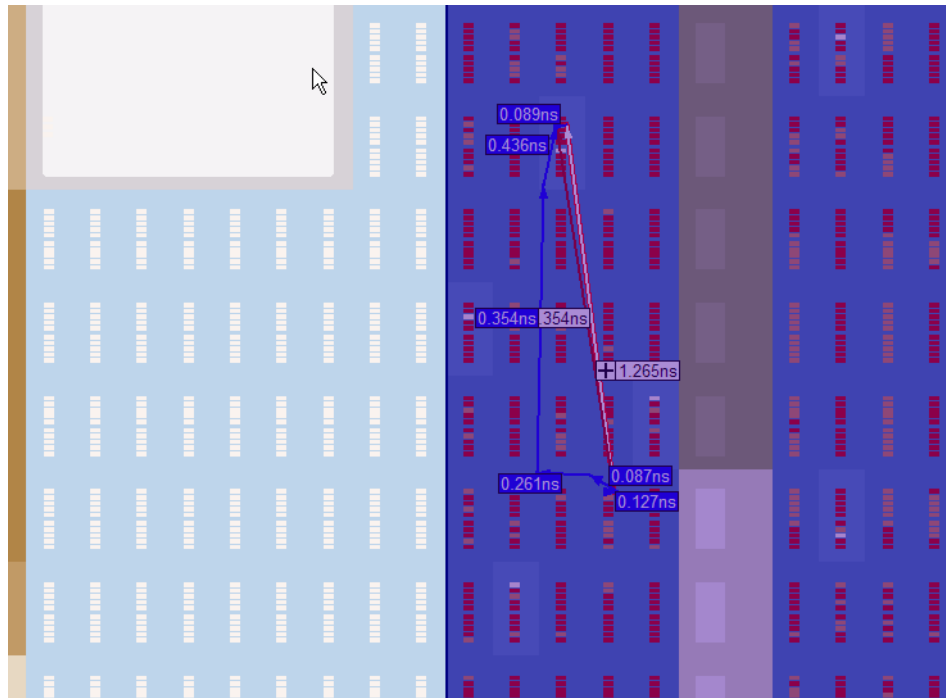
Viewing Assignments in the Chip Planner

You can view location assignments by selecting the appropriate layer set in the Chip Planner. To view location assignments, select the **Floorplan Editing** preset or any custom preset that displays block utilization, and the Assignment editing mode. The Chip Planner shows location assignments graphically, by displaying assigned resources in a particular color (gray, by default). You can create or move an assignment by dragging the selected resource to a new location.

Figure 12-4: Viewing Assignments in the Chip Planner
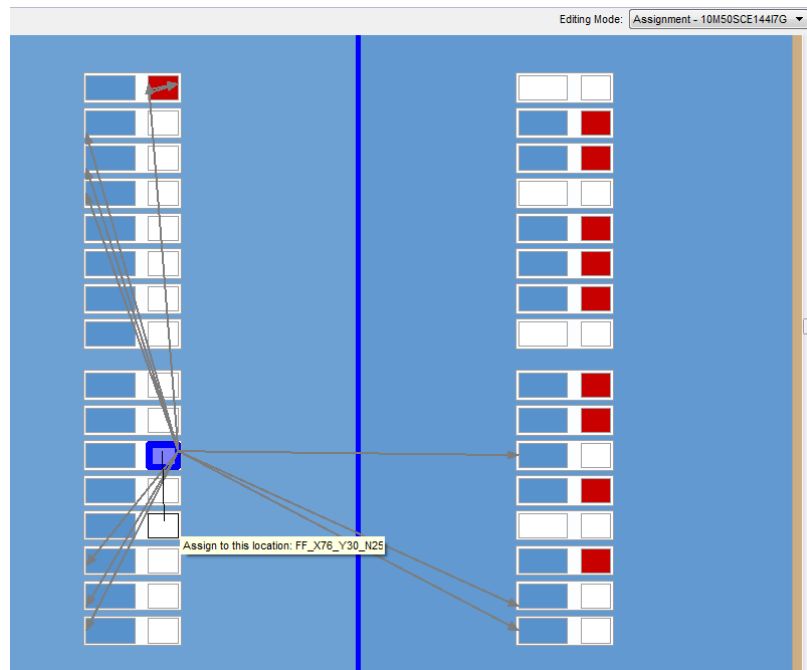
You can make node and pin location assignments to LogicLock Plus regions and custom regions using the drag-and-drop method in the Chip Planner. The Fitter applies the assignments that you create during the next place-and-route operation.

Viewing High-Speed and Low-Power Tiles in the Chip Planner

Some Altera devices have ALMs that can operate in either high-speed mode or low-power mode. The power mode is set during the fitting process in the Quartus Prime software. These ALMs are grouped together to form larger blocks, called "tiles."

To view a power map, select **Tasks** > **Report High-Speed/Low-Power Tiles** after running the Fitter. The Chip Planner displays low-power and high-speed tiles in contrasting colors; yellow tiles operate in a high-speed mode, while blue tiles operate in a low-power mode. When you select the **Power** task, you can perform all floorplan-related functions for this task; however, you cannot edit tiles to change the power mode.

Scripting Support

You can run procedures and specify the settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt.

For detailed information about scripting command options, refer to the Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

Related Information

[API Functions for Tcl](#) on page 5-1
For more information about Tcl scripting in Quartus Prime Help.

[Tcl Scripting](#) on page 5-1
[API Functions for Tcl](#) For more information about Tcl scripting .

[Command-Line Scripting](#) on page 4-1
For more information about command-line scripting.

[Quartus Prime Settings File Manual](#)
For information about all settings and constraints in the Quartus Prime software.

Creating LogicLock Plus Assignments

The Quartus Prime software supports Tcl commands to create or modify LogicLock Plus assignments.

Create or Modify a Placement Region

The following assignment creates a new placement region with bounding box coordinates 10, 10, 20, 20.
Use the same command format to modify an existing assignment.

```
set_instance_assignment -name PLACE_REGION -to <node name(s)> "10 10 20 20"
```

Note: To specify a non-rectangular or disjoint region, use a semicolon (;) as the delimiter between two or more bounding boxes.

Assign multiple instances to the same region with multiple PLACE_REGION instance assignments.

Create or Modify a Routing Region

The following assignment creates a routing region with bounding box coordinates 5, 5, 30, 30. Use the same command format to modify an existing assignment.

```
set_instance_assignment -name ROUTE_REGION -to <node name(s)> "5 5 30 30"
```

Note: All instances with a routing region assignment must have a respective placement region; the routing region must fully contain the placement region.

Routing regions cannot overlap.

Specify a Region as Reserved

The following assignment reserves an existing region:

```
set_instance_assignment -name <instance name> RESERVE_PLACE_REGION -to <node name(s)> ON
```

Note: You can only reserve placement regions.

Specify a Region as Core Only

By deafult, the Quartus Prime Pro Edition software includes pins in LogicLock Plus assignments. To specify a region as core only (i.e., periphery logic in the instance is not constrained), use the following assignment:

```
set_instance_assignment -name <instance name> CORE_ONLY_PLACE_REGION -to <node
name(s)> ON
```

Assigning Virtual Pins

Use the following Tcl command to turn on the virtual pin setting for a pin called `my_pin`:

```
set_instance_assignment -name VIRTUAL_PIN ON -to my_pin
```

Related Information

Virtual Pins on page 12-6

Tcl Scripting on page 5-1
For more information about Tcl scripting.

Document Revision History

Table 12-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*. • Added information on how to use LogicLock Plus regions.
2015.05.04	15.0.0	Added information about color coding of LogicLock Plus regions.
2014.12.15	14.1.0	Updated description of Virtual Pins assignement to clarify that assigned input is not available.
June 2014	14.0.0	Updated format
November 2013	13.1.0	Removed HardCopy device information.
May 2013	13.0.0	Updated "Viewing Routing Congestion" section Updated references to Quartus UI controls for the Chip Planner
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.

Date	Version	Changes
May 2011	11.0.0	• Updated for the 11.0 release. Edited "LogicLock Plus Regions" Updated "Viewing Routing Congestion" Updated "Locate History" Updated Figures 15-4, 15-9, 15-10, and 15-13 Added Figure 15-6
December 2010	10.1.0	• Updated for the 10.1 release.
July 2010	10.0.0	• Updated device support information • Removed references to Timing Closure Floorplan; removed "Design Analysis Using the Timing Closure Floorplan" section • Added links to online Help topics • Added "Using LogicLock Plus Regions with the Design Partition Planner" section • Updated "Viewing Critical Paths" section • Updated several graphics • Updated format of Document revision History table
November 2009	9.1.0	• Updated supported device information throughout • Removed deprecated sections related to the Timing Closure Floorplan for older device families. (For information on using the Timing Closure Floorplan with older device families, refer to previous versions of the Quartus Prime Handbook, available in the Quartus Prime Handbook Archive.) • Updated "Creating Nonrectangular LogicLock Plus Regions" section • Added "Selected Elements Window" section • Updated table 12-1

Date	Version	Changes
May 2008	8.0.0	• Updated the following sections: "Chip Planner Tasks and Layers" "LogicLock Plus Regions" "Back-Annotating LogicLock Plus Regions" "LogicLock Plus Regions in the Timing Closure Floorplan" • Added the following sections: "Reserve LogicLock Plus Region" "Creating Nonrectangular LogicLock Plus Regions" "Viewing Available Clock Networks in the Device" • Updated Table 10–1 • Removed the following sections: Reserve LogicLock Plus Region Design Analysis Using the Timing Closure Floorplan

Related Information

Quartus Handbook Archive

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

QPP5V2 ✉ **Subscribe** 💬 **Send Feedback**

Netlist Optimizations and Physical Synthesis

The Quartus Prime software offers additional physical synthesis optimizations that improve performance. The Compiler performs physical synthesis optimization during fitting.

Netlist optimizations operate with the atom netlist of your design, which describes a design in terms of specific primitives. An atom netlist file can be an Electronic Design Interchange Format (**.edf**) file or a Verilog Quartus Mapping (**.vqm**) file generated by a third-party synthesis tool. Quartus Prime synthesis generates and internally uses the atom netlist internally.

Physical synthesis optimizations are applied at different stages of the Quartus Prime compilation flow, either during synthesis, fitting, or both.

Note: Because the node names for primitives in the design can change when you use physical synthesis optimizations, you should evaluate whether your design flow requires fixed node names. If you use a verification flow that might require fixed node names, such as the SignalTap® II Logic Analyzer, formal verification, or the LogicLock Plus based optimization flow (for legacy devices), you must turn off physical synthesis options.

This chapter explains how physical synthesis optimizations in the Quartus Prime software can modify your design's netlist to improve the quality of results. This chapter also provides guidelines for applying the various physical synthesis options, and information about preserving compilation results through back-annotation.

WYSIWYG Primitive Resynthesis

If you use a third-party tool to synthesize your design, use the **Perform WYSIWYG primitive resynthesis** option to apply optimizations to the synthesized netlist.

The **Perform WYSIWYG primitive resynthesis** option directs the Quartus Prime software to un-map the logic elements (LEs) in an atom netlist to logic gates, and then re-map the gates back to Altera-specific primitives. Third-party synthesis tools generate either an **.edf** or **.vqm** atom netlist file using Altera-specific primitives. When you turn on the **Perform WYSIWYG primitive resynthesis** option, the Quartus Prime software uses device-specific techniques during the re-mapping process. This feature re-maps the design using the **Optimization Technique** specified for your project (**Speed**, **Area**, or **Balanced**).

The **Perform WYSIWYG primitive resynthesis** option unmaps and remaps only logic cells, also referred to as LCELL or LE primitives, and regular I/O primitives (which may contain registers). Double data rate

ISO 9001:2008 Registered

®

(DDR) I/O primitives, memory primitives, digital signal processing (DSP) primitives, and logic cells in carry/cascade chains are not remapped. This process does not process logic specified in an encrypted **.vqm** file or an **.edf** file, such as third-party intellectual property (IP).

The **Perform WYSIWYG primitive resynthesis** option can change node names in the **.vqm** file or **.edf** file from your third-party synthesis tool, because the primitives in the atom netlist are broken apart and then re-mapped by the Quartus Prime software. The re-mapping process removes duplicate registers. Registers that are not removed retain the same name after re-mapping.

Any nodes or entities that have the **Netlist Optimizations** logic option set to **Never Allow** are not affected during WYSIWYG primitive resynthesis. You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. This option disables WYSIWYG resynthesis for parts of your design.

Note: Primitive node names are specified during synthesis. When netlist optimizations are applied, node names might change because primitives are created and removed. HDL attributes applied to preserve logic in third-party synthesis tools cannot be maintained because those attributes are not written into the atom netlist, which the Quartus Prime software reads.

If you use the Quartus Prime software to synthesize your design, you can use the **Preserve Register (preserve)** and **Keep Combinational Logic (keep)** attributes to maintain certain nodes in the design.

Figure 13-1: Quartus Prime Flow for WYSIWYG Primitive Resynthesis
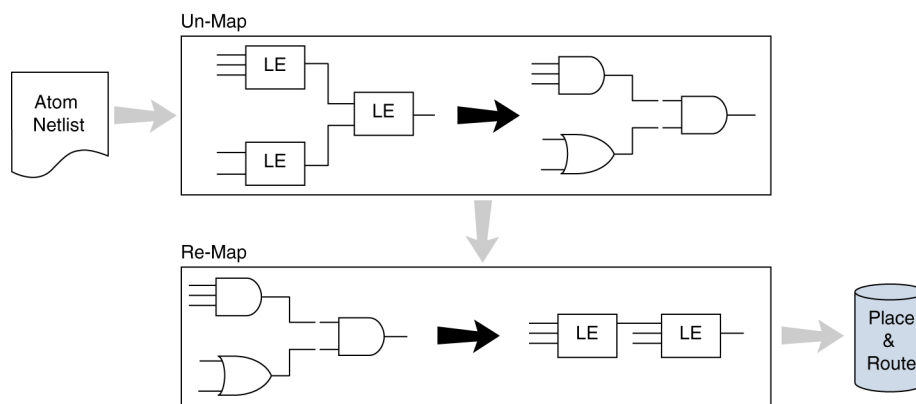
Perform WYSIWYG Primitive Resynthesis

Note: The **Perform WYSIWYG primitive resynthesis** option has no effect if you are using Quartus Prime synthesis to synthesize your design.

To turn on the **Perform WYSIWYG primitive resynthesis** option, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Synthesis)**.

Performing Physical Synthesis Optimizations

The Quartus Prime Fitter places and routes the logic cells to ensure critical portions of logic are close together and use the fastest possible routing resources.

However, routing delays are often a significant part of the typical critical path delay. Physical synthesis optimizations take into consideration placement information, routing delays, and timing information. The Fitter then focuses timing-driven optimizations at those critical parts of the design. The tight integration of the synthesis and fitting processes is known as physical synthesis.

The following sections describe the physical synthesis optimizations available in the Quartus Prime software, and how they can help improve performance and fitting for the selected device.

Nodes or entities that have the **Netlist Optimizations** logic option set to **Never Allow** are not affected by physical synthesis algorithms. You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. Use this option to disable physical synthesis optimizations for parts of your design.

Some physical synthesis options affect only registered logic, while others affect only combinational logic. Select options based on whether you want to keep the registers intact. For example, if your verification flow involves formal verification, you might want to keep the registers intact.

To choose physical synthesis optimization options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

Related Information
[Compiler Settings Page (Settings Dialog Box)](#)

Spectra-Q Physical Synthesis Optimization

The Quartus Prime software includes advanced physical synthesis optimization. The **Spectra-Q Physical Synthesis** option uses the Spectra-Q engine to perform combinational and sequential optimization during fitting to improve circuit performance for Arria 10 designs.

Spectra-Q Physical Synthesis fine tunes the physical placement and structure of logic to achieve timing closure and meet performance requirements. Enable the Spectra-Q physical synthesis option by clicking **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

Setting Physical Synthesis Options

You enable **Spectra-Q Physical Synthesis** options in the Settings dialog box. When you select **Performance (High effort)** or **Performance (Aggressive)**, the **Spectra-Q Physical Synthesis** option is set

to **On** automatically for supported devices. For all other optimization modes, the **Spectra-Q Physical Synthesis** option is set to **Off** by default.
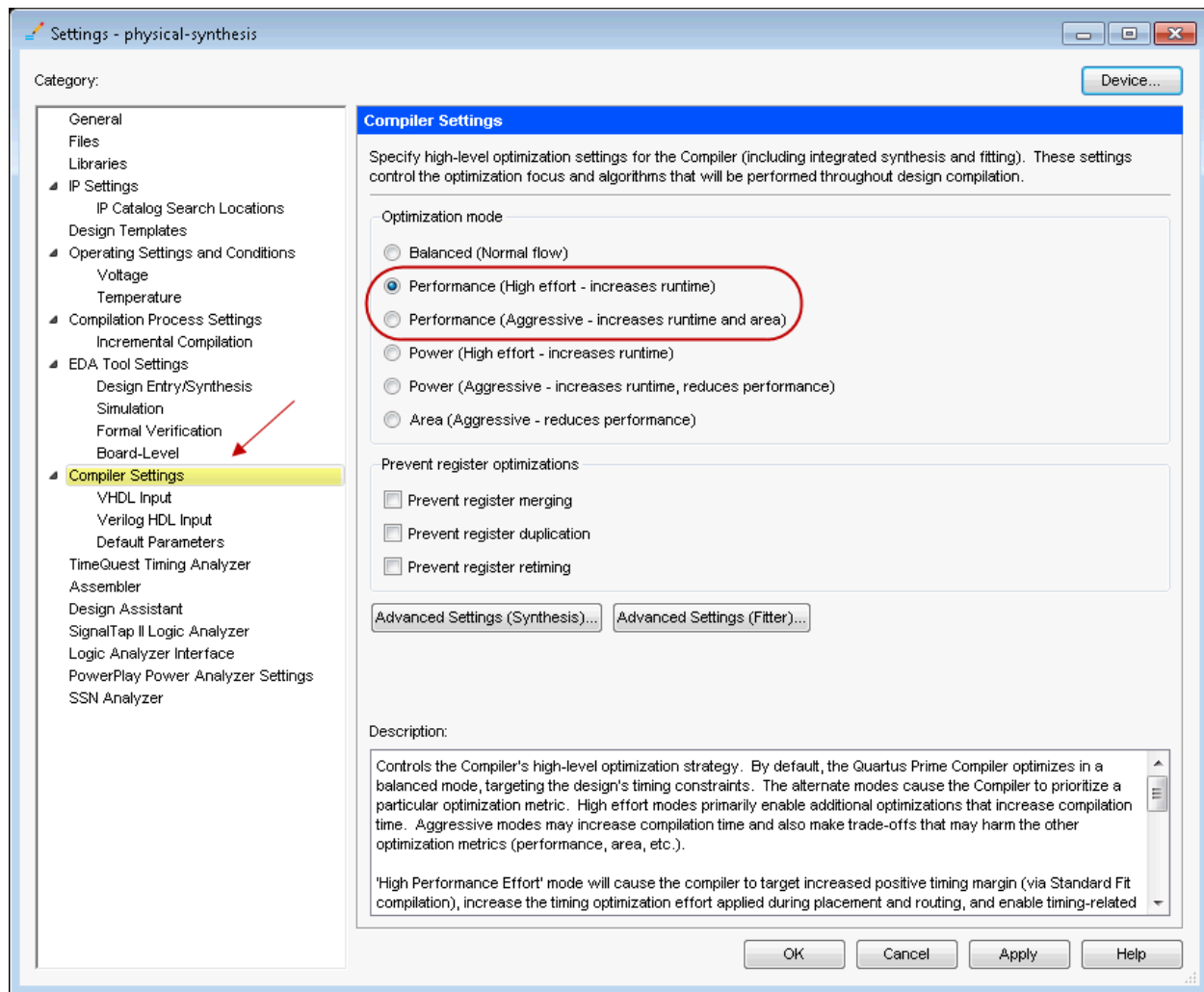
To specify physical synthesis options, follow these steps:

1. Click **Assignments** > **Settings** > **Compiler Settings**.
2. In the **Compiler Settings**, select either high effort or aggressive performance for the **Optimization mode** to automatically enable **Spectra-Q Physical Synthesis**.
3. If you select an optimization mode other than high effort or aggressive, and you want to turn on Spectra-Q Physical Synthesis, click **Advanced Settings (Fitter)**.
4. In the **Advanced Fitter Settings** dialog box, turn on **Spectra-Q Physical Synthesis**.

Physical Synthesis Options

The Quartus Prime software provides physical synthesis optimization options to improve fitting results. To access these options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

Table 13-1: Physical Synthesis Options

Option	Description
Netlist Optimizations	You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. Use this option to disable physical synthesis optimizations for parts of your design.
Perform asynchronous signal pipelining	The Fitter performs automatic insertion of pipeline stages for asynchronous clear and asynchronous load signals during fitting when these signals negatively affect performance. You can use this option if asynchronous control signal recovery and removal times are not achieving requirements.
Perform physical synthesis for combinational logic	Swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. Also allows the duplication of LUTs to enable further optimizations on the critical path.
Perform Register Retiming for Performance	Enables the movement of registers across combinational logic, allowing the Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths.
Physical Synthesis for Combinational Logic for Fitting	Causes registers that do not have a **Power-Up Level logic** option setting to power up with a don't care logic level (x). When the **Power-Up Don't Care** option is turned on, the Compiler determines when it is beneficial to change the power-up level of a register to minimize the area of the design. A power-up state of zero is maintained unless there is an immediate area advantage.
Perform WYSIWYG Primitive Resynthesis	Specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the **Optimization Technique** logic option.
Spectra-Q Physical Synthesis	Uses the Spectra-Q physical synthesis engine to perform combinational and sequential optimization during fitting to improve circuit performance.

Applying Netlist Optimization Options

The improvement in performance when using netlist optimizations is design dependent. If you have restructured your design to balance critical path delays, netlist optimizations might yield minimal improvement in performance.

You may have to experiment with available options to see which combination of settings works best for a particular design. Refer to the messages in the compilation report to see the magnitude of improvement with each option, and to help you decide whether you should turn on a given option or specific effort level.

Turning on more netlist optimization options can result in more changes to the node names in the design; bear this in mind if you are using a verification flow, such as the SignalTap II Logic Analyzer or formal verification that requires fixed or known node names.

To find the best results, you can use the Quartus Prime Design Space Explorer II (DSE) to apply various sets of netlist optimization options.

Related Information
About Design Space Explorer II

Viewing Synthesis and Netlist Optimization Reports

Physical synthesis optimizations performed during synthesis write results to the synthesis report. To access this report, perform the following steps:

1. On the Processing menu, click **Compilation Report**.
2. In the **Compilation Report** list, open the **Synthesis**. folder to view synthesis results.
3. In the **Compilation Report** list, open the **Fitter** folder to view the **Netlist Optimizations** table.

Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Quartus Prime Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section on either an instance or global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <QSF variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <QSF variable name> <value> \
-to <instance name>
```

Related Information

- **Tcl Scripting** on page 5-1
- **API Functions for Tcl**
- **Command-Line Scripting** on page 4-1
- **Quartus Prime Settings File Manual**

Synthesis Netlist Optimizations

The project **.qsf** file preserves the settings that you specify in the GUI. Alternatively, you can edit the **.qsf** directly. The **.qsf** file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 13-2: Synthesis Netlist Optimizations and Associated Settings

Setting Name	Quartus Prime Settings File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	`ADV_NETLIST_OPT_SYNTH_WYSIWYG_ REMAP`	`ON`, `OFF`	Global, Instance
Optimization Mode	`OPTIMIZATION_MODE`	`BALANCEDHIGH PERFORMANCE EFFOR AGGRESSIVE PERFORMANCE`	Global, Instance
Power-Up Don't Care	`ALLOW_POWER_UP_DONT_CARE`	`ON`, `OFF`	Global

Physical Synthesis Optimizations

The project **.qsf** file preserves the settings that you specify in the GUI. Alternatively, you can edit the **.qsf** directly. The **.qsf** file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 13-3: Physical Synthesis Optimizations and Associated Settings

Setting Name	Quartus Prime Settings File Variable Name	Values	Type
Spectra-Q Physical Synthesis	`SPECTRAQ_PHYSICAL_SYNTHESIS`	`ON`, `OFF`	Global
Perform Register Retiming	`PHYSICAL_SYNTHESIS_REGISTER_ RETIMING`	`ON`, `OFF`	Global

Document Revision History

Table 13-4: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*. • Added Spectra-Q Physical Synthesis.
2014.12.15	14.1.0	• Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations Settings to Compiler Settings. • Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Removed HardCopy device information.

Date	Version	Changes
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	• Added links to Quartus Prime Help in several sections. • Removed Referenced Documents section. • Reformatted Document Revision History
November 2009	9.1.0	• Added information to "Physical Synthesis for Registers—Register Retiming" • Added information to "Applying Netlist Optimization Options" • Made minor editorial updates
March 2009	9.0.0	• Was chapter 11 in the 8.1.0 release. • Updated the "Physical Synthesis for Registers—Register Retiming" and "Physical Synthesis Options for Fitting" • Updated "Performing Physical Synthesis Optimizations" • Deleted Gate-Level Register Retiming section. • Updated the referenced documents
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	• Updated "Physical Synthesis Optimizations for Performance on page 11-9 • Added Physical Synthesis Options for Fitting on page 11-16

Related Information

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

2015.11.02

QPP5V2 ✉ **Subscribe** 💬 **Send Feedback**

Signal Integrity Analysis with Third-Party Tools

With the ever-increasing operating speed of interfaces in traditional FPGA design, the timing and signal integrity margins between the FPGA and other devices on the board must be within specification and tolerance before a single PCB is built.

If the board trace is designed poorly or the route is too heavily loaded, noise in the signal can cause data corruption, while overshoot and undershoot can potentially damage input buffers over time.

As FPGA devices are used in high-speed applications, signal integrity and timing margin between the FPGA and other devices on the printed circuit board (PCB) are important aspects to consider to ensure proper system operation. To avoid time-consuming redesigns and expensive board respins, the topology and routing of critical signals must be simulated. The high-speed interfaces available on current FPGA devices must be modeled accurately and integrated into timing models and board-level signal integrity simulations. The tools used in the design of an FPGA and its integration into a PCB must be "board-aware"—able to take into account properties of the board routing and the connected devices on the board.

The Quartus Prime software provides methodologies, resources, and tools to ensure good signal integrity and timing margin between Altera FPGA devices and other components on the board. Three types of analysis are possible with the Quartus Prime software:

- I/O timing with a default or user-specified capacitive load and no signal integrity analysis (default)
- The Quartus Prime **Enable Advanced I/O Timing** option utilizing a user-defined board trace model to produce enhanced timing reports from accurate "board-aware" simulation models
- Full board routing simulation in third-party tools using Altera-provided or generated Input/Output Buffer Information Specification (IBIS) or HSPICE I/O models

I/O timing using a specified capacitive test load requires no special configuration other than setting the size of the load. I/O timing reports from the Quartus Prime TimeQuest or the Quartus Prime Classic Timing Analyzer are generated based only on point-to-point delays within the I/O buffer and assume the presence of the capacitive test load with no other details about the board specified. The default size of the load is based on the I/O standard selected for the pin. Timing is measured to the FPGA pin with no signal integrity analysis details.

The **Enable Advanced I/O Timing** option expands the details in I/O timing reports by taking board topology and termination components into account. A complete point-to-point board trace model is defined and accounted for in the timing analysis. This ability to define a board trace model is an example of how the Quartus Prime software is "board-aware."

ISO 9001:2008 Registered

In this case, timing and signal integrity metrics between the I/O buffer and the defined far end load are analyzed and reported in enhanced reports generated by the Quartus Prime TimeQuest Timing Analyzer.

Related Information

- **I/O Management** on page 2-1
  For more information about defining capacitive test loads or how to use the **Enable Advanced I/O Timing** option to configure a board trace model.

Signal Integrity Simulations with HSPICE and IBIS Models

The Quartus Prime software can export accurate HSPICE models with the built-in HSPICE Writer. You can run signal integrity simulations with these complete HSPICE models in Synopsys HSPICE. IBIS models of the FPGA I/O buffers are also created easily with the Quartus Prime IBIS Writer.

You can run signal integrity simulations with these complete HSPICE models in Synopsys HSPICE.

You can integrate IBIS models into any third-party simulation tool that supports them, such as the Mentor Graphics® Hyperlynx software. With the ability to create industry-standard model definition files quickly, you can build accurate simulations that can provide data to help improve board-level signal integrity.

The I/O's IBIS and HSPICE model creation available in the Quartus Prime software can help prevent problems before a costly board respin is required. In general, creating and running accurate simulations is difficult and time consuming. The tools in the Quartus Prime software automate the I/O model setup and creation process by configuring the models specifically for your design. With these tools, you can set up and run accurate simulations quickly and acquire data that helps guide your FPGA and board design.

The information about signal integrity in this chapter refers to board-level signal integrity based on I/O buffer configuration and board parameters, not simultaneous switching noise (SSN), also known as ground bounce or V_{CC} sag. SSN is a product of multiple output drivers switching at the same time, causing an overall drop in the voltage of the chip's power supply. This can cause temporary glitches in the specified level of ground or V_{CC} for the device.

This chapter is intended for FPGA and board designers and includes details about the concepts and steps involved in getting designs simulated and how to adjust designs to improve board-level timing and signal integrity. Also included is information about how to create accurate models from the Quartus Prime software and how to use those models in simulation software.

The information in this chapter is meant for those who are familiar with the Quartus Prime software and basic concepts of signal integrity and the design techniques and components in good PCB design. Finally, you should know how to set up simulations and use your selected third-party simulation tool.

Related Information

- **AN 315: Guidelines for Designing High-Speed FPGA PCBs**
  For a more information about SSN and ways to prevent it.
- **Altera Signal Integrity Center**
  For information about basic signal integrity concepts and signal integrity details pertaining to Altera FPGA devices.

I/O Model Selection: IBIS or HSPICE

The Quartus Prime software can export two different types of I/O models that are useful for different simulation situations, IBIS models and HSPICE models.

IBIS models define the behavior of input or output buffers through the use of voltage-current (V-I) and voltage-time (V-t) data tables. HSPICE models, often referred to as HSPICE decks, include complete physical descriptions of the transistors and parasitic capacitances that make up an I/O buffer along with all the parameter settings required to run a simulation. The HSPICE decks generated by the Quartus Prime software are preconfigured with the I/O standard, voltage, and pin loading settings for each pin in your design.

The choice of I/O model type is based on many factors.

Table 14-1: IBIS and HSPICE Model Comparison

Feature	IBIS Model	HSPICE Model
I/O Buffer Description	**Behavioral**—I/O buffers are described by voltage-current and voltage-time tables in typical, minimum, and maximum supply voltage cases.	**Physical**—I/O buffers and all components in a circuit are described by their physical properties, such as transistor characteristics and parasitic capacitances, as well as their connections to one another.
Model Customization	**Simple and limited**—The model completely describes the I/O buffer and does not usually have to be customized.	**Fully customizable**—Unless connected to an arbitrary board description, the description of the board trace model must be customized in the model file. All parameters of the simulation are also adjustable.
Simulation Set Up and Run Time	**Fast**—Simulations run quickly after set up correctly.	**Slow**—Simulations take time to set up and take longer to run and complete.
Simulation Accuracy	**Good**—For most simulations, accuracy is sufficient to make useful adjustments to the FPGA and/or board design to improve signal integrity.	**Excellent**—Simulations are highly accurate, making HSPICE simulation almost a requirement for any high-speed design where signal integrity and timing margins are tight.
Third-Party Tool Support	**Excellent**—Almost all third-party board simulation tools support IBIS.	**Good**—Most third-party tools that support SPICE support HSPICE. However, Synopsys HSPICE is required for simulations of Altera's encrypted HSPICE models.

Related Information

AN 283: Simulating Altera Devices with IBIS Models
For more information about IBIS files created by the Quartus Prime IBIS Writer and IBIS files in general, as well as links to websites with detailed information.

FPGA to Board Signal Integrity Analysis Flow

Board signal integrity analysis can take place at any point in the FPGA design process and is often performed before and after board layout. If it is performed early in the process as part of a pre-PCB layout analysis, the models used for simulations can be more generic.

These models can be changed as much as required to see how adjustments improve timing or signal integrity and help with the design and routing of the PCB. Simulations and the resulting changes made at this stage allow you to analyze "what if" scenarios to plan and implement your design better. To assist with early board signal integrity analysis, you can download generic IBIS model files for each device family and obtain HSPICE buffer simulation kits from the "Board Level Tools" section of the EDA Tool Support Resource Center.

Typically, if board signal integrity analysis is performed late in the design, it is used for a post-layout verification. The inputs and outputs of the FPGA are defined, and required board routing topologies and constraints are known. Simulations can help you find problems that might still exist in the FPGA or board design before fabrication and assembly. In either case, a simple process flow illustrates how to create accurate IBIS and HSPICE models from a design in the Quartus Prime software and transfer them to third-party simulation tools.

Your design depends on the type of model, IBIS or HSPICE, that you use for your simulations. When you understand the steps in the analysis flow, refer to the section of this chapter that corresponds to the model type you are using.

Figure 14-1: Third-Party Board Signal Integrity Analysis Flow
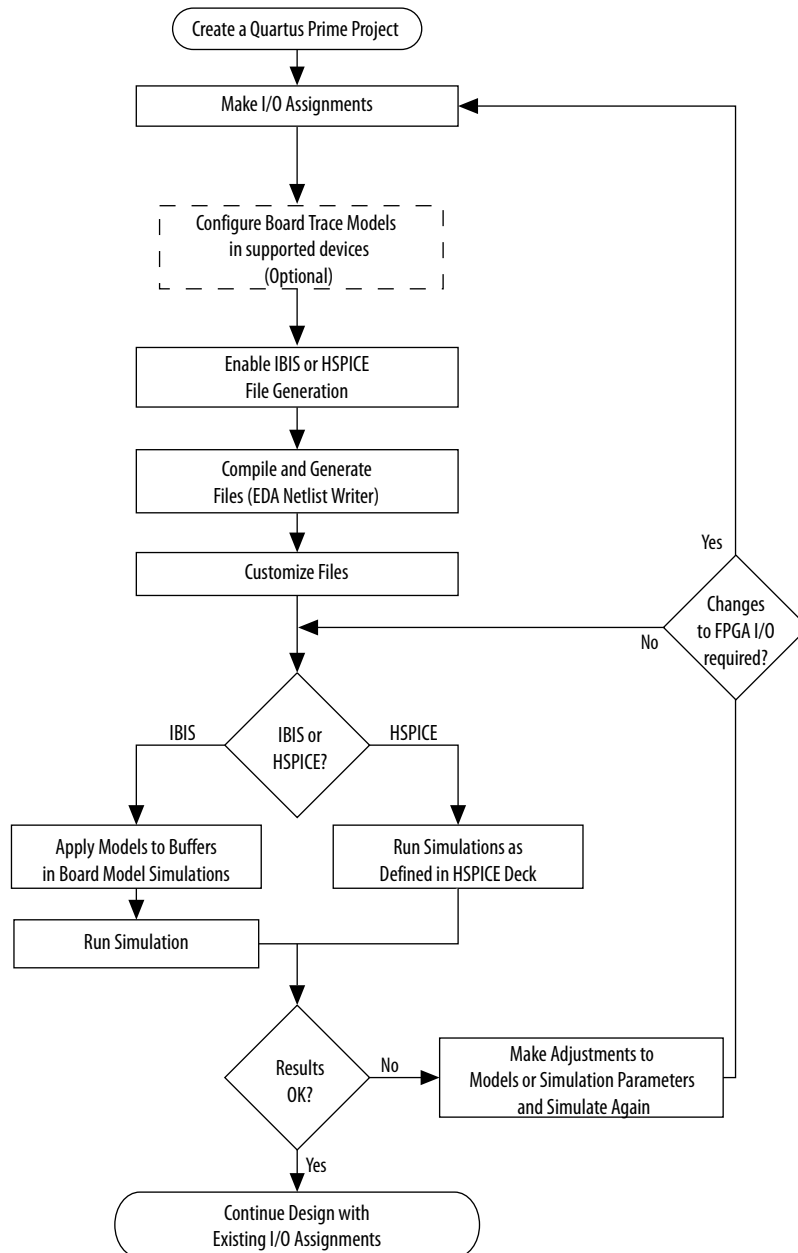
Related Information

EDA Tool Support Resource Center

For more information, generic IBIS model files for each device family, and to obtain HSPICE buffer simulation kits.

Create I/O and Board Trace Model Assignments

You can configure a board trace model for output signals or for bidirectional signals in output mode. You can then automatically transfer its description to HSPICE decks generated by the HSPICE Writer. This helps improve simulation accuracy.

To configure a board trace model, in the **Settings** dialog box, in the **TimeQuest Timing Analyzer** page, turn on the **Enable Advanced I/O Timing** option and configure the board trace model assignment settings for each I/O standard used in your design. You can add series or parallel termination, specify the transmission line length, and set the value of the far-end capacitive load. You can configure these parameters either in the Board Trace Model view of the Pin Planner, or click **SettingsDeviceDevice and Pin Options**.

The Quartus Prime software can generate IBIS models and HSPICE decks without having to configure a board trace model with the **Enable Advanced I/O Timing** option. In fact, IBIS models ignore any board trace model settings other than the far-end capacitive load. If any load value is set other than the default, the delay given by IBIS models generated by the IBIS Writer cannot be used to account correctly for the double counting problem. The load value mismatch between the IBIS delay and the t_{CO} measurement of the Quartus Prime software prevents the delays from being safely added together. Warning messages displayed when the EDA Netlist Writer runs indicate when this mismatch occurs.

Related Information

- **I/O Management** on page 2-1
  For information about how to use the **Enable Advanced I/O Timing** option and configure board trace models for the I/O standards used in your design.

Output File Generation

IBIS and HSPICE model files are not generated by the Quartus Prime software by default. To generate or update the files automatically during each project compilation, select the type of file to generate and a location where to save the file in the project settings.

The IBIS and HSPICE Writers in the Quartus Prime software are run as part of the EDA Netlist Writer during normal project compilation. If either writer is turned on in the project settings, IBIS or HSPICE files are created and stored in the specified location. For IBIS, a single file is generated containing information about all assigned pins. HSPICE file generation creates separate files for each assigned pin. You can run the EDA Netlist Writer separately from a full compilation in the Quartus Prime software or at the command line.

Customize the Output Files

The files generated by either the IBIS or HSPICE Writer are text files that you can edit and customize easily for design or experimentation purposes.

IBIS files downloaded from the Altera website must be customized with the correct RLC values for the specific device package you have selected for your design. IBIS files generated by the IBIS Writer do not require this customization because they are configured automatically with the RLC values for your selected device. HSPICE decks require modification to include a detailed description of your board. With **Enable Advanced I/O Timing** turned on and a board trace model defined in the Quartus Prime software, generated HSPICE decks automatically include that model's parameters. However, Altera recommends that you replace that model with a more detailed model that describes your board design more accurately. A default simulation included in the generated HSPICE decks measures delay between the FPGA and the far-end device. You can make additions or adjustments to the default simulation in the generated files to change the parameters of the default simulation or to perform additional measurements.

Set Up and Run Simulations in Third-Party Tools

When you have generated the files, you can use them to perform simulations in your selected simulation tool.

With IBIS models, you can apply them to input, output, or bidirectional buffer entities and quickly set up and run simulations. For HSPICE decks, the simulation parameters are included in the files. Open the files in Synopsys HSPICE and run simulations for each pin as required.

With HSPICE decks generated from the HSPICE Writer, the double counting problem is accounted for, which ensures that your simulations are accurate. Simulations that involve IBIS models created with anything other than the default loading settings in the Quartus Prime software must take the change in the size of the load between the IBIS delay and the Quartus Prime t_{CO} measurement into account. Warning messages during compilation alert you to this change.

Interpret Simulation Results

If you encounter timing or signal integrity issues with your high-speed signals after running simulations, you can make adjustments to I/O assignment settings in the Quartus Prime software.

These could include such things as drive strength or I/O standard, or making changes to your board routing or topology. After regenerating models in the Quartus Prime software based on the changes you have made, rerun the simulations to check whether your changes corrected the problem.

Simulation with IBIS Models

IBIS models provide a way to run accurate signal integrity simulations quickly. IBIS models describe the behavior of I/O buffers with voltage-current and voltage-time data curves.

Because of their behavioral nature, IBIS models do not have to include any information about the internal circuit design of the I/O buffer. Most component manufacturers, including Altera, provide IBIS models for free download and use in signal integrity analysis simulation tools. You can download generic device family IBIS models from the Altera website for early design simulation or use the IBIS Writer to create custom IBIS models for your existing design.

Elements of an IBIS Model

An IBIS model file (**.ibs**) is a text file that describes the behavior of an I/O buffer across minimum, typical, and maximum temperature and voltage ranges with a specified test load.

The tables and values specified in the IBIS file describe five basic elements of the I/O buffer.

Figure 14-2: Five Basic Elements of an I/O Buffer in IBIS Models
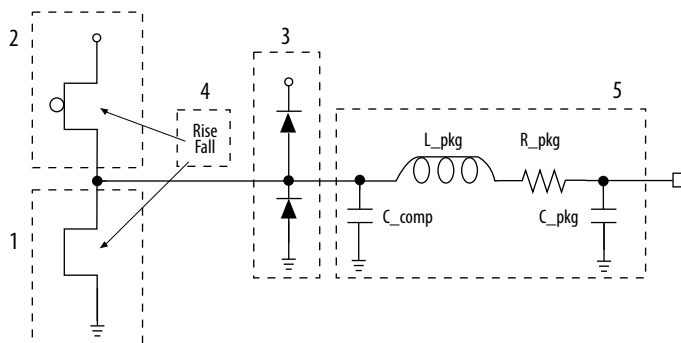
The following elements correspond to each numbered block.

1. **Pulldown**—A voltage-current table describes the current when the buffer is driven low based on a pull-down voltage range of $-V_{CC}$ to 2 V_{CC}.
2. **Pullup**—A voltage-current table describes the current when the buffer is driven high based on a pull-up voltage range of $-V_{CC}$ to V_{CC}.
3. **Ground and Power Clamps**—Voltage-current tables describe the current when clamping diodes for electrostatic discharge (ESD) are present. The ground clamp voltage range is $-V_{CC}$ to V_{CC}, and the power clamp voltage range is $-V_{CC}$ to ground.
4. **Ramp and Rising/Falling Waveform**—A voltage-time (dv/dt) ratio describes the rise and fall time of the buffer during a logic transition. Optional rising and falling waveform tables can be added to more accurately describe the characteristics of the rising and falling transitions.
5. **Total Output Capacitance and Package RLC**—The total output capacitance includes the parasitic capacitances of the output pad, clamp diodes (if present), and input transistors. The package RLC is device package-specific and defines the resistance, inductance, and capacitance of the bond wire and pin of the I/O.

Related Information

AN 283: Simulating Altera Devices with IBIS Models

For more information about IBIS models and Altera-specific features, including links to the official IBIS specification.

Creating Accurate IBIS Models

There are two methods to obtain Altera device IBIS files for your board-level signal integrity simulations. You can download generic IBIS models from the Altera website. You can also use the IBIS writer in the Quartus Prime software to create design-specific models.

The IBIS file generated by the Quartus Prime software contains models of both input and output termination, and is supported for IBIS model versions of 4.2 and later .Arria V, Cyclone V, and Stratix V device families allow the use of bidirectional I/O with dynamic on-chip termination (OCT).

Dynamic OCT is used where a signal uses a series on-chip termination during output operation and a parallel on-chip termination during input operation. Typically this is used in Altera External Memory Interface IP.

The Quartus Prime IBIS dynamic OCT IBIS model names end in **g50c_r50c**. For example : **sstl15i_ctnio_g50c_r50c**.

In the simulation tool, the IBIS model is attached to a buffer.

- When the buffer is assigned as an output, use the series termination r50c.
- When the buffer is assigned as an input, use the parallel termination g50c.

Download IBIS Models

Altera provides IBIS models for almost all FPGA and FPGA configuration devices. You can use the IBIS models from the website to perform early simulations of the I/O buffers you expect to use in your design as part of a pre-layout analysis.

Downloaded IBIS models have the RLC package values set to one particular device in each device family.

The **.ibs** file can be customized for your device package and can be used for any simulation. IBIS models downloaded and used for simulations in this manner are generic. They describe only a certain set of models listed for each device on the Altera IBIS Models page of the Altera website. To create customized models for your design, use the IBIS Writer as described in the next section.

To simulate your design with the model accurately, you must adjust the RLC values in the IBIS model file to match the values for your particular device package by performing the following steps:

1. Download and expand the ZIP file (**.zip**) of the IBIS model for the device family you are using for your design. The **.zip** file contains the **.ibs** file along with an IBIS model user guide and a model data correlation report.
2. Download the Package RLC Values spreadsheet for the same device family.
3. Open the spreadsheet and locate the row that describes the device package used in your design.
4. From the package's **I/O** row, copy the minimum, maximum, and typical values of resistance, inductance, and capacitance for your device package.
5. Open the **.ibs** file in a text editor and locate the [Package] section of the file.
6. Overwrite the listed values copied with the values from the spreadsheet and save the file.

Related Information

Altera IBIS Models
For information about whether models for your selected device are available.

Generate Custom IBIS Models with the IBIS Writer

If you have started your FPGA design and have created custom I/O assignments, you can use the Quartus Prime IBIS Writer to create custom IBIS models to accurately reflect your assignments.

Examples of custom assignments include drive strength settings or the enabling of clamping diodes for ESD protection. IBIS models created with the IBIS Writer take I/O assignment settings into account.

If the **Enable Advanced I/O Timing** option is turned off, the generated **.ibs** files are based on the load value setting for each I/O standard on the **Capacitive Loading** page of the **Device and Pin Options** dialog box in the **Device** dialog box. With the **Enable Advanced I/O Timing** option turned on, IBIS models use an effective capacitive load based on settings found in the board trace model on the **Board Trace Model** page in the **Device and Pin Options** dialog box or the **Board Trace Model** view in the Pin Planner. The effective capacitive load is based on the sum of the **Near capacitance**, **Transmission line distributed capacitance**, and the **Far capacitance** settings in the board trace model. Resistance values and transmission line inductance values are ignored.

Note: If you made any changes from the default load settings, the delay in the generated IBIS model cannot safely be added to the Quartus Prime t_{CO} measurement to account for the double counting problem. This is because the load values between the two delay measurements do not match. When this happens, the Quartus Prime software displays warning messages when the EDA Netlist Writer runs to remind you about the load value mismatch.

Related Information

- **Generating IBIS Output Files with the Quartus Prime Software**
  For step-by-step instructions on how to generate IBIS models with the Quartus Prime software, refer to Quartus Prime Help.
- **AN 283: Simulating Altera Devices with IBIS Models**
  For more information about IBIS model generation.

Design Simulation Using the Mentor Graphics HyperLynx® Software

You must integrate IBIS models downloaded from the Altera website or created with the Quartus Prime IBIS Writer into board design simulations to accurately model timing and signal integrity.

The HyperLynx software from Mentor Graphics is one of the most popular tools for design simulation. The HyperLynx software makes it easy to integrate IBIS models into simulations.

The HyperLynx software is a PCB analysis and simulation tool for high-speed designs, consisting of two products, LineSim and BoardSim. LineSim is an early simulation tool. Before any board routing takes place, LineSim is used to simulate "what if" scenarios to assist in creating routing rules and defining board parameters. BoardSim is a post-layout tool used to analyze existing board routing. Specific nets are selected from a board layout file and simulated in a manner similar to LineSim. With board and routing parameters, and surrounding signal routing known, highly accurate simulations of the final fabricated PCB are possible. This section focuses on LineSim. Because the process of creating and running simulations is very similar for both LineSim and BoardSim, the details of IBIS model use in LineSim applies to simulations in BoardSim.

Simulations in LineSim are configured using a schematic GUI to create connections and topologies between I/O buffers, route trace segments, and termination components. LineSim provides two methods for creating routing schematics: cell-based and free-form. Cell-based schematics are based on fixed cells consisting of typical placements of buffers, trace impedances, and components. Parts of the grid-based cells are filled with the desired objects to create the topology. A topology in a cell-based schematic is limited by the available connections within and between the cells.

A more robust and expandable way to create a circuit schematic for simulation is to use the free-form schematic format in LineSim. The free-form schematic format makes it easy to place parts into any configuration and edit them as required. This section describes the use of IBIS models with free-form schematics, but the process is nearly identical for cell-based schematics.

Figure 14-3: HyperLynx LineSim Free-Form Schematic Editor
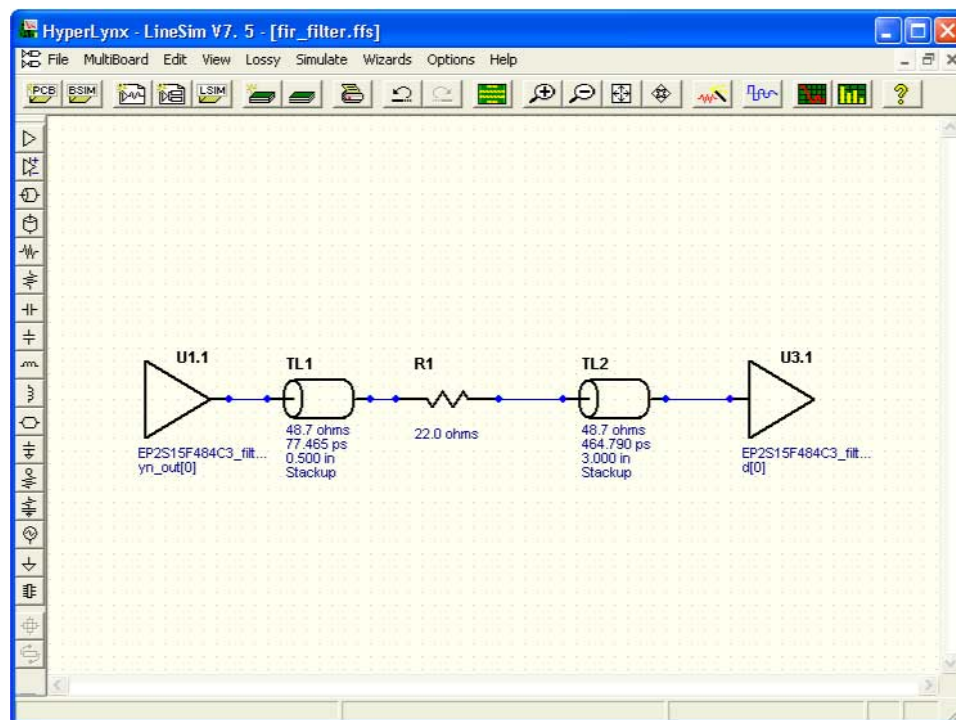
When you use HyperLynx software to perform simulations, you typically perform the following steps:

1. Create a new LineSim free-form schematic document and set up the board stackup for your PCB using the Stackup Editor. In this editor, specify board layer properties including layer thickness, dielectric constant, and trace width.

2. Create a circuit schematic for the net you want to simulate. The schematic represents all the parts of the routed net including source and destination I/O buffers, termination components, transmission line segments, and representations of impedance discontinuities such as vias or connectors.

3. Assign IBIS models to the source and destination I/O buffers to represent their behavior during operation.

4. Attach probes from the digital oscilloscope that is built in to LineSim to points in the circuit that you want to monitor during simulation. Typically, at least one probe is attached to the pin of a destination I/O buffer. For differential signals, you can attach a differential probe to both the positive and negative pins at the destination.

5. Configure and run the simulation. You can simulate a rising or falling edge and test the circuit under different drive strength conditions.

6. Interpret the results and make adjustments. Based on the waveforms captured in the digital oscillo-scope, you can adjust anything in the circuit schematic to correct any signal integrity issues, such as overshoot or ringing. If necessary, you can make I/O assignment changes in the Quartus Prime software, regenerate the IBIS file with the IBIS Writer, and apply the updated IBIS model to the buffers in your HyperLynx software schematic.

7. Repeat the simulations and circuit adjustments until you are satisfied with the results. When the operation of the net meets your design requirements, implement changes to your I/O assignments in the Quartus Prime software and/or adjust your board routing constraints, component values, and placement to match the simulation.

Related Information

www.mentor.com

For more information about HyperLynx software, including schematic creation, simulation setup, model usage, product support, licensing, and training.

Configuring LineSim to Use Altera IBIS Models

You must configure LineSim to find and use the downloaded or generated IBIS models for your design. To do this, add the location of your **.ibs** file or files to the LineSim Model Library search path. Then you apply a selected model to a buffer in your schematic.

To add the Quartus Prime software's default IBIS model location, *<project directory>*/**board/ibis**, to the HyperLynx LineSim model library search path, perform the following steps in LineSim:

1. From the Options menu, click **Directories**. The **Set Directories** dialog box appears. The **Model-library file path(s)** list displays the order in which LineSim searches file directories for model files.

Figure 14-4: LineSim Set Directories Dialog Box
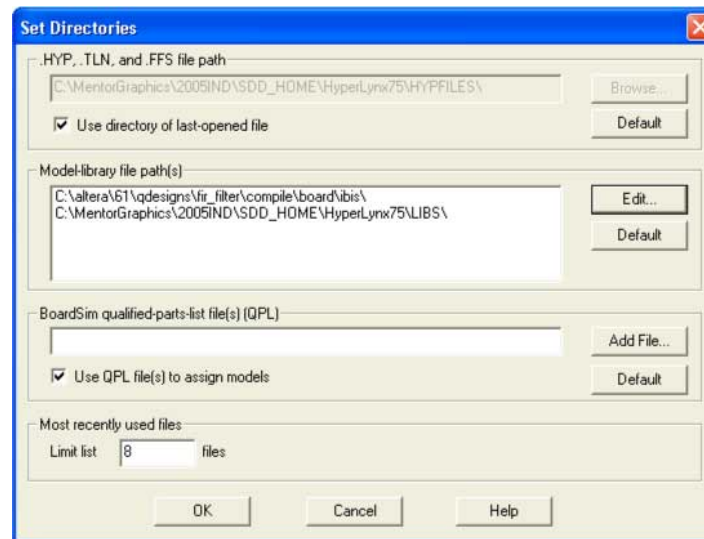
2. Click **Edit**. A dialog box appears where you can add directories and adjust the order in which LineSim searches them.

Figure 14-5: LineSim Select Directories Dialog Box
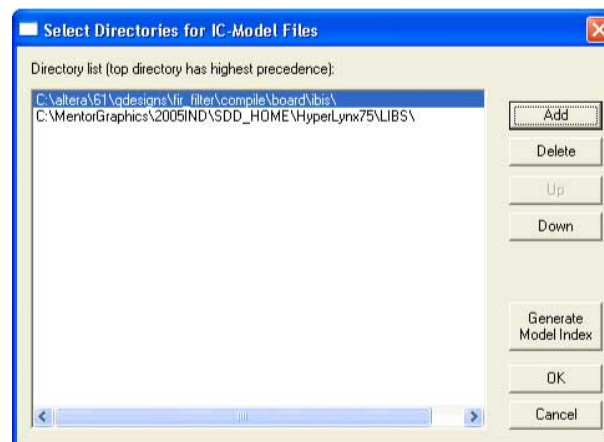
3. Click **Add**

4. Browse to the default IBIS model location, *<project directory>*/**board/ibis**. Click **OK**.

5. Click **Up** to move the IBIS model directory to the top of the list. Click **Generate Model Index** to update LineSim's model database with the models found in the added directory.

6. Click **OK**. The IBIS model directory for your project is added to the top of the Model-library file path(s) list.

7. To close the **Set Directories** dialog box, click **OK**.

Integrating Altera IBIS Models into LineSim Simulations

When the location for IBIS files has been set, you can assign the downloaded or generated IBIS models to the buffers in your schematic. To do this, perform the following steps:

1. Double-click a buffer symbol in your schematic to open the **Assign Models** dialog box. You can also click **Assign Models** from the buffer symbol's right-click menu.

Figure 14-6: LineSim Assign Model Dialog Box
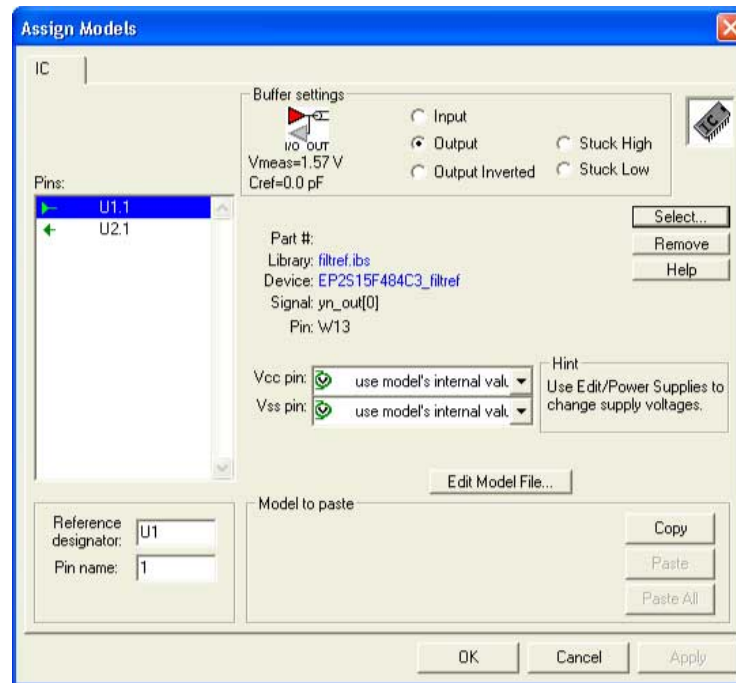
2. The pin of the buffer symbol you selected should be highlighted in the **Pins** list. If you want to assign a model to a different symbol or pin, select it from the list.
3. Click **Select**. The **Select IC Model** dialog box appears.

Figure 14-7: LineSim Select IC Model Dialog Box
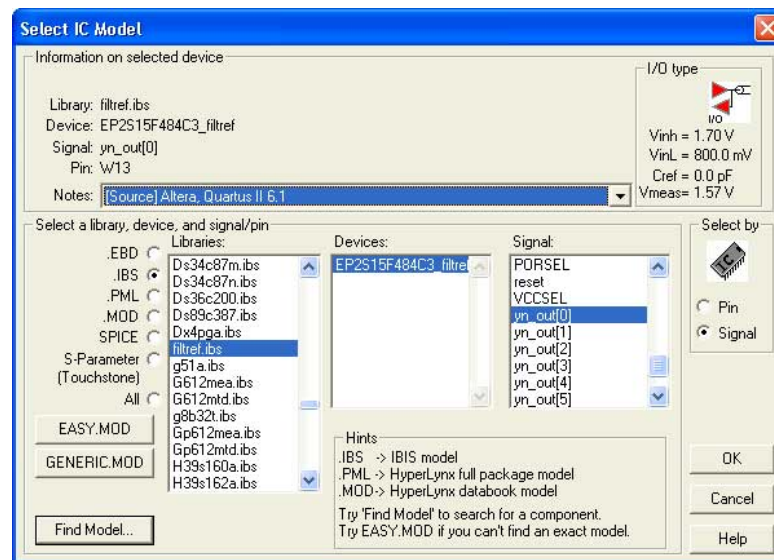
4. To filter the list of available libraries to display only IBIS models, select **.IBS**. Scroll through the **Libraries** list, and click the name of the library for your design. By default, this is *<project name>*.**ibs**.

5. The device for your design should be selected as the only item in the **Devices** list. If not, select your device from the list.

6. From the **Signal** list, select the name of the signal you want to simulate. You can also choose to select by device pin number.

7. Click **OK**. The **Assign Models** dialog box displays the selected **.ibs** file and signal.

8. If applicable to the signal you chose, adjust the buffer settings as required for the simulation.

9. Select and configure other buffer pins from the **Pins** list in the same manner.

10. Click **OK** when all I/O models are assigned.

Running and Interpreting LineSim Simulations

You can now run any desired simulations and make adjustments to the I/O assignments or simulation parameters as required.

For example, if you see too much overshoot in the simulated signal at the destination buffer after running a simulation, you could adjust the drive strength I/O assignment setting to a lower value. Regenerate the **.ibs** file, and run the simulation again to verify whether the change fixed the problem.

Figure 14-8: Example of Overshoot in HyperLynx with IBIS Models
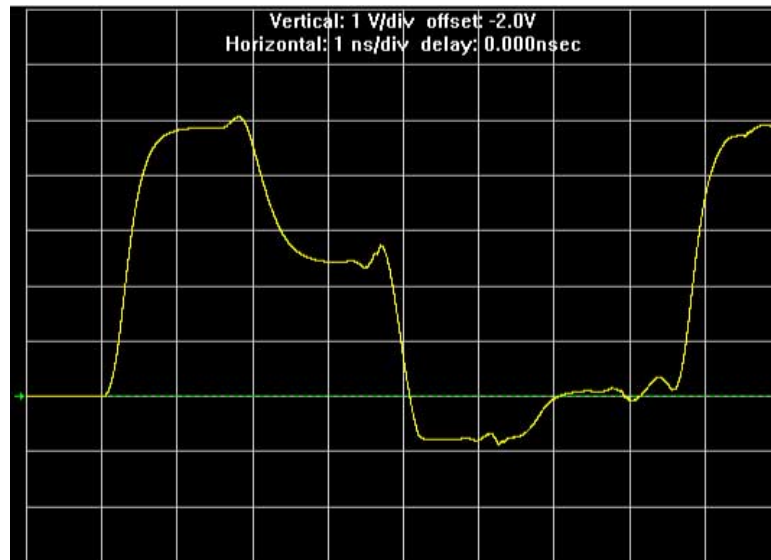
If you see a discontinuity or other anomalies at the destination, such as slow rise and fall times, adjust the termination scheme or termination component values. After making these changes, rerun the simulation to check whether your adjustments solved the problem. In this case, it is not necessary to regenerate the **.ibs** file.

Figure 14-9: Example of Signal Integrity Anomaly in HyperLynx with IBIS Models
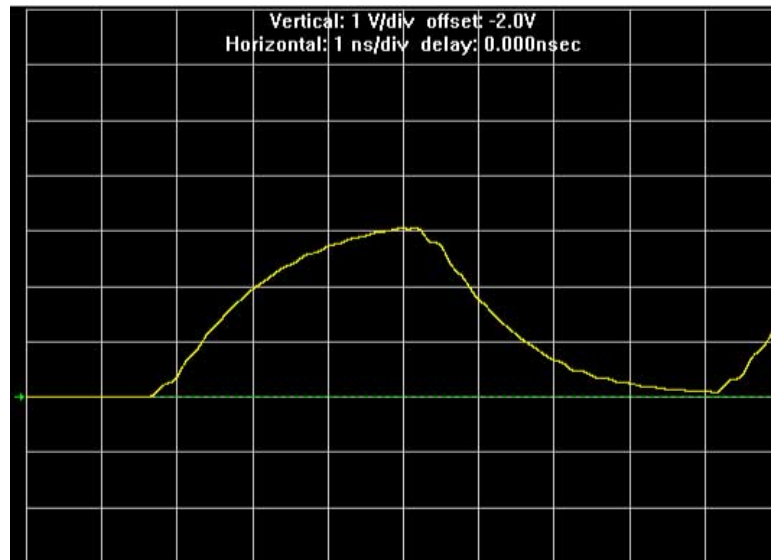
Related Information

Altera Signal Integrity Center
For more information about board-level signal integrity and to learn about ways to improve it with simple changes to your design.

Simulation with HSPICE Models

HSPICE decks are used to perform highly accurate simulations by describing the physical properties of all aspects of a circuit precisely. HSPICE decks describe I/O buffers, board components, and all of the connections between them, as well as defining the parameters of the simulation to be run.

By their nature, HSPICE decks are highly customizable and require a detailed description of the circuit under simulation. For devices that support advanced I/O timing, when **Enable Advanced I/O Timing** is turned on, the HSPICE decks generated by the Quartus Prime HSPICE Writer automatically include board components and topology defined in the Board Trace Model. Configure the board components and topology in the Pin Planner or in the **Board Trace Model** tab of the **Device and Pin Options** dialog box. All HSPICE decks generated by the Quartus Prime software include compensation for the double count problem. You can simulate with the default simulation parameters built in to the generated HSPICE decks or make adjustments to customize your simulation.

Related Information

[The Double Counting Problem in HSPICE Simulations](#) on page 14-17

Supported Devices and Signaling

The HSPICE Writer in the Quartus Prime software supports Arria, Cylcone, and Stratix devices for the creation of a board trace model in the Quartus Prime software for automatic inclusion in an HSPICE deck.

The HSPICE files include the board trace description you create in the Board Trace Model view in the Pin Planner or the **Board Trace Model** tab in the **Device and Pin Options** dialog box.

Note: Note that for Arria 10 devices, you may need to download the Encrypted HSPICE model from the Altera website.

Related Information

[I/O Management](#) on page 2-1
For more information about the **Enable Advanced I/O Timing** option and configuring board trace models for the I/O standards in your design.

[SPICE Models for Altera Devices](#)
For more information about the Encrypted HSPICE model.

Accessing HSPICE Simulation Kits

You can access the available HSPICE models with the Quartus Prime software's HSPICE Writer tool and also at the Spice Models for Altera Devices web page.

The Quartus Prime software HSPICE Writer tool removes many common sources of user error from the I/O simulation process. The HSPICE Writer tool automatically creates preconfigured I/O simulation spice decks that only require the addition of a user board model. All the difficult tasks required to configure the I/O modes and interpret the timing results are handled automatically by the HSPICE Writer tool.

Related Information

[Spice Models for Altera Devices](#)
For more information about downloadable HSPICE models.

The Double Counting Problem in HSPICE Simulations

Simulating I/Os using accurate models is extremely helpful for finding and fixing FPGA I/O timing and board signal integrity issues before any boards are built. However, the usefulness of such simulations is directly related to the accuracy of the models used and whether the simulations are set up and performed correctly. To ensure accuracy in models and simulations created for FPGA output signals, the timing hand-off between t_{CO} timing in the Quartus Prime software and simulation-based board delay must be taken into account. If this hand-off is not handled correctly, the calculated delay could either count some of the delay twice or even miss counting some of the delay entirely.

Defining the Double Counting Problem

The double counting problem is inherent to the method output timing is analyzed versus the method used for HSPICE models. The timing analyzer tools in the Quartus Prime software measure delay timing for an output signal from the core logic of the FPGA design through the output buffer ending at the FPGA pin with a default capacitive load or a specified value for the selected I/O standard. This measurement is the t_{CO} timing variable.

Figure 14-10: Double Counting Problem
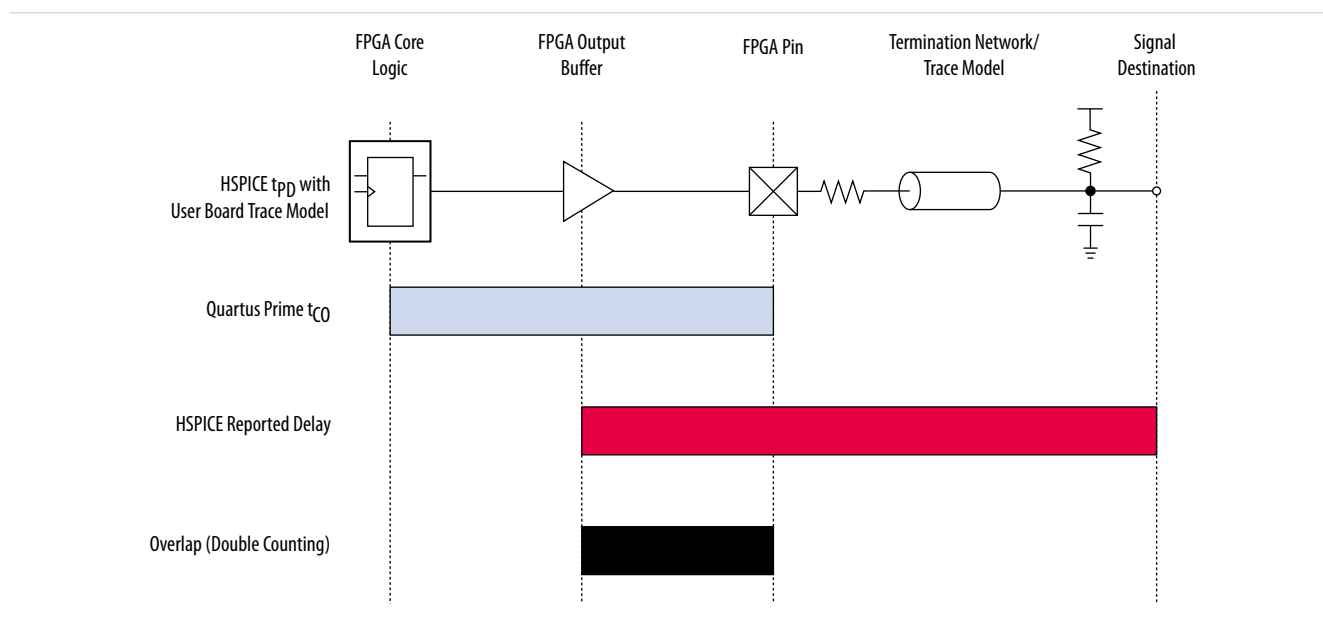
HSPICE models for board simulation measure t_{PD} (propagation delay) from an arbitrary reference point in the output buffer, through the device pin, out along the board routing, and ending at the signal destination.

It is apparent immediately that if these two delays were simply added together, the delay between the output buffer and the device pin would be counted twice in the calculation. A model or simulation that does not account for this double count would create overly pessimistic simulation results, because the double-counted delay can limit I/O performance artificially. To fix the problem, it might seem that simply subtracting the overlap between t_{CO} and t_{PD} would account for the double count. However, this adjustment would not be accurate because each measurement is based on a different load.

Note: Input signals do not exhibit this problem because the HSPICE models for inputs stop at the FPGA pin instead of at the input buffer. In this case, simply adding the delays together produces an accurate measurement of delay timing.

The Solution to Double Counting

To adjust the measurements to account for the double-counting, the delay between the arbitrary point in the output buffer selected by the HSPICE model and the FPGA pin must be subtracted from either t_{CO} or t_{PD} before adding the results together. The subtracted delay must also be based on a common load between the two measurements. This is done by repeating the HSPICE model measurement, but with the same load used by the Quartus Prime software for the t_{CO} measurement.

Figure 14-11: Common Test Loads Used for Output Timing
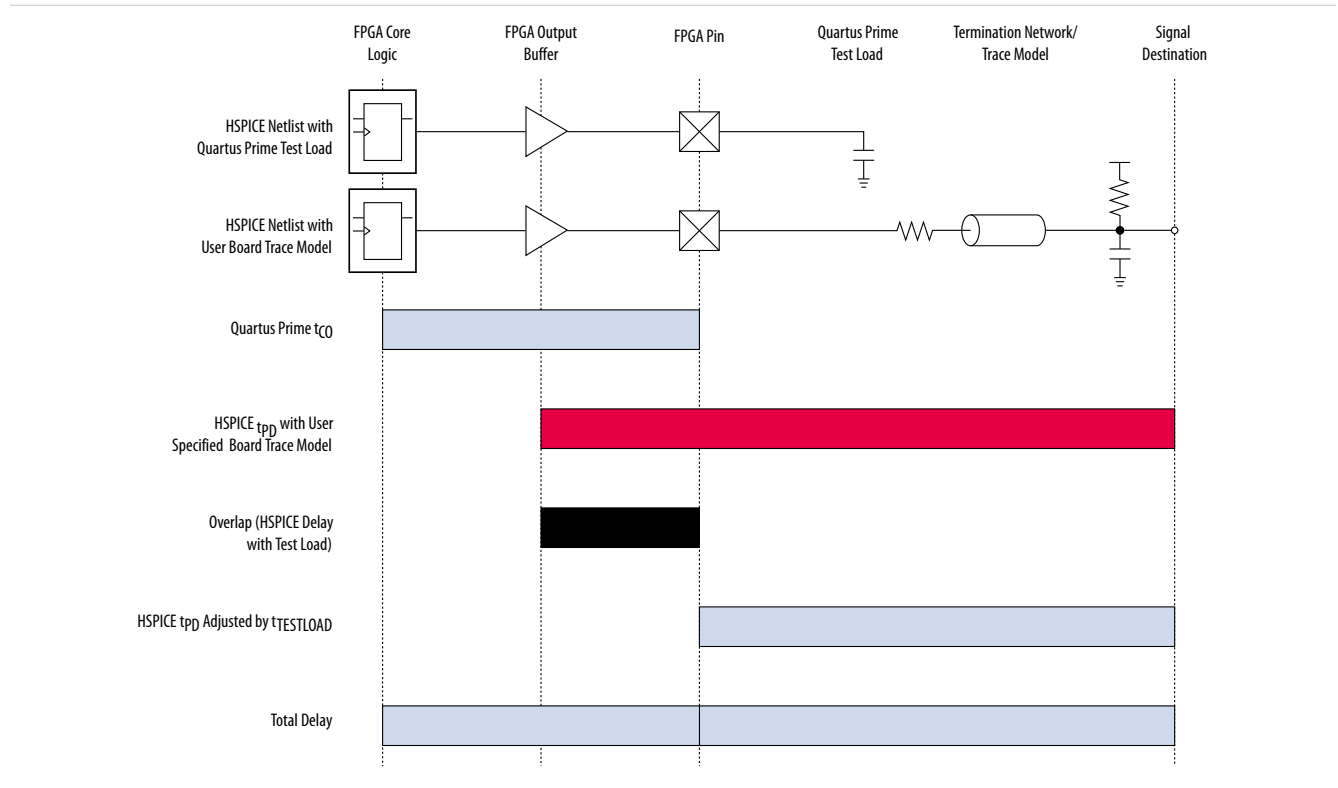
With $t_{TESTLOAD}$ known, the total delay is calculated for the output signal from the FPGA logic to the signal destination on the board, accounting for the double count.

$t_{delay} = t_{CO} + (t_{PD} - t_{TESTLOAD})$

The preconfigured simulation files generated by the HSPICE Writer in the Quartus Prime software are designed to account for the double-counting problem based on this calculation automatically. Performing accurate timing simulations is easy without having to make adjustments for double counting manually.

HSPICE Writer Tool Flow

This section includes information to help you get started using the Quartus Prime software HSPICE Writer tool. The information in this section assumes you have a basic knowledge of the standard Quartus Prime software design flow, such as project and assignment creation, compilation, and timing analysis.

Related Information

Quartus Prime Handbook
For additional information about standard design flows.

Applying I/O Assignments

The first step in the HSPICE Writer tool flow is to configure the I/O standards and modes for each of the pins in your design properly. In the Quartus Prime software, these settings are represented by assignments that map I/O settings, such as pin selection, and I/O standard and drive strength, to corresponding signals in your design.

The Quartus Prime software provides multiple methods for creating these assignments:

- Using the Pin Planner
- Using the assignment editor
- Manually editing the **.qsf** file
- By making assignments in a scripted Quartus Prime flow using Tcl

Enabling HSPICE Writer

You must enable the HSPICE Writer in the **Settings** dialog box of the Quartus Prime software to generate the HSPICE decks from the Quartus Prime software.

Figure 14-12: EDA Tool Settings: Board Level Options Dialog Box
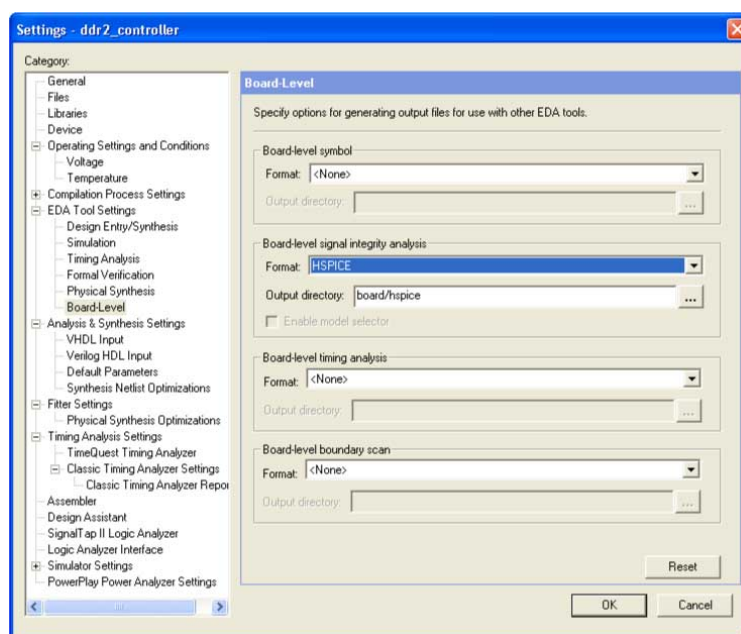
Enabling HSPICE Writer Using Assignments

You can also use HSPICE Writer in conjunction with a scripted Tcl flow. To enable HSPICE Writer during a full compile, include the following lines in your Tcl script.

Enable HSPICE Writer

```
set_global_assignment -name EDA_BOARD_DESIGN_SIGNAL_INTEGRITY_TOOL \
"HSPICE (Signal Integrity)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT HSPICE \
-section_id eda_board_design_signal_integrity
set_global_assignment -name EDA_NETLIST_WRITER_OUTPUT_DIR <output_directory> \
-section_id eda_board_design_signal_integrity
```

As with command-line invocation, specifying the output directory is optional. If not specified, the output directory defaults to **board/hspice**.

Naming Conventions for HSPICE Files

HSPICE Writer automatically generates simulation files and names them using the following naming convention: *<device>_<pin #>_<pin_name>_<in/out>*.**sp**.

For bidirectional pins, two spice decks are produced; one with the I/O buffer configured as an input, and the other with the I/O buffer configured as an output.

The Quartus Prime software supports alphanumeric pin names that contain the underscore (_) and dash (–) characters. Any illegal characters used in file names are converted automatically to underscores.

Related Information

- **Sample Output for I/O HSPICE Simulation Deck** on page 14-30
- **Sample Input for I/O HSPICE Simulation Deck** on page 14-26

Invoking HSPICE Writer

After HSPICE Writer is enabled, the HSPICE simulation files are generated automatically each time the project is completely compiled. The Quartus Prime software also provides an option to generate a new set of simulation files without having to recompile manually. In the Processing menu, click **Start EDA Netlist Writer** to generate new simulation files automatically.

Note: You must perform both Analysis & Synthesis and Fitting on a design before invoking the HSPICE Writer tool.

Invoking HSPICE Writer from the Command Line

If you use a script-based flow to compile your project, you can create HSPICE model files by including the following commands in your Tcl script (**.tcl** file).

Create HSPICE Model Files

```
set_global_assignment -name EDA_BOARD_DESIGN_SIGNAL_INTEGRITY_TOOL \
"HSPICE (Signal Integrity)"
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT HSPICE \
-section_ideda_board_design_signal_integrity
set_global_assignment -name EDA_NETLIST_WRITER_OUTPUT_DIR <output_directory> \
-section_id eda_board_design_signal_integrity
```

The *<output_directory>* option specifies the location where HSPICE model files are saved. By default, the *<project directory>*/**board/hspice** directory is used.

Invoke HSPICE Writer

To invoke the HSPICE Writer tool through the command line, type:

```
quartus_eda.exe <project_name> --board_signal_integrity=on --format=HSPICE \
--output_directory=<output_directory>
```

<output_directory> specifies the location where the generated spice decks will be written (relative to the design directory). This is an optional parameter and defaults to **board/hspice**.

Customizing Automatically Generated HSPICE Decks

HSPICE models generated by the HSPICE Writer can be used for simulation as generated.

A default board description is included, and a default simulation is set up to measure rise and fall delays for both input and output simulations, which compensates for the double counting problem. However, Altera recommends that you customize the board description to more accurately represent your routing and termination scheme.

The sample board trace loading in the generated HSPICE model files must be replaced by your actual trace model before you can run a correct simulation. To do this, open the generated HSPICE model files for all pins you want to simulate and locate the following section.

Sample Board Trace Section

```
* I/O Board Trace and Termination Description
* - Replace this with your board trace and termination description
```

You must replace the example load with a load that matches the design of your PCB board. This includes a trace model, termination resistors, and, for output simulations, a receiver model. The spice circuit node that represents the pin of the FPGA package is called **pin**. The node that represents the far pin of the external device is called **load-in** (for output SPICE decks) and **source-in** (for input SPICE decks).

For an input simulation, you must also modify the stimulus portion of the spice file. The section of the file that must be modified is indicated in the following comment block.

Sample Source Stimulus Section

```
* Sample source stimulus placeholder
*  - Replace this with your I/O driver model
```

Replace the sample stimulus model with a model for the device that will drive the FPGA.

Running an HSPICE Simulation

Because simulation parameters are configured directly in the HSPICE model files, running a simulation requires only that you open an HSPICE file in the HSPICE user interface and start the simulation.

Figure 14-13: HSPICE User Interface Window
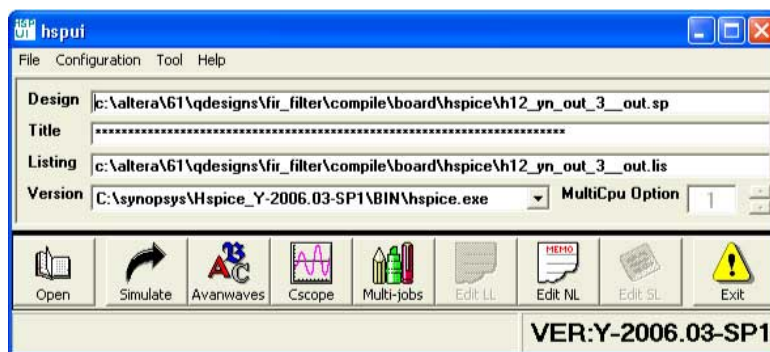
Click **Open** and browse to the location of the HSPICE model files generated by the Quartus Prime HSPICE Writer. The default location for HSPICE model files is *<project directory>*/**board/hspice**. Select the **.sp** file generated by the HSPICE Writer for the signal you want to simulate. Click **OK**.

To run the simulation, click **Simulate**. The status of the simulation is displayed in the window and saved in an **.lis** file with the same name as the **.sp** file when the simulation is complete. Check the **.lis** file if an error occurs during the simulation requiring a change in the **.sp** file to fix.

Interpreting the Results of an Output Simulation

By default, the automatically generated output simulation spice decks are set up to measure three delays for both rising and falling transitions. Two of the measurements, `tpd_rise` and `tpd_fall`, measure the double-counting corrected delay from the FPGA pin to the load pin. To determine the complete clock-edge to load-pin delay, add these numbers to the Quartus Prime software reported default loading t_{CO} delay.

The remaining four measurements, `tpd_uncomp_rise`, `tpd_uncomp_fall`, `t_dblcnt_rise`, and `t_dblcnt_fall`, are required for the double-counting compensation process and are not required for further timing usage.

Related Information

Interpreting the Results of an Input Simulation

By default, the automatically generated input simulation SPICE decks are set up to measure delays from the source's driver pin to the FPGA's input pin for both rising and falling transitions.

The propagation delay is reported by HSPICE measure statements as `tpd_rise` and `tpd_fall`. To determine the complete source driver pin-to-FPGA register delay, add these numbers to the Quartus Prime software reported T_H and T_{SU} input timing numbers.

Viewing and Interpreting Tabular Simulation Results

The **.lis** file stores the collected simulation data in tabular form. The default simulation configured by the HSPICE Writer produces delay measurements for rising and falling transitions on both input and output simulations.

These measurements are found in the **.lis** file and named `tpd_rise` and `tpd_fall`. For output simulations, these values are already adjusted for the double count. To determine the complete delay from the FPGA logic to the load pin, add either of these measurements to the Quartus Prime t_{CO} delay. For input simulations, add either of these measurements to the Quartus Prime t_{SU} and t_H delay values to calculate the complete delay from the far end stimulus to the FPGA logic. Other values found in the **.lis** file, such as `tpd_uncomp_rise`, `tpd_uncomp_fall`, `t_dblcnt_rise`, and `t_dblcnt_fall`, are parts of the double count compensation calculation. These values are not necessary for further analysis.

Viewing Graphical Simulation Results

You can view the results of the simulation quickly as a graphical waveform display using the AvanWaves viewer included with HSPICE. With the default simulation configured by the HSPICE Writer, you can view the simulated waveforms at both the source and destination in input and output simulations.

To see the waveforms for the simulation, in the HSPICE user interface window, click **AvanWaves**. The AvanWaves viewer opens and displays the **Results Browser**.

Figure 14-14: HSPICE AvanWaves Results Browser
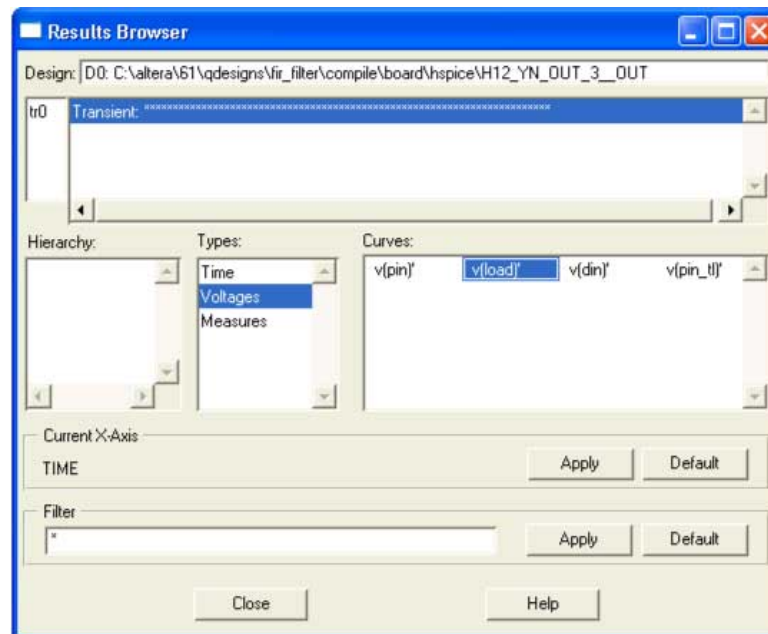
The **Results Browser** lets you select which waveform to view quickly in the main viewing window. If multiple simulations are run on the same signal, the list at the top of the **Results Browser** displays the results of each simulation. Click the simulation description to select which simulation to view. By default, the descriptions are derived from the first line of the HSPICE file, so the description might appear as a line of asterisks.

Select the type of waveform to view, by performing the following steps:

1. To see the source and destination waveforms with the default simulation, from the **Types** list, select **Voltages**.
2. On the **Curves** list, double-click the waveform you want to view. The waveform appears in the main viewing window.

You can zoom in and out and adjust the view as desired.

Figure 14-15: AvanWaves Waveform Viewer
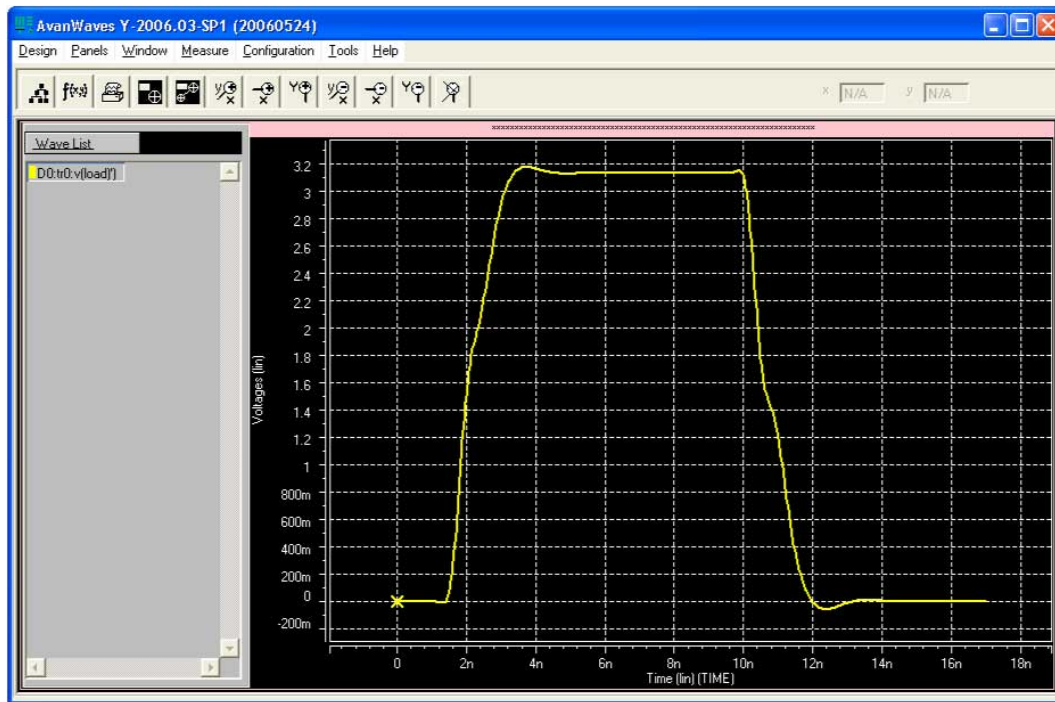
Making Design Adjustments Based on HSPICE Simulations

Based on the results of your simulations, you can make adjustments to the I/O assignments or simulation parameters if required. For example, after you run a simulation and see overshoot or ringing in the simulated signal at the destination buffer, you can adjust the drive strength I/O assignment setting to a lower value. Regenerate the HSPICE deck, and run the simulation again to verify that the change fixed the problem.

Figure 14-16: Example of Overshoot in the AvanWaves Waveform Viewer
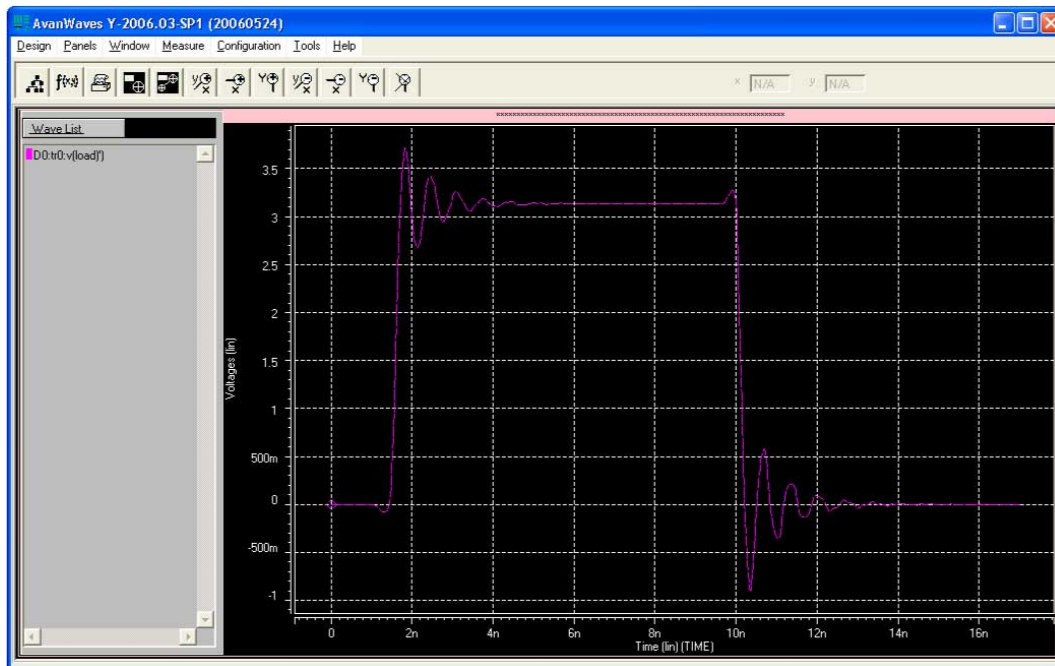
If there is a discontinuity or any other anomalies at the destination, adjust the board description in the Quartus Prime Board Trace Model, or in the generated HSPICE model files to change the termination scheme or adjust termination component values. After making these changes, regenerate the HSPICE files if necessary, and rerun the simulation to verify whether your adjustments solved the problem.

Figure 14-17: Example of Signal Integrity Anomaly in the AvanWaves Waveform Viewer
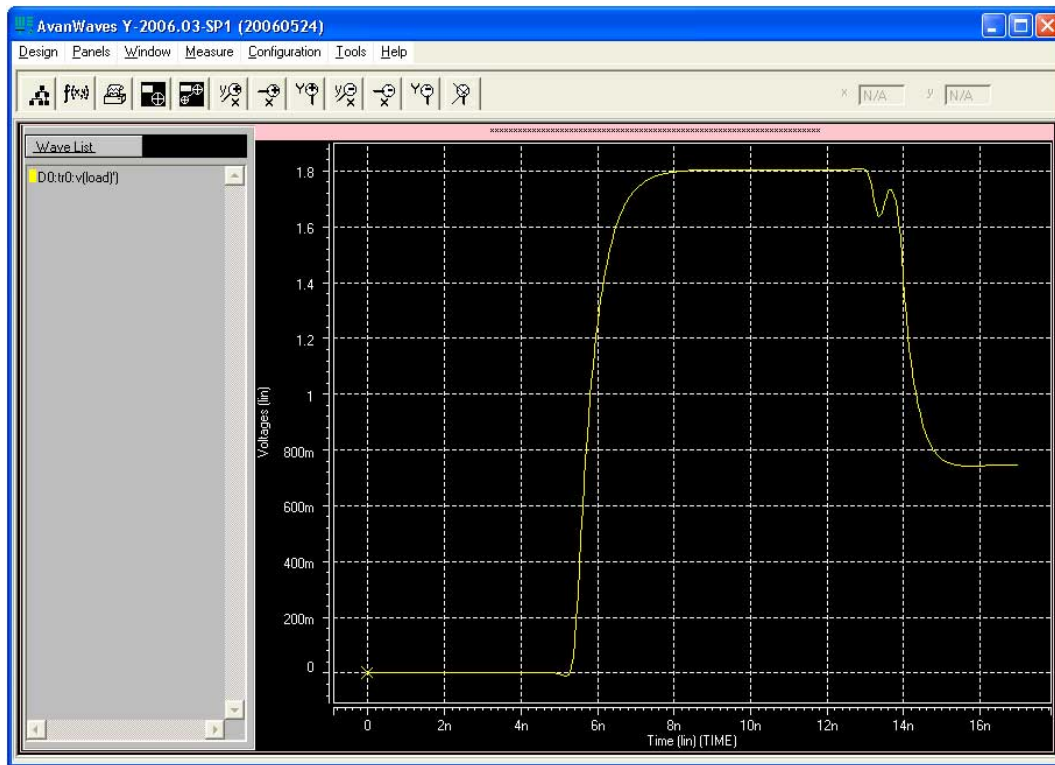
Related Information

Altera Signal Integrity Center
For more information about board-level signal integrity and to learn about ways to improve it with simple changes to your FPGA design.

Sample Input for I/O HSPICE Simulation Deck

The following sections examine a typical HSPICE simulation spice deck for an I/O of type input. Each section presents the simulation file one block at a time.

Header Comment

The first block of an input simulation spice deck is the header comment. The purpose of this block is to provide an easily readable summary of how the simulation file has been automatically configured by the Quartus Prime software.

This block has two main components: The first component summarizes the I/O configuration relevant information such as device, speed grade, and so on. The second component specifies the exact test condition that the Quartus Prime software assumes for the given I/O standard.

Sample Header Comment Block

```
* Quartus Prime HSPICE Writer I/O Simulation Deck*

* This spice simulation deck was automatically generated by
* Quartus for the following IO settings:
*
```

```
*   Device:        EP2S60F1020C3
*   Speed Grade:   C3
*   Pin:           AA4 (out96)
*   Bank:          IO Bank 6 (Row I/O)
*   I/O Standard:  LVTTL, 12mA
*   OCT:           Off
*
* Quartus Prime's default I/O timing delays assume the following slow
* corner simulation conditions.
*
*   Specified Test Conditions For Quartus Prime Tco
*       Temperature:       85C (Slowest Temperature Corner)
*       Transistor Model: TT (Typical Transistor Corner)
*       Vccn:              3.135V (Vccn_min = Nominal - 5%)
*       Vccpd:             2.97V (Vccpd_min = Nominal - 10%)
*       Load:              No Load
*       Vtt:               1.5675V (Voltage reference is Vccn/2)
*
* Note: The I/O transistors are specified to operate at least as
*       fast as the TT transistor corner, actual production
*       devices can be as fast as the FF corner. Any simulations
*       for hold times should be conducted using the fast process
*       corner with the following simulation conditions.
*           Temperature:       0C (Fastest Commercial Temperature Corner **)
*         Transistor Model: FF (Fastest Transistor Corner)
*         Vccn:              1.98V (Vccn_hold = Nominal + 10%)
*         Vccpd:             3.63V (Vccpd_hold = Nominal + 10%)
*         Vtt:               0.95V (Vtt_hold = Vccn/2 - 40mV)
*         Vcc:               1.25V (Vcc_hold = Maximum Recommended)
*         Package Model:     Short-circuit from pad to pin (no parasitics)
*
* Warnings:
```

Simulation Conditions

The simulation conditions block loads the appropriate process corner models for the transistors. This condition is automatically set up for the slow timing corner and is modified only if other simulation corners are desired.

Simulation Conditions Block

```
* Process Settings

.options brief
.inc 'sii_tt.inc' * TT process corner
```

Simulation Options

The simulation options block configures the simulation temperature and configures HSPICE with typical simulation options.

Simulation Options Block

```
* Simulation Options

.options brief=0
.options badchr co=132 scale=1e-6 acct ingold=2 nomod dv=1.0
+       dcstep=1 absv=1e-3 absi=1e-8 probe csdf=2 accurate=1
+       converge=1
.temp 85
```

Note: For a detailed description of these options, consult your *HSPICE* manual.

Constant Definition

The constant definition block of the simulation file instantiates the voltage sources that controls the configuration modes of the I/O buffer.

Constant Definition Block

```
* Constant Definition

voeb       oeb       0     vc  * Set to 0 to enable buffer output
vopdrain   opdrain   0     0   * Set to vc to enable open drain
vrambh     rambh     0     0   * Set to vc to enable bus hold
vrpullup   rpullup   0     0   * Set to vc to enable weak pullup
vpcdp5     rpcdp5    0     rp5 * Set the IO standard
vpcdp4     rpcdp4    0     rp4
vpcdp3     rpcdp3    0     rp3
vpcdp2     rpcdp2    0     rp2
vpcdp1     rpcdp1    0     rp1
vpcdp0     rpcdp0    0     rp0
vpcdn4     rpcdn4    0     rn4
vpcdn3     rpcdn3    0     rn3
vpcdn2     rpcdn2    0     rn2
vpcdn1     rpcdn1    0     rn1
vpcdn0     rpcdn0    0     rn0
vdin din     0       0
```

Where:

- Voltage source `voeb` controls the output enable of the buffer and is set to disabled for inputs.
- `vopdrain` controls the open drain mode for the I/O.
- `vrambh` controls the bus hold circuitry in the I/O.
- `vrpullup` controls the weak pullup.
- The next 11 voltages sources control the I/O standard of the buffer and are configured through a later library call.
- `vdin` is not used on input pins because it is the data pin for the output buffer.

Buffer Netlist

The buffer netlist block of the simulation spice deck loads all the load models required for the corresponding input pin.

Buffer Netlist Block

```
* IO Buffer Netlist

.include 'vio_buffer.inc'
```

Drive Strength

The drive strength block of the simulation SPICE deck loads the configuration bits necessary to configure the I/O into the proper I/O standard and drive strengths.

Although these settings are not relevant to an input buffer, they are provided to allow the SPICE deck to be modifiable to support bidirectional simulations.

Drive Strength Block

```
* Drive Strength Settings

.lib 'drive_select_hio.lib' 3p3ttl_12ma
```

I/O Buffer Instantiation

The I/O buffer instantiation block of the simulation SPICE deck instantiates the necessary power supplies and I/O model components that are necessary to simulate the given I/O.

I/O Buffer Instantiation

```
I/O Buffer Instantiation

* Supply Voltages Settings
.param vcn=3.135
.param vpd=2.97
.param vc=1.15

* Instantiate Power Supplies|
vvcc        vcc        0      vc      * FPGA core voltage
vvss        vss        0      0       * FPGA core ground
vvccn       vccn       0      vcn     * IO supply voltage
vvssn       vssn       0      0       * IO ground
vvccpd      vccpd      0      vpd     * Pre-drive supply voltage

* Instantiate I/O Buffer
xvio_buf din oeb opdrain die rambh
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp5 rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup vccn vccpd vcpad0 vio_buf

* Internal Loading on Pad
* - No loading on this pad due to differential buffer/support
*   circuitry

* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg
```

Board Trace and Termination

The board trace and termination block of the simulation SPICE deck is provided only as an example. Replace this block with your own board trace and termination models.

Board Trace and Termination Block

```
* I/O Board Trace and Termination Description
* - Replace this with your board trace and termination description

wtline pin vssn load vssn N=1 L=1 RLGCMODEL=tlinemodel
.MODEL tlinemodel W MODELTYPE=RLGC N=1 Lo=7.13n Co=2.85p
Rterm2 load vssn 1x
```

Stimulus Model

The stimulus model block of the simulation spice deck is provided only as a place holder example. Replace this block with your own stimulus model. Options for this include an IBIS or HSPICE model, among others.

Stimulus Model Block

```
* Sample source stimulus placeholder
* - Replace this with your I/O driver model

Vsource source 0 pulse(0 vcn 0s 0.4ns 0.4ns 8.5ns 17.4ns)
```

Simulation Analysis

The simulation analysis block of the simulation file is configured to measure the propagation delay from the source to the FPGA pin. Both the source and end point of the delay are referenced against the 50% V_{CCN} crossing point of the waveform.

Simulation Analysis Block

```
* Simulation Analysis Setup

* Print out the voltage waveform at both the source and the pin
.print tran v(source)  v(pin)
.tran 0.020ns 17ns

* Measure the propagation delay from the source pin to the pin
* referenced against the 50% voltage threshold crossing point

.measure TRAN tpd_rise TRIG v(source) val='vcn*0.5' rise=1
+ TARG v(pin) val ='vcn*0.5' rise=1
.measure TRAN tpd_fall TRIG v(source) val='vcn*0.5' fall=1
+ TARG v(pin) val ='vcn*0.5' fall=1
```

Sample Output for I/O HSPICE Simulation Deck

A typical HSPICE simulation SPICE deck for an I/O-type output has several sections. Each section presents the simulation file one block at a time.

Header Comment

The first block of an output simulation SPICE deck is the header comment. The purpose of this block is to provide a readable summary of how the simulation file has been automatically configured by the Quartus Prime software.

This block has two main components:

- The first component summarizes the I/O configuration relevant information such as device, speed grade, and so on.
- The second component specifies the exact test condition that the Quartus Prime software assumes when generating t_{CO} delay numbers. This information is used as part of the double-counting correction circuitry contained in the simulation file.

The SPICE decks are preconfigured to calculate the slow process corner delay but can also be used to simulate the fast process corner as well. The fast corner conditions are listed in the header under the notes section.

The final section of the header comment lists any warning messages that you must consider when you use the SPICE decks.

Header Comment Block

```
* Quartus Prime HSPICE Writer I/O Simulation Deck
*
* This spice simulation deck was automatically generated by
* Quartus Prime for the following IO settings:
*
*  Device:        EP2S60F1020C3
*  Speed Grade:   C3
*  Pin:           AA4 (out96)
*  Bank:          IO Bank 6 (Row I/O)
*    I/O Standard: LVTTL, 12mA
*  OCT:           Off
*
* Quartus' default I/O timing delays assume the following slow
* corner simulation conditions.
*   Specified Test Conditions For Quartus Prime Tco
*     Temperature:       85C (Slowest Temperature Corner)
*     Transistor Model: TT (Typical Transistor Corner)
*     Vccn:              3.135V (Vccn_min = Nominal - 5%)
*     Vccpd:             2.97V (Vccpd_min = Nominal - 10%)
*     Load:              No Load
*     Vtt:               1.5675V (Voltage reference is Vccn/2)
* For C3 devices, the TT transistor corner provides an
* approximation for worst case timing. However, for functionality
* simulations, it is recommended that the SS corner be simulated
* as well.
*
* Note: The I/O transistors are specified to operate at least as
*       fast as the TT transistor corner, actual production
*       devices can be as fast as the FF corner. Any simulations
*       for hold times should be conducted using the fast process
*       corner with the following simulation conditions.
*             Temperature:      0C (Fastest Commercial Temperature Corner **)
*             Transistor Model: FF (Fastest Transistor Corner)
*             Vccn:             1.98V (Vccn_hold = Nominal + 10%)
*             Vccpd:            3.63V (Vccpd_hold = Nominal + 10%)
*        Vtt:              0.95V (Vtt_hold = Vccn/2 - 40mV)
*        Vcc:              1.25V (Vcc_hold = Maximum Recommended)
*        Package Model:    Short-circuit from pad to pin
* Warnings:
```

Simulation Conditions

The simulation conditions block loads the appropriate process corner models for the transistors. This condition is automatically set up for the slow timing corner and must be modified only if other simulation corners are desired.

Simulation Conditions Block

```
* Process Settings

.options brief
.inc 'sii_tt.inc' * typical-typical process corner
```

Note: Two separate corners cannot be simulated at the same time. Instead, simulate the base case using the Quartus corner as one simulation and then perform a second simulation using the desired customer corner. The results of the two simulations can be manually added together.

Simulation Options

The simulation options block configures the simulation temperature and configures HSPICE with typical simulation options.

Simulation Options Block

```
* Simulation Options
.options brief=0
.options badchr co=132 scale=1e-6 acct ingold=2 nomod dv=1.0
+       dcstep=1 absv=1e-3 absi=1e-8 probe csdf=2 accurate=1
+       converge=1
.temp 85
```

Note: For a detailed description of these options, consult your *HSPICE* manual.

Constant Definition

The constant definition block of the output simulation SPICE deck instantiates the voltage sources that controls the configuration modes of the I/O buffer.

Constant Definition Block

```
* Constant Definition

voeb        oeb        0       0 * Set to 0 to enable buffer output
vopdrain    opdrain    0       0 * Set to vc to enable open drain
vrambh      rambh      0       0 * Set to vc to enable bus hold
vrpullup    rpullup    0       0 * Set to vc to enable weak pullup
vpci        rpci       0       0 * Set to vc to enable pci mode
vpcdp4      rpcdp4     0       rp4  * These control bits set the IO standard
vpcdp3      rpcdp3     0       rp3
vpcdp2      rpcdp2     0       rp2
vpcdp1      rpcdp1     0       rp1
vpcdp0      rpcdp0     0       rp0
vpcdn4      rpcdn4     0       rn4
vpcdn3      rpcdn3     0       rn3
vpcdn2      rpcdn2     0       rn2
vpcdn1      rpcdn1     0       rn1
vpcdn0      rpcdn0     0       rn0
vdin        din        0       pulse(0 vc 0s 0.2ns 0.2ns 8.5ns 17.4ns)
```

Where:

- Voltage source `voeb` controls the output enable of the buffer.
- `vopdrain` controls the open drain mode for the I/O.
- `vrambh` controls the bus hold circuitry in the I/O.
- `vrpullup` controls the weak pullup.
- `vpci` controls the PCI clamp.
- The next ten voltage sources control the I/O standard of the buffer and are configured through a later library call.
- `vdin` is connected to the data input of the I/O buffer.
- The edge rate of the input stimulus is automatically set to the correct value by the Quartus Prime software.

I/O Buffer Netlist

The I/O buffer netlist block loads all of the models required for the corresponding pin. These include a model for the I/O output buffer, as well as any loads that might be present on the pin.

I/O Buffer Netlist Block

```
*IO Buffer Netlist

.include 'hio_buffer.inc'
.include 'lvds_input_load.inc'
.include 'lvds_oct_load.inc'
```

Drive Strength

The drive strength block of the simulation spice deck loads the configuration bits for configuring the I/O to the proper I/O standard and drive strength. These options are set by the HSPICE Writer tool and are not changed for expected use.

Drive Strength Block

```
* Drive Strength Settings

.lib 'drive_select_hio.lib' 3p3ttl_12ma
```

Slew Rate and Delay Chain

Stratix and Cyclone devices have sections for configuring the slew rate and delay chain settings.

Slew Rate and Delay Chain Settings

```
* Programmable Output Delay Control Settings

.lib 'lib/output_delay_control.lib' no_delay

* Programmable Slew Rate Control Settings

.lib 'lib/slew_rate_control.lib' slow_slow
```

I/O Buffer Instantiation

The I/O buffer instantiation block of the output simulation spice deck instantiates the necessary power supplies and I/O model components that are necessary to simulate the given I/O.

I/O Buffer Instantiation Block

```
* I/O Buffer Instantiation

* Supply Voltages Settings
.param vcn=3.135
.param vpd=2.97
.param vc=1.15

* Instantiate Power Supplies
vvcc       vcc      0      vc     * FPGA core voltage
vvss       vss      0      0      * FPGA core ground
vvccn      vccn     0      vcn    * IO supply voltage
vvssn      vssn     0      0      * IO ground
vvccpd     vccpd    0      vpd    * Pre-drive supply voltage

* Instantiate I/O Buffer
xhio_buf din oeb opdrain die rambh
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup vccn vccpd vcpad0 hio_buf
```

```
* Internal Loading on Pad
* - This pad has an LVDS input buffer connected to it, along
*    with differential OCT circuitry. Both are disabled but
*    introduce loading on the pad that is modeled below.
xlvds_input_load die vss vccn lvds_input_load
xlvds_oct_load die vss vccpd vccn vcpad0 vccn lvds_oct_load

* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg
```

Board and Trace Termination

The board trace and termination block of the simulation SPICE deck is provided only as an example.
Replace this block with your specific board loading models.

Board Trace and Termination Block

```
* I/O Board Trace And Termination Description
* - Replace this with your board trace and termination description
wtline pin vssn load vssn N=1 L=1 RLGCMODEL=tlinemodel
.MODEL tlinemodel W MODELTYPE=RLGC N=1 Lo=7.13n Co=2.85p
Rterm2 load vssn 1x
```

Double-Counting Compensation Circuitry

The double-counting compensation circuitry block of the simulation SPICE deck instantiates a second
I/O buffer that is used to measure double-counting. The buffer is configured identically to the user I/O
buffer but is connected to the Quartus Prime software test load. The simulated delay of this second buffer
can be interpreted as the amount of double-counting between the Quartus Prime software and
HSPICE Writer simulated results.

As the amount of double-counting is constant for a given I/O standard on a given pin, consider separating
the double-counting circuitry from the simulation file. In doing so, you can perform any number of I/O
simulations while referencing the delay only once.

(Part of)Double-Counting Compensation Circuitry Block

```
* Double Counting Compensation Circuitry
*
* The following circuit is designed to calculate the amount of
* double counting between Quartus Prime and the HSPICE models. If
* you have not changed the default simulation temperature or
* transistor corner the double counting will be automatically
* compensated by this spice deck. In the event you wish to
* simulate an IO at a different temperature or transistor corner
* you will need to remove this section of code and manually
* account for double counting. A description of Altera's
* recommended procedure for this process can be found in the
* Quartus Prime HSPICE Writer AppNote.
*

* Supply Voltages Settings
.param vcn_tl=3.135
.param vpd_tl=2.97

* Test Load Constant Definition
vopdrain_tl   opdrain_tl   0      0
vrambh_tl     rambh_tl     0      0
```

```
vrpullup_tl    rpullup_tl    0    0

* Instantiate Power Supplies
vvccn_tl        vccn_tl       0      vcn_tl
vvssn_tl        vssn_tl       0      0
vvccpd_tl       vccpd_tl      0      vpd_tl

* Instantiate I/O Buffer
xhio_testload din oeb opdrain_tl die_tl rambh_tl
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup_tl vccn_tl vccpd_tl vcpad0_tl hio_buf

* Internal Loading on Pad
xlvds_input_testload die_tl vss vccn_tl lvds_input_load
xlvds_oct_testload die_tl vss vccpd_tl vccn_tl vcpad0_tl vccn_tl
lvds_oct_load

* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg

* Default Altera Test Load
* - 3.3V LVTTL default test condition is an open load
```

Related Information

[The Double Counting Problem in HSPICE Simulations](#) on page 14-17

Simulation Analysis

The simulation analysis block is set up to measure double-counting corrected delays. This is accomplished by measuring the uncompensated delay of the I/O buffer when connected to the user load, and when subtracting the simulated amount of double-counting from the test load I/O buffer.

Simulation Analysis Block

```
*Simulation Analysis Setup

* Print out the voltage waveform at both the pin and far end load
.print tran v(pin) v(load)
.tran 0.020ns 17ns

* Measure the propagation delay to the load pin. This value will
* include some double counting with Quartus Prime's Tco
.measure TRAN tpd_uncomp_rise TRIG v(din) val='vc*0.5' rise=1
+ TARG v(load) val='vcn*0.5' rise=1
.measure TRAN tpd_uncomp_fall TRIG v(din) val='vc*0.5' fall=1
+ TARG v(load) val='vcn*0.5' fall=1

* The test load buffer can calculate the amount of double counting
.measure TRAN t_dblcnt_rise TRIG v(din) val='vc*0.5' rise=1
+ TARG v(pin_tl) val='vcn_tl*0.5' rise=1
.measure TRAN t_dblcnt_fall TRIG v(din) val='vc*0.5' fall=1
+ TARG v(pin_tl) val='vcn_tl*0.5' fall=1

* Calculate the true propagation delay by subtraction
.measure TRAN tpd_rise PARAM='tpd_uncomp_rise-t_dblcnt_rise'
.measure TRAN tpd_fall PARAM='tpd_uncomp_fall-t_dblcnt_fall'
```

Advanced Topics

The information in this section describes some of the more advanced topics and methods employed when setting up and running HSPICE simulation files.

PVT Simulations

The automatically generated HSPICE simulation files are set up to simulate the slow process corner using low voltage, high temperature, and slow transistors. To ensure a fully robust link, Altera recommends that you run simulations over all process corners.

To perform process, voltage, and temperature (PVT) simulations, manually modify the spice decks in a two step process:

1. Remove the double-counting compensation circuitry from the simulation file. This is required as the amount of double-counting is dependant upon how the Quartus Prime software calculates delays and is not based on which PVT corner is being simulated. By default, the Quartus Prime software provides timing numbers using the slow process corner.
2. Select the proper corner for the PVT simulation by setting the correct HSPICE temperature, changing the supply voltage sources, and loading the correct transistor models.

A more detailed description of HSPICE process corners can be found in the family-specific HSPICE model documentation.

Related Information

Accessing HSPICE Simulation Kits on page 14-16

Hold Time Analysis

Altera recommends performing worst-case hold time analysis using the fast corner models, which use fast transistors, high voltage, and low temperature. This involves modifying the SPICE decks to select the correct temperature option, change the supply voltage sources, and load the correct fast transistor models. The values of these parameters are located in the header comment section of the corresponding simulation deck files.

For a truly worst-case analysis, combine the HSPICE Writer hold time analysis results with the Quartus Prime software fast timing model. This requires that you change the double-counting compensation circuitry in the simulations files to also simulate the fast process corners, as this is what the Quartus Prime software uses for the fast timing model.

Note: This method of hold time analysis is recommended only for globally synchronous buses. Do not apply this method of hold-time analysis to source synchronous buses. This is because the source synchronous clocking scheme is designed to cancel out some of the PVT timing effects. If this is not taken into account, the timing results will not be accurate. Proper source synchronous timing analysis is beyond the scope of this document.

I/O Voltage Variations

Use each of the FPGA family datasheets to verify the recommended operating conditions for supply voltages. For current FPGA families, the maximum recommended voltage corresponds to the fast corner, while the minimum recommended voltage corresponds to the slow corner. These voltage recommendations are specified at the power pins of the FPGA and are not necessarily the same voltage that are seen by the I/O buffers due to package IR drops.

The automatically generated HSPICE simulation files model this IR effect pessimistically by including a 50-mV IR drop on the V_{CCPD} supply when a high drive strength standard is being used.

Correlation Report

Correlation reports for the HSPICE I/O models are located in the family-specific HSPICE I/O buffer simulation kits.

Related Information

Accessing HSPICE Simulation Kits on page 14-16

Document Revision History

Table 14-2: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*.
June 2014	14.0.0	Updated format.
December 2010	10.0.1	Template update.
July 2010	10.0.0	Updated device support.
November 2009	9.1.0	No change to content.
March 2009	9.0.0	• Was volume 3, chapter 12 in the 8.1.0 release. • No change to content.
November 2008	8.1.0	• Changed to 8-1/2 x 11 page size. • Added information for Stratix III devices. • Input signals for Cyclone III devices are supported.

Date	Version	Changes
May 2008	8.0.0	- Updated "Introduction" on page 12–1. - Updated Figure 12–1. - Updated Figure 12–3. - Updated Figure 12–13. - Updated "Output File Generation" on page 12–6. - Updated "Simulation with HSPICE Models" on page 12–17. - Updated "Invoking HSPICE Writer from the Command Line" on page 12–22. - Added "Sample Input for I/O HSPICE Simulation Deck" on page 12–29. - Added "Sample Output for I/O HSPICE Simulation Deck" on page 12–33. - Updated "Correlation Report" on page 12–41. - Added hyperlinks to referenced documents and websites throughout the chapter. - Made minor editorial updates.

Related Information

Quartus Handbook Archive

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

Cadence PCB Design Tools Support

The Quartus Prime software interacts with the following software to provide a complete FPGA-to-board integration design workflow: the Cadence Allegro Design Entry HDL software and the Cadence Allegro Design Entry CIS (Component Information System) software (also known as OrCAD Capture CIS). The information is useful for board design and layout engineers who want to begin the FPGA board integration process while the FPGA is still in the design phase. Part librarians can also benefit by learning the method to use output from the Quartus Prime software to create new library parts and symbols.

With today's large, high-pin-count and high-speed FPGA devices, good PCB design practices are important to ensure the correct operation of your system. The PCB design takes place concurrently with the design and programming of the FPGA. An FPGA or ASIC designer initially creates the signal and pin assignments and the board designer must transfer these assignments to the symbols used in their system circuit schematics and board layout correctly. As the board design progresses, you must perform pin reassignments to optimize the layout. You must communicate pin reassignments to the FPGA designer to ensure the new assignments are processed through the FPGA with updated placement and routing.

You require the following software:

- The Quartus Prime software version 5.1 or later
- The Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software version 15.2 or later
- The OrCAD Capture software with the optional CIS option version 10.3 or later (optional)

Note: These programs are very similar because the Cadence Allegro Design Entry CIS software is based on the OrCAD Capture software. Any procedural information can also apply to the OrCAD Capture software unless otherwise noted.

Related Information

- **www.cadence.com**
  For more information about obtaining and licensing the Cadence tools and for product information, support, and training
- **www.cadence.com**
  For more information about the OrCAD Capture software and the CIS option
- **www.ema-eda.com**
  For more information about Cadence and OrCAD support and training

---

**ISO
9001:2008
Registered**

ALTERA®

Product Comparison

Table 15-1: Cadence and OrCAD Product Comparison

Description	Cadence Allegro Design Entry HDL	Cadence Allegro Design Entry CIS	OrCAD Capture CIS
Former Name	Concept HDL Expert	Capture CIS Studio	—
History	More commonly known by its former name, Cadence renamed all board design tools in 2004 under the Allegro name.	Based directly on OrCAD Capture CIS, the Cadence Allegro Design Entry CIS software is still developed by OrCAD but sold and marketed by Cadence. EMA provides support and training.	The basis for Design Entry CIS is still developed by OrCAD for continued use by existing OrCAD customers. EMA provides support and training for all OrCAD products.
Vendor Design Flow	Cadence Allegro 600 series, formerly known as the Expert Series, for high-end, high-speed design.	Cadence Allegro 200 series, formerly known as the Studio Series, for small- to medium-level design.	—

Related Information

- www.cadence.com
- www.ema-eda.com

FPGA-to-PCB Design Flow

You can create a design flow integrating an Altera FPGA design from the Quartus Prime software through a circuit schematic in the Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software.

Figure 15-1: Design Flow with the Cadence Allegro Design Entry HDL Software

```
                Start FPGA Design                    Start PCB Design
                                                  (Allegro Design Entry HDL)

                    Quartus Prime Software      Project Manager

                     Create or Change              Create or Open a Project
                     Pin Assignments
                                                Part Developer

              Run I/O Assignment    Run Full        Import or Update Pin
                  Analysis         Compilation         Assignments

                                                  Create or Update FPGA Symbol

                                                   Edit or Fracture Symbol
                         .pin
                                               Design Entry HDL

                                                 Instantiate Symbol in Schematic

                                                 Forward to Board Layout Tool

                                               Board Layout Tool

                                                   Layout and Route FPGA

                                                        End
```

Figure 15-2: Design Flow with the Cadence Allegro Design Entry CIS Software

```
                Start FPGA Design                    Start PCB Design
                                                  (Allegro Design Entry CIS)

                    Quartus Prime Software      Design Entry CIS

                     Create or Change              Create or Open Project
                     Pin Assignments
                                                  Generate or Update Schematic

              Run I/O Assignment    Run Full        Edit or Fracture Symbol
                  Analysis         Compilation
                                                  Instantiate Symbol in Schematic

                                                  Forward to Board Developement Tool
                         .pin
                                               Board Layout Tool

                                                   Layout and Route FPGA

                                                        End
```
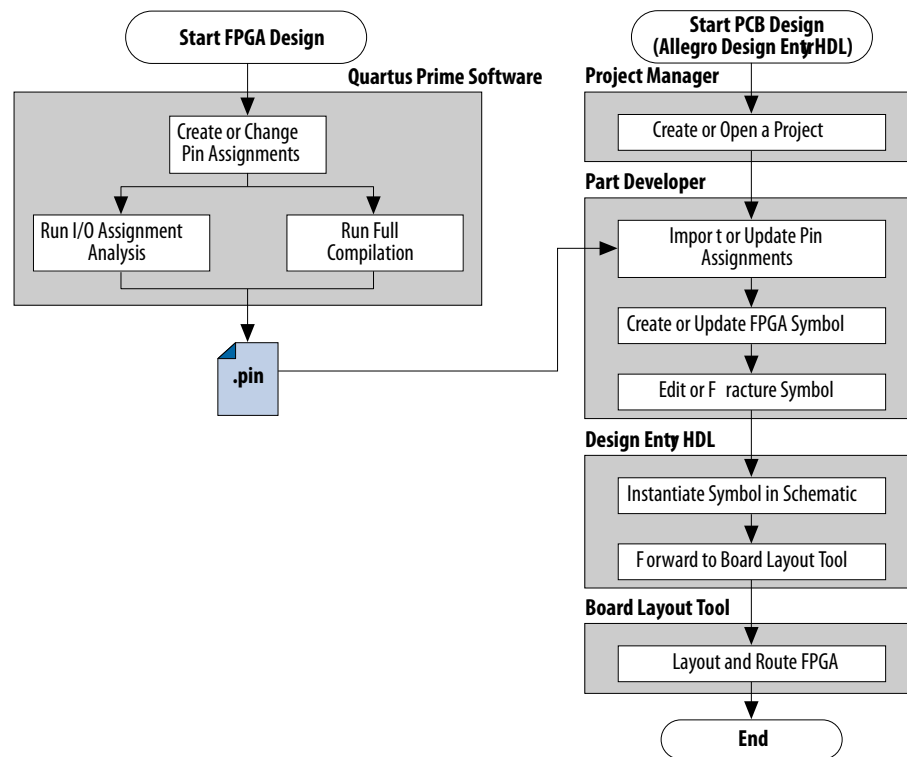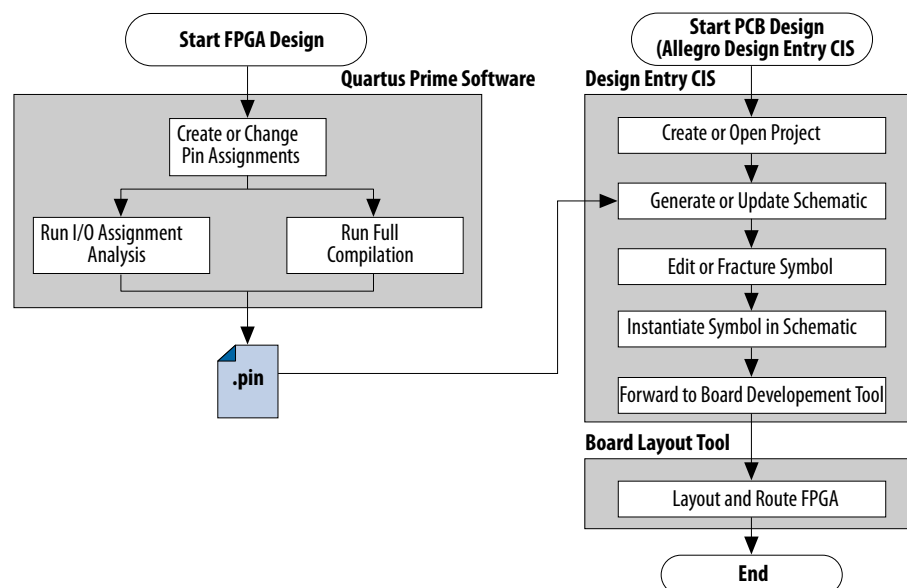
To create FPGA symbols using the Cadence Allegro PCB Librarian Part Developer tool, you must obtain the Cadence PCB Librarian Expert license. You can update symbols with changes made to the FPGA design using any of these tools.

Integrating Altera FPGA Design

To integrate an Altera FPGA design starting in the Quartus Prime software through to a circuit schematic in the Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software, follow these steps:

1. In the Quartus Prime software, compile your design to generate a Pin-Out File (**.pin**) to transfer the assignments to the Cadence software.

2. If you are using the Cadence Allegro Design Entry HDL software for your schematic design, follow these steps:

   a. Open an existing project or create a new project in the Cadence Allegro Project Manager tool.

   b. Construct a new symbol or update an existing symbol using the Cadence Allegro PCB Librarian Part Developer tool.

   c. With the Cadence Allegro PCB Librarian Part Developer tool, edit your symbol or fracture it into smaller parts (optional).

   d. Instantiate the symbol in your Cadence Allegro Design Entry HDL software schematic and transfer the design to your board layout tool.

      or

      If you are using the Cadence Allegro Design Entry CIS software for your schematic design, follow these steps:

   e. Generate a new part in a new or existing Cadence Allegro Design Entry CIS project, referencing the **.pin** output file from the Quartus Prime software. You can also update an existing symbol with a new **.pin**.

   f. Split the symbol into smaller parts as necessary.

   g. Instantiate the symbol in your Cadence Allegro Design Entry CIS schematic and transfer the design to your board layout tool.

Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA

With the Quartus Prime software, you can extract pin assignment data and perform SSN analysis of your FPGA design for designs targeting the Stratix III device family.

You can analyze SSN in your device early in the board layout stage as part of your overall pin planning process; however, you do not have to perform SSN analysis to generate pin assignment data from the Quartus Prime software. You can use the SSN Analyzer tool to optimize the pin assignments for better SSN performance of your device.

Related Information

- **Simultaneous Switching Noise (SSN) Analysis and Optimizations**

Setting Up the Quartus Prime Software

You can transfer pin and signal assignments from the Quartus Prime software to the Cadence design tools by generating the Quartus Prime project **.pin**. The **.pin** is an output file generated by the Quartus Prime Fitter containing pin assignment information. You can use the Quartus Prime Pin Planner to set and change the assignments in the **.pin** and then transfer the assignments to the Cadence design tools. You

cannot, however, import pin assignment changes from the Cadence design tools into the Quartus Prime software with the **.pin**.

The **.pin** lists all used and unused pins on your selected Altera device. The **.pin** also provides the following basic information fields for each assigned pin on the device:

- Pin signal name and usage
- Pin number
- Signal direction
- I/O standard
- Voltage
- I/O bank
- User or Fitter-assigned

Related Information

- **I/O Management** on page 2-1
  For more information about using the Quartus Prime Pin Planner to create or change pin assignment details.

Generating a .pin File

To generate a **.pin**, follow these steps:

1. Compile your design.
2. Locate the **.pin** in your Quartus Prime project directory with the name <project name>**.pin**.

Related Information

- **I/O Management** on page 2-1
  For more information about pin and signal assignment transfer and the files that the Quartus Prime software can import and export.

FPGA-to-Board Integration with the Cadence Allegro Design Entry HDL Software

The Cadence Allegro Design Entry HDL software is a schematic capture tool and is part of the Cadence 600 series design flow. Use the Cadence Allegro Design Entry HDL software to create flat circuit schematics for all types of PCB design. The Cadence Allegro Design Entry HDL software can also create hierarchical schematics to facilitate design reuse and team-based design. With the Cadence Allegro Design Entry HDL software, the design flow from FPGA-to-board is one-way, using only the **.pin** generated by the Quartus Prime software. You can only make signal and pin assignment changes in the Quartus Prime software and these changes reflect as updated symbols in a Cadence Allegro Design Entry HDL project.

For more information about the design flow with the Cadence Allegro Design Entry HDL software, refer to **Figure 15-1**.

Note: Routing or pin assignment changes made in a board layout tool or a Cadence Allegro Design Entry HDL software symbol cannot be back-annotated to the Quartus Prime software.

Creating Symbols

In addition to circuit simulation, circuit board schematic creation is one of the first tasks required when
designing a new PCB. Schematics must understand how the PCB works, and to generate a netlist for a
board layout tool for board design and routing. The Cadence Allegro PCB Librarian Part Developer tool
allows you to create schematic symbols based on FPGA designs exported from the Quartus Prime
software.

You can create symbols for the Cadence Allegro Design Entry HDL project with the Cadence Allegro PCB
Librarian Part Developer tool, which is available in the Cadence Allegro Project Manager tool. Altera
recommends using the Cadence Allegro PCB Librarian Part Developer tool to import FPGA designs into
the Cadence Allegro Design Entry HDL software.

You must obtain a PCB Librarian Expert license from Cadence to run the Cadence Allegro PCB Librarian
Part Developer tool. The Cadence Allegro PCB Librarian Part Developer tool provides a GUI with many
options for creating, editing, fracturing, and updating symbols. If you do not use the Cadence Allegro
PCB Librarian Part Developer tool, you must create and edit symbols manually in the Symbol Schematic
View in the Cadence Allegro Design Entry HDL software.

Note: If you do not have a PCB Librarian Expert license, you can automatically create FPGA symbols
using the programmable IC (PIC) design flow found in the Cadence Allegro Project Manager tool.

Before creating a symbol from an FPGA design, you must open a Cadence Allegro Design Entry HDL
project with the Cadence Allegro Project Manager tool. If you do not have an existing Cadence Allegro
Design Entry HDL project, you can create one with the Cadence Allegro Design Entry HDL software. The
Cadence Allegro Design Entry HDL project directory with the name <project name>**.cpm** contains your
Cadence Allegro Design Entry HDL projects.

While the Cadence Allegro PCB Librarian Part Developer tool refers to symbol fractures as slots, the other
tools use different names to refer to symbol fractures.

Table 15-2: Symbol Fracture Naming Conventions

	Cadence Allegro PCB Librarian Part Developer Tool	Cadence Allegro Design Entry HDL Software	Cadence Allegro Design Entry CIS Software
During symbol generation	Slots	—	Sections
During symbol schematic instantiation	—	Versions	Parts

Cadence Allegro PCB Librarian Part Developer Tool

You can create, fracture, and edit schematic symbols for your designs using the Cadence Allegro PCB Librarian Part Developer tool. Symbols designed in the Cadence Allegro PCB Librarian Part Developer tool can be split or fractured into several functional blocks called slots, allowing multiple smaller part fractures to exist on the same schematic page or across multiple pages.

Cadence Allegro PCB Librarian Part Developer Tool in the Design Flow
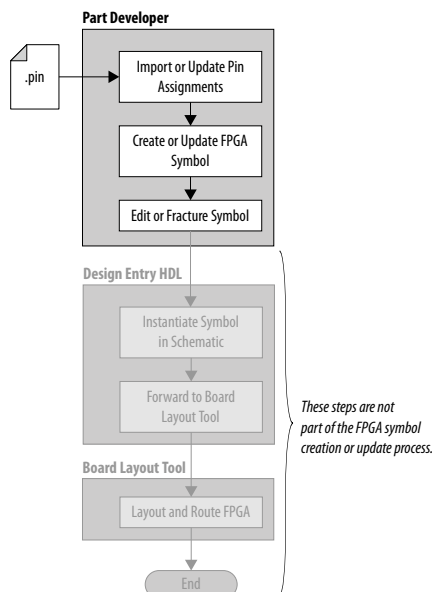
To run the Cadence Allegro PCB Librarian Part Developer tool, you must open a Cadence Allegro Design Entry HDL project in the Cadence Allegro Project Manager tool. To open the Cadence Allegro PCB Librarian Part Developer tool, on the Flows menu, click **Library Management**, and then click **Part Developer**.

Related Information

- **FPGA-to-PCB Design Flow** on page 15-2

Import and Export Wizard

After starting the Cadence Allegro PCB Librarian Part Developer tool, use the **Import and Export**wizard to import your pin assignments from the Quartus Prime software.

Note: Altera recommends using your PCB Librarian Expert license file. To point to your PCB Librarian Expert license file, on the File menu, click **Change Product** and then select the correct product license.

To access the Import and Export wizard, follow these steps:

1. On the File menu, click **Import and Export**.
2. Select **Import ECO-FPGA**, and then click **Next**.
3. In the **Select Source** page of the **Import and Export** wizard, specify the following settings:

    a. In the **Vendor** list, select **Altera**.

    b. In the **PnR Tool** list, select **quartusII**.

    c. In the **PR File** box, browse to select the **.pin** in your Quartus Prime project directory.

    d. Click **Simulation Options** to select simulation input files.

    e. Click **Next.**

4. In the **Select Destination** dialog box, specify the following settings:

    a. Under **Select Component**, click **Generate Custom Component** to create a new component in a library,

    or

    Click **Use standard component** to base your symbol on an existing component.

    Note: Altera recommends creating a new component if you previously created a generic component for an FPGA device. Generic components can cause some problems with your design. When you create a new component, you can place your pin and signal assignments from the Quartus Prime software on this component and reuse the component as a base when you have a new FPGA design.

    b. In the **Library** list, select an existing library. You can select from the cells in the selected library. Each cell represents all the symbol versions and part fractures for a particular part. In the **Cell** list, select the existing cell to use as a base for your part.

    c. In the **Destination Library** list, select a destination library for the component. Click **Next**.

    d. Review and edit the assignments you import into the Cadence Allegro PCB Librarian Part Developer tool based on the data in the **.pin** and then click **Finish**. The location of each pin is not included in the **Preview of Import Data** page of the **Import and Export** wizard, but input pins are on the left side of the created symbol, output pins on the right, power pins on the top, and ground pins on the bottom.

Editing and Fracturing Symbol

After creating your new symbol in the Cadence Allegro PCB Librarian Part Developer tool, you can edit the symbol graphics, fracture the symbol into multiple slots, and add or change package or symbol properties.

The Part Developer Symbol Editor contains many graphical tools to edit the graphics of a particular symbol. To edit the symbol graphics, select the symbol in the cell hierarchy. The **Symbol Pins** tab appears. You can edit the preview graphic of the symbol in the **Symbol Pins** tab.

Fracturing a Cadence Allegro PCB Librarian Part Developer package into separate symbol slots is useful for FPGA designs. A single symbol for most FPGA packages might be too large for a single schematic page. Splitting the part into separate slots allows you to organize parts of the symbol by function, creating cleaner circuit schematics. For example, you can create one slot for an I/O symbol, a second slot for a JTAG symbol, and a third slot for a power/ground symbol.

Figure 15-3: Splitting a Symbol into Multiple Slots
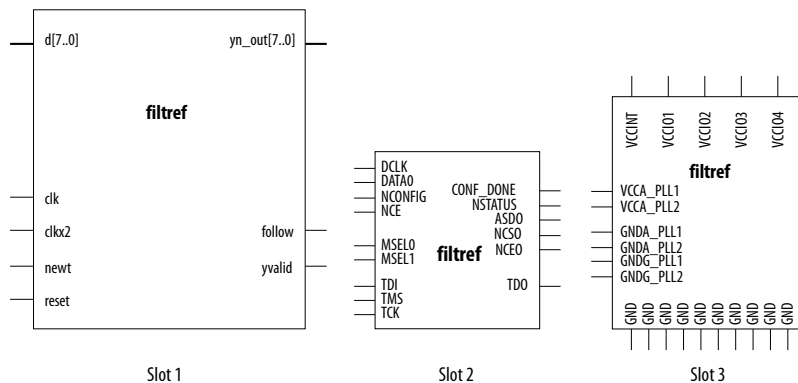
- This diagram represents a Cyclone device with JTAG or passive serial (PS) mode configuration option settings. Symbols created for other devices or other configuration modes may have different sets of configuration pins, but can be fractured in a similar manner.
- The power/ground slot shows only a representation of power and ground pins because the device contains a large number of power and ground pins.

To fracture a part into separate slots, or to modify the slot locations of pins on parts fractured in the Cadence Allegro PCB Librarian Part Developer tool, follow these steps:

1. Start the Cadence Allegro Design Project Manager.
2. On the Flows menu, click **Library Management**.
3. Click **Part Developer**.
4. Click the name of the package you want to change in the cell hierarchy.
5. Click **Functions/Slots**. If you are not creating new slots but want to change the slot location of some pins, proceed to Step **6**. If you are creating new slots, click **Add**. A dialog box appears, allowing you to add extra symbol slots. Set the number of extra slots you want to add to the existing symbol, not the total number of desired slots for the part. Click **OK**.
6. Click **Distribute Pins**. Specify the slot location for each pin. Use the checkboxes in each column to move pins from one slot to another. Click **OK**.
7. After distributing the pins, click the **Package Pin** tab and click **Generate Symbol(s)**.
8. Select whether to create a new symbol or modify an existing symbol in each slot. Click **OK**.

The newly generated or modified slot symbols appear as separate symbols in the cell hierarchy. Each of these symbols can be edited individually.

Caution: The Cadence Allegro PCB Librarian Part Developer tool allows you to remap pin assignments in the **Package Pin** tab of the main Cadence Allegro PCB Librarian Part Developer window. If signals remap to different pins in the Cadence Allegro PCB Librarian Part Developer tool, the changes reflect only in regenerated symbols for use in your schematics. You cannot transfer pin assignment changes to the Quartus Prime software from the Cadence Allegro PCB Librarian Part Developer tool, which creates a potential mismatch of the schematic symbols and assignments in the FPGA design. If pin assignment changes are necessary, make the changes in the Quartus Prime Pin Planner instead of the Cadence Allegro PCB Librarian Part Developer tool, and update the symbol as described in the following sections.

For more information about creating, editing, and organizing component symbols with the Cadence Allegro PCB Librarian Part Developer tool, refer to the Part Developer Help.

Updating FPGA Symbols

As the design process continues, you must make logic changes in the Quartus Prime software, placing signals on different pins after recompiling the design, or use the Quartus Prime Pin Planner to make changes manually. The board designer can request such changes to improve the board routing and layout. To ensure signals connect to the correct pins on the FPGA, you must carry forward these types of changes to the circuit schematic and board layout tools. Updating the **.pin** in the Quartus Prime software facilitates this flow.

Figure 15-4: Updating the FPGA Symbol in the Design Flow
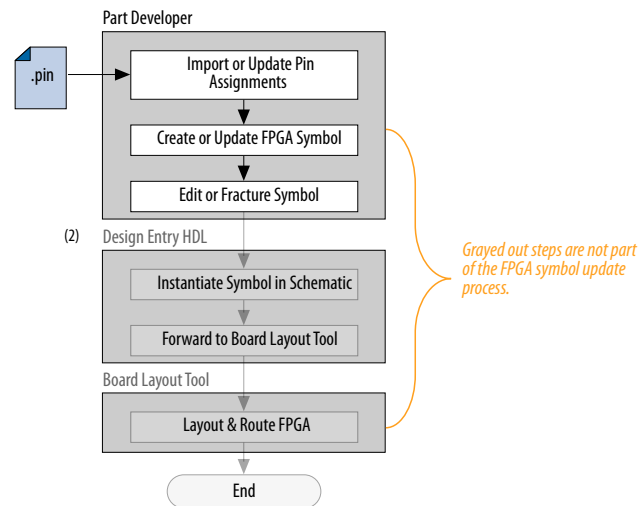
To update the symbol using the Cadence Allegro PCB Librarian Part Developer tool after updating the **.pin**, follow these steps:

1. On the File menu, click **Import and Export**. The Import and Export wizard appears.
2. In the list of actions to perform, select **Import ECO - FPGA**. Click **Next**. The **Select Source** dialog box appears.
3. Select the updated source of the FPGA assignment information. In the **Vendor** list, select **Altera**. In the **PnR Tool** list, select **quartusII**. In the **PR File** field, click **browse** to specify the updated **.pin** in your Quartus Prime project directory. Click **Next**. The Select Destination window appears.
4. Select the source component and a destination cell for the updated symbol. To create a new component based on the updated pin assignment data, select **Generate Custom Component**. Selecting **Generate Custom Component** replaces the cell listed under the **Specify Library and Cell** name header with a new, nonfractured cell. You can preserve these edits by selecting **Use standard component and select the existing library and cell**. Select the destination library for the component and click **Next**. The **Preview of Import Data** dialog box appears.
5. Make any additional changes to your symbol. Click **Next**. A list of ECO messages appears summarizing the changes made to the cell. To accept the changes and update the cell, click **Finish**.
6. The main Cadence Allegro PCB Librarian Part Developer window appears. You can edit, fracture, and generate the updated symbols as usual from the main Cadence Allegro PCB Librarian Part Developer window.

Note: If the Cadence Allegro PCB Librarian Part Developer tool is not set up to point to your PCB Librarian Expert license file, an error message appears in red at the bottom of the message text

QPP5V2
2015.11.02

Instantiating the Symbol in the Cadence Allegro Design Entry HDL...

15-11

window of the Part Developer when you select the **Import and Export** command. To point to your PCB Librarian Expert license, on the File menu, click **Change Product**, and select the correct product license.

Related Information

- **FPGA-to-PCB Design Flow** on page 15-2

Instantiating the Symbol in the Cadence Allegro Design Entry HDL Software

To instantiate the symbol in your Cadence Allegro Design Entry HDL schematic after saving the new symbol in the Cadence Allegro PCB Librarian Part Developer tool, follow these steps:

1. In the Cadence Allegro Project Manager tool, switch to the board design flow.
2. On the Flows menu, click **Board Design**.
3. To start the Cadence Allegro Design Entry HDL software, click **Design Entry**.
4. To add the newly created symbol to your schematic, on the Component menu, click **Add**. The **Add Component** dialog box appears.
5. Select the new symbol library location, and select the name of the cell you created from the list of cells.

   The symbol attaches to your cursor for placement in the schematic. To fracture the symbol into slots, right-click the symbol and choose **Version** to select one of the slots for placement in the schematic.
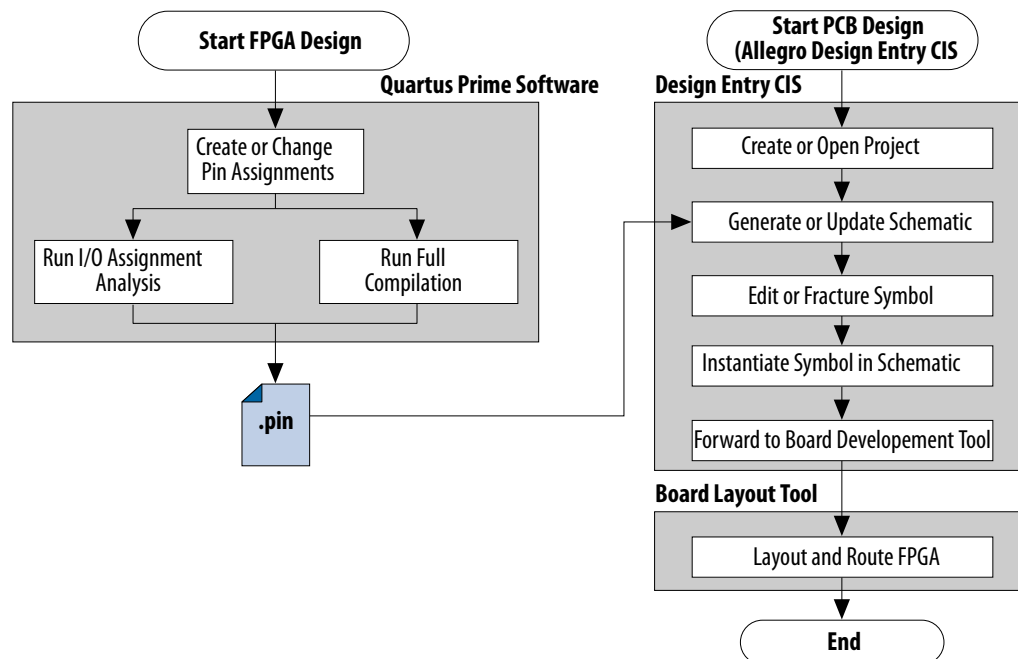
Related Information

- **www.cadence.com**
  Provides more information about the Cadence Allegro Design Entry HDL software, including licensing, support, usage, training, and product updates.

FPGA-to-Board Integration with Cadence Allegro Design Entry CIS Software

The Cadence Allegro Design Entry CIS software is a schematic capture tool (part of the Cadence 200 series design flow based on OrCAD Capture CIS). Use the Cadence Allegro Design Entry CIS software to create flat circuit schematics for all types of PCB design. You can also create hierarchical schematics to facilitate design reuse and team-based design using the Cadence Allegro Design Entry CIS software. With the Cadence Allegro Design Entry CIS software, the design flow from FPGA-to-board is unidirectional using only the **.pin** generated by the Quartus Prime software. You can only make signal and pin assignment changes in the Quartus Prime software. These changes reflect as updated symbols in a Cadence Allegro Design Entry CIS schematic project.

Figure 15-5: Design Flow with the Cadence Allegro Design Entry CIS Software

Note: Routing or pin assignment changes made in a board layout tool or a Cadence Allegro Design Entry CIS symbol cannot be back-annotated to the Quartus Prime software.

Related Information

- **www.cadence.com**
  For more information about the Cadence Allegro Design Entry CIS software, including licensing, support, usage, training, and product updates.
- **www.ema-eda.com**
  For more information about the Cadence Allegro Design Entry CIS software, including licensing, support, usage, training, and product updates.

Creating a Cadence Allegro Design Entry CIS Project

The Cadence Allegro Design Entry CIS software has built-in support for creating schematic symbols using pin assignment information imported from the Quartus Prime software.

To create a new project in the Cadence Allegro Design Entry CIS software, follow these steps:

1. On the File menu, point to **New** and click **Project.** The New Project wizard starts.

   When you create a new project, you can select the PC Board wizard, the Programmable Logic wizard, or a blank schematic.

2. Select the PC Board wizard to create a project where you can select which part libraries to use, or select a blank schematic.

The Programmable Logic wizard only builds an FPGA logic design in the Cadence Allegro Design Entry CIS software.

Your new project is in the specified location and consists of the following files:

- OrCAD Capture Project File (**.opj**)
- Schematic Design File (**.dsn**)

Generating a Part

After you create a new project or open an existing project in the Cadence Allegro Design Entry CIS software, you can generate a new schematic symbol based on your Quartus Prime FPGA design. You can also update an existing symbol. The Cadence Allegro Design Entry CIS software stores component symbols in OrCAD Library File (**.olb**). When you place a symbol in a library attached to a project, it is immediately available for instantiation in the project schematic.

You can add symbols to an existing library or you can create a new library specifically for the symbols generated from your FPGA designs. To create a new library, follow these steps:

1. On the File menu, point to **New** and click **Library** in the Cadence Allegro Design Entry CIS software to create a default library named **library1.olb**. This library appears in the **Library** folder in the Project Manager window of the Cadence Allegro Design Entry CIS software.
2. To specify a desired name and location for the library, right-click the new library and select **Save As**. Saving the new library creates the library file.

Generating Schematic Symbol

You can now create a new symbol to represent your FPGA design in your schematic.

To generate a schematic symbol, follow these steps:

1. Start the Cadence Allegro Design Entry CIS software.
2. On the Tools menu, click **Generate Part**. The **Generate Part** dialog box appears.
3. To specify the **.pin** from your Quartus Prime design, in the **Netlist/source file type** field, click **Browse**.
4. In the **Netlist/source file type** list, select **Altera Pin File**
5. Type the new part name.
6. Specify the **Destination part library** for the symbol. Failing to select an existing library for the part creates a new library with a default name that matches the name of your Cadence Allegro Design Entry CIS project.
7. To create a new symbol for this design, select **Create new part**. If you updated your **.pin** in the Quartus Prime software and want to transfer any assignment changes to an existing symbol, select **Update pins on existing part in library**.
8. Select any other desired options and set **Implementation type** to **<none>**. The symbol is for a primitive library part based only on the **.pin** and does not require special implementation. Click **OK**.
9. Review the Undo warning and click **Yes** to complete the symbol generation.

You can locate the generated symbol in the selected library or in a new library found in the **Outputs** folder of the design in the Project Manager window. Double-click the name of the new symbol to see its graphical representation and edit it manually using the tools available in the Cadence Allegro Design Entry CIS software.

Note: For more information about creating and editing symbols in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

Splitting a Part

After saving a new symbol in a project library, you can fracture the symbol into multiple parts called sections. Fracturing a part into separate sections is useful for FPGA designs. A single symbol for most FPGA packages might be too large for a single schematic page. Splitting the part into separate sections allows you to organize parts of the symbol by function, creating cleaner circuit schematics. For example, you can create one slot for an I/O symbol, a second slot for a JTAG symbol, and a third slot for a power/ground symbol.

Figure 15-6: Splitting a Symbol into Multiple Sections
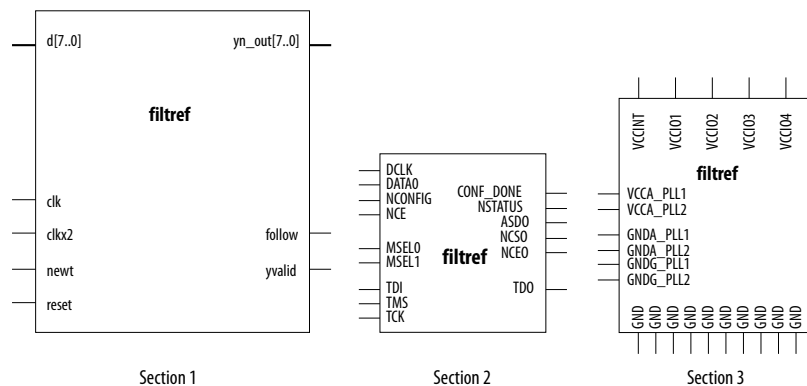
- This diagram represents a Cyclone device with JTAG or passive serial (PS) mode configuration option settings. Symbols created for other devices or other configuration modes might have different sets of configuration pins, but can be fractured in a similar manner.
- The power/ground section shows only a representation of power and ground pins because the device contains a high number of power and ground pins.

Note: Although symbol generation in the Design Entry CIS software refers to symbol fractures as sections, other tools use different names to refer to symbol fractures.

To split a part into sections, select the part in its library in the Project Manager window of the Cadence Allegro Design Entry CIS software. On the Tools menu, click **Split Part** or right-click the part and choose **Split Part**. The **Split Part Section Input Spreadsheet** appears.

Figure 15-7: Split Part Section Input Spreadsheet

Section Column

	Number	Name	Type	Order	Length	User Assig	I/O Bank	Voltage	I/O Standard	Location	Section
1	H1	clk	Input	0	Line		1			Left	1
2	G1	clkx2	Input	1	Line		1			Left	1
3	K13	CONF_DONE	Passive	2	Line		3			Left	2
4	C7	d[0]	Input	3	Line		2			Left	1
5	A6	d[1]	Input	4	Line		2			Left	1
6	D7	d[2]	Input	5	Line		2			Left	1
7	B7	d[3]	Input	6	Line		2			Left	1
8	B8	d[4]	Input	7	Line		2			Left	1
9	M7	d[5]	Input	8	Line		4			Left	1
10	A8	d[6]	Input	9	Line		2			Left	1
11	B6	d[7]	Input	10	Line		2			Left	1
12	H2	DATA0	Input	11	Line		1			Left	2
13	K4	DCLK	Bidirectional	12	Line		1			Left	2
14	C6	follow	Output	13	Line		2			Right	1
15	J3	MSEL0	Passive	14	Line		1			Left	2
16	J2	MSEL1	Passive	15	Line		1			Left	2
17	J4	nCE	Passive	16	Line		1			Left	2
18	H4	nCEO	Passive	17	Line		1			Left	2
19	H3	nCONFIG	Passive	18	Line		1			Left	2
20	H5	newt	Input	19	Line		1			Left	1
21	J13	nSTATUS	Passive	20	Line		3			Left	2
22	G16	reset	Input	21	Line		3			Left	1

Split...   Cancel   Help

Each row in the spreadsheet represents a pin in the symbol. The **Section** column indicates the section of the symbol to which each pin is assigned. You can locate all pins in a new symbol in section 1. You can change the values in the **Section** column to assign pins to various sections of the symbol. You can also specify the side of a section on the location of the pin by changing the values in the **Location** column. When you are ready, click **Split**. A new symbol appears in the same library as the original with the name <original part name>_**Split1.**

View and edit each section individually. To view the new sections of the part, double-click the part. The Part Symbol Editor window appears and the first section of the part displays for editing. On the View menu, click **Package** to view thumbnails of all the part sections. To edit the section of the symbol, double-click the thumbnail.

For more information about splitting parts into sections and editing symbol sections in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

Instantiating a Symbol in a Design Entry CIS Schematic

After saving a new symbol in a library in your Cadence Allegro Design Entry CIS project, you can instantiate the new symbol on a page in your schematic. Open a schematic page in the Project Manager window of the Cadence Allegro Design Entry CIS software. To add the newly created symbol to your schematic on the schematic page, on the Place menu, click **Part**. The **Place Part** dialog box appears.

Figure 15-8: Place Part Dialog Box
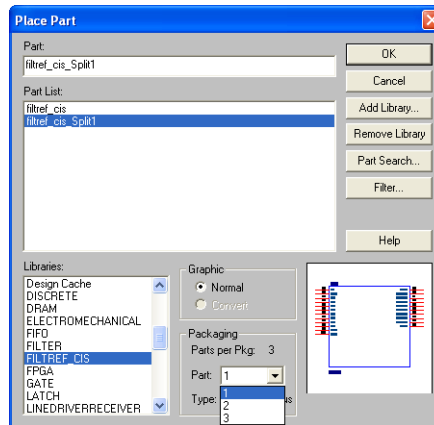
Select the new symbol library location and the newly created part name. If you select a part that is split into sections, you can select the section to place from the **Part** pop-up menu. Click **OK**. The symbol attaches to your cursor for placement in the schematic. To place the symbol, click on the schematic page.

For more information about using the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

Altera Libraries for the Cadence Allegro Design Entry CIS Software

Altera provides downloadable **.olb** for many of its device packages. You can add these libraries to your Cadence Allegro Design Entry CIS project and update the symbols with the pin assignments contained in the **.pin** generated by the Quartus Prime software. You can use the downloaded library symbols as a base for creating custom schematic symbols with your pin assignments that you can edit or fracture. This method increases productivity by reducing the amount of time it takes to create and edit a new symbol.

Using the Altera-provided Libraries with your Cadence Allegro Design Entry CIS Project

To use the Altera-provided libraries with your Cadence Allegro Design Entry CIS project, follow these steps:

1. Download the library of your target device from the Download Center page found through the Support page on the Altera website.
2. Create a copy of the appropriate **.olb** to maintain the original symbols. Place the copy in a convenient location, such as your Cadence Allegro Design Entry CIS project directory.
3. In the Project Manager window of the Cadence Allegro Design Entry CIS software, click once on the **Library** folder to select it. On the Edit menu, click **Project** or right-click the **Library** folder and choose **Add File** to select the copy of the downloaded **.olb** and add it to your project. You can locate the new library in the list of part libraries for your project.
4. On the Tools menu, click **Generate Part.** The **Generate Part** dialog box appears.
5. In the **Netlist/source file** field, click **Browse** to specify the **.pin** in your Quartus Prime design.
6. From the **Netlist/source file type** list, select **Altera Pin File**.
7. For **Part name**, type the name of the target device the same as it appears in the downloaded library file. For example, if you are using a device from the **CYCLONE06.OLB** library, type the part name to match one of the devices in this library such as `ep1c6f256`. You can rename the symbol in the Project Manager window after updating the part.

8. Set the **Destination part library** to the copy of the downloaded library you added to the project.
9. Select **Update pins on existing part in library**. Click **OK.**
10. Click **Yes**.

The symbol is updated with your pin assignments. Double-click the symbol in the Project Manager window to view and edit the symbol. On the View menu, click **Package** if you want to view and edit other sections of the symbol. If the symbol in the downloaded library is fractured into sections, you can edit each section but you cannot further fracture the part. You can generate a new part without using the downloaded part library if you require additional sections.

For more information about creating, editing, and fracturing symbols in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

Document Revision History

Table 15-3: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*.
June 2014	14.0.0	Converted to DITA format.
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	• General style editing. • Removed Referenced Document Section. • Added a link to Help in "Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA" on page 9–5.
November 2009	9.1.0	• Added "Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA" on page 9–5. • General style editing. • Edited Figure 9–4 on page 9–10 and Figure 9–8 on page 9–16.
March 2009	9.0.0	• Chapter 9 was previously Chapter 7 in the 8.1 software release. • No change to content.
November 2008	8.1.0	Changed to 8-1/2 x 11 page size.
May 2008	8.0.0	Updated references.

Related Information

Quartus Handbook Archive

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

2015.11.02

QPP5V2 ✉ **Subscribe** 💬 **Send Feedback**

You can integrate the Mentor Graphics® I/O Designer or DxDesigner PCB design tools into the Quartus Prime design flow. This combination provides a complete FPGA-to-board design workflow.

With today's large, high-pin-count and high-speed FPGA devices, good and correct PCB design practices are essential to ensure correct system operation. The PCB design takes place concurrently with the design and programming of the FPGA. The FPGA or ASIC designer initially creates signal and pin assignments, and the board designer must correctly transfer these assignments to the symbols in their system circuit schematics and board layout. As the board design progresses, Altera recommends reassigning pins to optimize the PCB layout. Ensure that you inform the FPGA designer of the pin reassignments so that the new assignments are included in an updated placement and routing of the design.

The Mentor Graphics I/O Designer software allows you to take advantage of the full FPGA symbol design, creation, editing, and back-annotation flow supported by the Mentor Graphics tools.

This chapter covers the following topics:

- Mentor Graphics and Altera software integration flow
- Generating supporting files
- Adding Quartus Prime I/O assignments to I/O Designer
- Updating assignment changes between the I/O Designer the Quartus Prime software
- Generating I/O Designer symbols
- Creating DxDesigner symbols from the Quartus Prime output files

This chapter is intended for board design and layout engineers who want to start the FPGA board integration while the FPGA is still in the design phase. Alternatively, the board designer can plan the FPGA pin-out and routing requirements in the Mentor Graphics tools and pass the information back to the Quartus Prime software for placement and routing. Part librarians can also benefit from this chapter by learning how to use output from the Quartus Prime software to create new library parts and symbols.

The procedures in this chapter require the following software:

- The Quartus Prime software version 5.1 or later
- DxDesigner software version 2004 or later
- Mentor Graphics I/O Designer software (optional)

Note: To obtain and license the Mentor Graphics tools and for product information, support, and training, refer to the Mentor Graphics website.
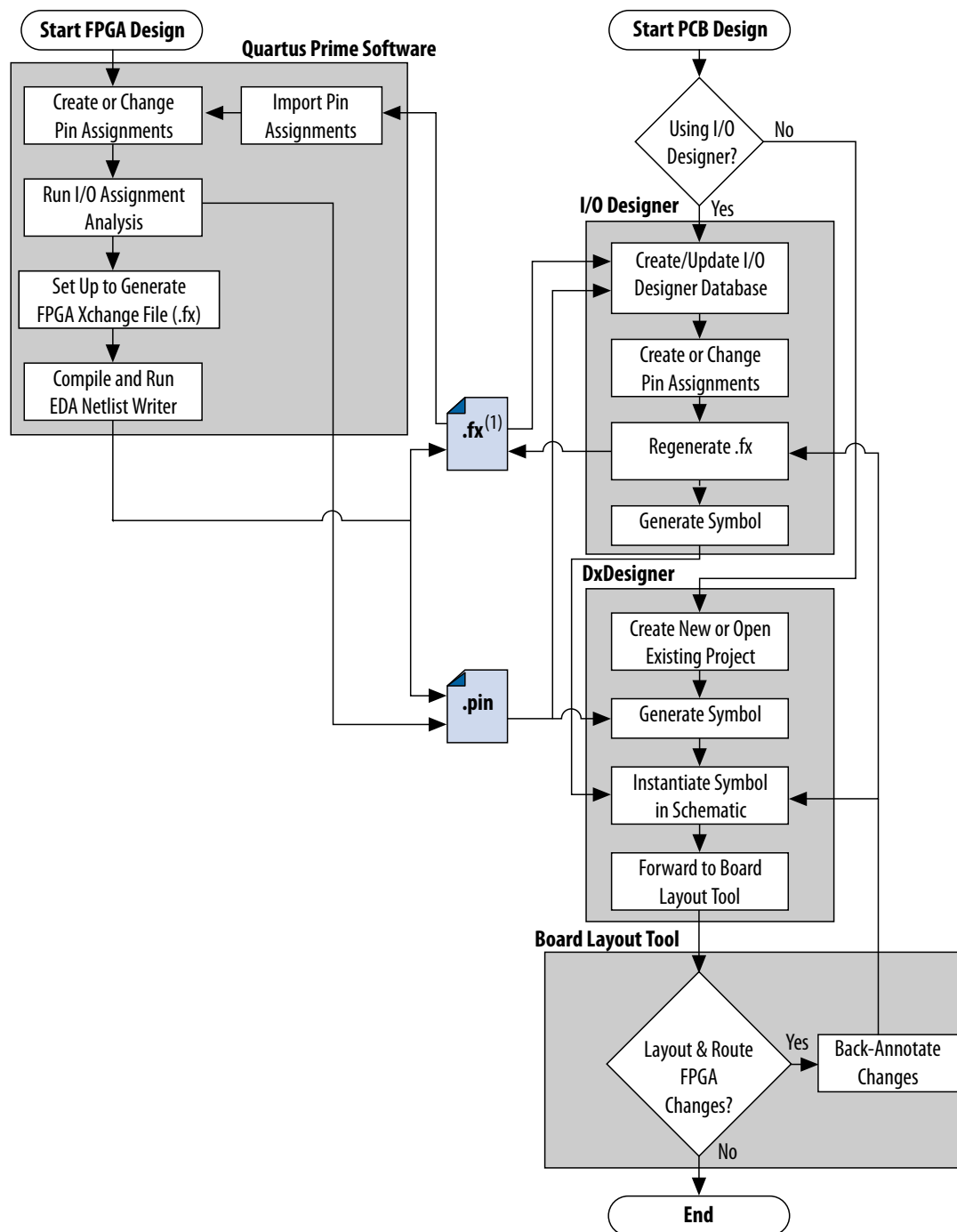
ISO 9001:2008 Registered

ALTERA®

FPGA-to-PCB Design Flow

You can create a design flow integrating an Altera® FPGA design from the Quartus Prime software, and a circuit schematic in the DxDesigner software.

Figure 16-1: Design Flow with and Without the I/O Designer Software

> **Note:** The Quartus Prime software generates the **.fx** in the output directory you specify in the **Board-Level** page of the **Settings** dialog box. However, the Quartus Prime software and the I/O Designer software can import pin assignments from an **.fx** located in any directory. Use a backup **.fx** to prevent overwriting existing assignments or importing invalid assignments.

To integrate the I/O Designer into your design flow, follow these steps:

1. In the Quartus Prime software, click **Assignments** > **Settings** > **EDA Tool Settings** > **Board-Level** to specify settings for **.fx** symbol file generation.
2. Compile your design to generate the **.fx** and Pin-Out File (**.pin**) in the Quartus Prime project directory.
3. Create a board design with the DxDesigner software and the I/O Designer software by performing the following steps:

   a. Create a new I/O Designer database based on the **.fx** and the **.pin** files.
   b. In the I/O Designer software, make adjustments to signal and pin assignments.
   c. Regenerate the **.fx** in the I/O Designer software to export the I/O Designer software changes to the Quartus Prime software.
   d. Generate a single or fractured symbol for use in the DxDesigner software.
   e. Add the symbol to the **sym** directory of a DxDesigner project, or specify a new DxDesigner project with the new symbol.
   f. Instantiate the symbol in your DxDesigner schematic and export the design to the board layout tool.
   g. Back-annotate pin changes created in the board layout tool to the DxDesigner software and back to the I/O Designer software and the Quartus Prime software.

4. Create a board design with the DxDesigner software without the I/O Designer software by performing the following steps:

   a. Create a new DxBoardLink symbol with the **Symbol** wizard and reference the **.pin** from the Quartus Prime software in an existing DxDesigner project.
   b. Instantiate the symbol in your DxDesigner schematic and export the design to a board layout tool.

> **Note:** You can update these symbols with design changes with or without the I/O Designer software. If you use the Mentor Graphics I/O Designer software and you change symbols with the DxDesigner software, you must reimport the symbols into I/O Designer to avoid overwriting your symbol changes.

Integrating with I/O Designer

You can integrate the Mentor Graphics I/O Designer software into the Quartus Prime design flow. Pin and signal assignment changes can be made anywhere in the design flow with either the Quartus Prime Pin Planner or the I/O Designer software. The I/O Designer software facilitates moving these changes, as well as synthesis, placement, and routing changes, between the Quartus Prime software, an external synthesis tool (if used), and a schematic capture tool such as the DxDesigner software.

This section describes how to use the I/O Designer software to transfer pin and signal assignment information to and from the Quartus Prime software with an **.fx**, and how to create symbols for the DxDesigner software.

Figure 16-2: I/O Designer Design Flow
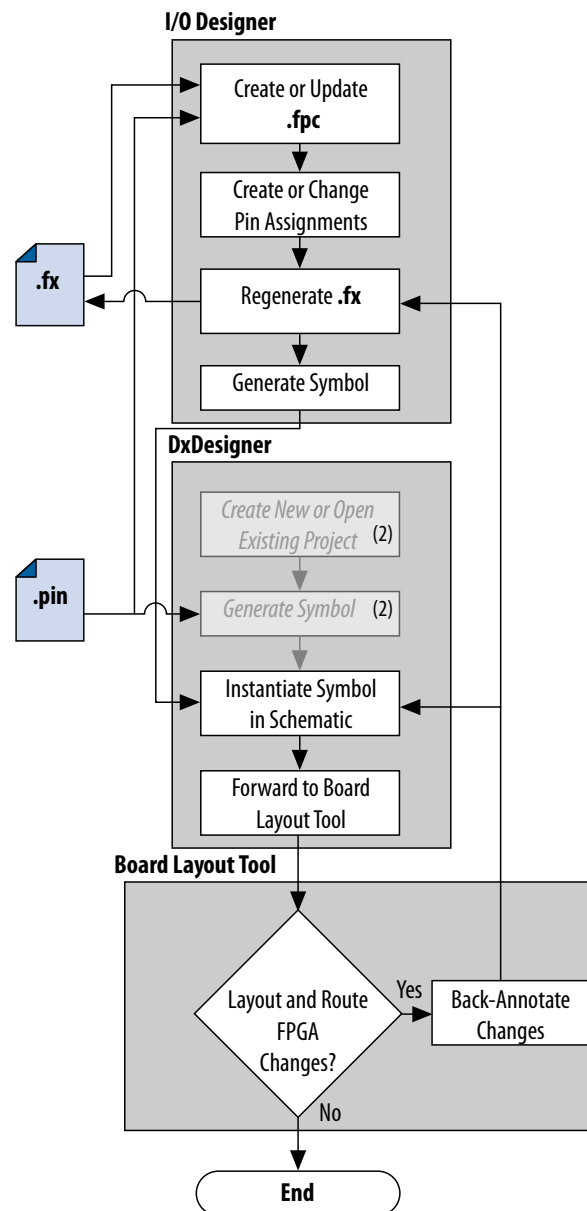
Note: (2) DxDesigner software-specific steps in the design flow are not part of the I/O Designer flow.

Generating Pin Assignment Files

You transfer I/O pin assignments from the Quartus Prime software to the Mentor Graphics PCB tools by generating optional **.pin** and **.fx** files during Quartus Prime compilation. These files contain pin assignment information for use in other tools. Click **Assignments** > **Settings** > **Board-Level** to specify settings for optional PCB tool file generation. Click **Processing** > **Start Compilation** to compile the design to generate the file(s) in the project directory.

The Quartus Prime-generated **.pin** contains the I/O pin name, number, location, direction, and I/O standard for all used and unused pins in the design. Click **Assignments** > **Pin Planner** to modify I/O pin

assignments. You cannot import pin assignment changes from a Mentor Graphics **.pin** into the Quartus Prime software.

The **.fx** is an input or output of either the Quartus Prime or I/O Designer software. You can generate an **.fx** in the Quartus Prime software for symbol generation in the Mentor Graphics I/O Designer software. A Quartus Prime **.fx** contains the pin name, number, location, direction, I/O standard, drive strength, termination, slew rate, IOB delay, and differential pins. An I/O Designer **.fx** additionally includes information about unused pins and pin set groups.

The I/O Designer software can also read from or update a Quartus Prime Settings File (**.qsf**). You can use the **.qsf** in the same way as use of the **.fx**, but pin swap group information does not transfer between I/O Designer and the Quartus Prime software. Use the **.fx** rather than the **.qsf** for transferring I/O assignment information.

Figure 16-3: Generating .pin and .fx files
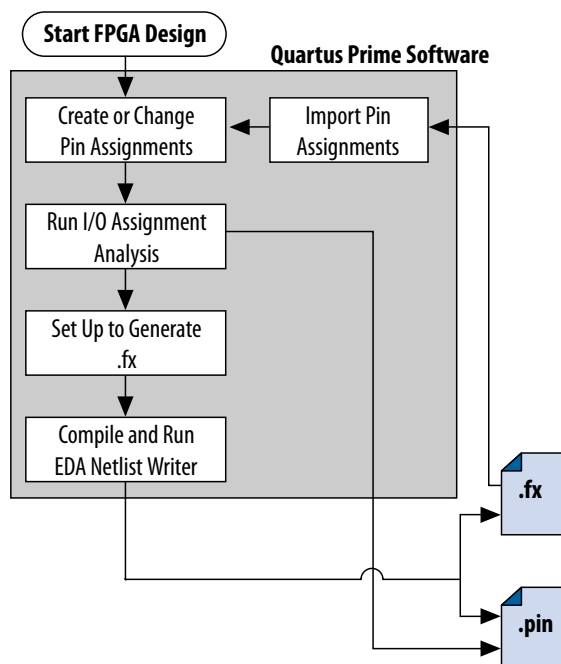
I/O Designer Settings

You can directly export I/O Designer symbols to the DxDesigner software. To set options for integrating I/O Designer with Dx Designer, follow these steps:

1. Start the I/O Designer software.
2. Click **Tools** > **Prefences**.
3. Click **Paths**, and then double-click the **DxDesigner executable file** path field to select the location of the DxDesigner application.
4. Click **Apply**.
5. Click **Symbol Editor**, and then click **Export**. In the Export type menu, under **General**, select **DxDesigner/PADS-Designer**.
6. Click **Apply**, and then click **OK**.

7. Click **File** > **Properties**.
8. Click the **PCB Flow** tab, and then click **Path to a DxDesigner project directory**.
9. Click **OK**.
   If you do not have a new DxDesigner project in the Database wizard and a DxDesigner project, you must create a new database with the DxDesigner software, and then specify the project location in I/O Designer.

Transferring I/O Assignments

You can transfer Quartus Prime signal and pin assignments contained in **.pin** and **.fx** files into an I/O Designer database. Use the I/O Designer Database Wizard to create a new database incorporating the **.fx** and **.pin** files. You can also create a new, empty database and manually add the assignment information. If there is no available signal or pin assignment information, you can create an empty database containing only a selection of the target device. This technique is useful if you know the signals in your design and the pins you want to assign. You can subsequently transfer this information to the Quartus Prime software for placement and routing.

Before you begin

You may create a very simple I/O Designer database that includes only the **.pin** or **.fx** file information. However, when using only a **.pin**, you cannot import I/O assignment changes from I/O Designer back into the Quartus Prime software without also generating an **.fx**. If your I/O Designer database includes only **.fx** file information, the database may not contain all the available I/O assignment information. The Quartus Prime **.fx** file only lists assigned pins. The **.pin** lists all assigned and unassigned device pins. Use both the **.pin** and the **.fx** to produce the most complete set of I/O Designer database information.

To create a new I/O Designer database using the Database wizard, follow these steps;

1. Start the I/O Designer software. The **Welcome to I/O Designer** dialog box appears. Select **Wizard to create new database** and click **OK.**
   If the **Welcome to I/O Designer** dialog box does not appear, you can access the wizard through the menu. To access the wizard, click **File** > **Database Wizard**.
2. Click **Next**. The **Define HDL source file** page appears
   If no HDL files are available, or if the **.fx** contains your signal and pin assignments, you can skip Step 3 and proceed to Step 4.
3. If your design includes a Verilog HDL or VHDL file, you can add a top-level Verilog HDL or VHDL file in the I/O Designer software. Adding a file allows you to create functional blocks or get signal names from your design. You must create all physical pin assignments in I/O Designer if you are not using an **.fx** or a **.pin**. Click **Next**. The **Database Name** page appears.
4. In the **Database Name** page, type your database file name. Click **Next**. The Database Location window appears.
5. Add a path to the new or an existing database in the **Location** field, or browse to a database location. Click **Next**. The **FPGA flow** page appears.
6. In the Vendor menu, click **Altera**.
7. In the Tool/Library menu, click **Quartus Prime <version>** to select your version of the Quartus Prime software.

   Note: The Quartus Prime software version listed may not match your actual software version. If your version is not listed, select the latest version. If your target device is not available, the device may not yet be supported by the I/O Designer software.

8. Select the appropriate device family, device, package, and speed (if applicable), from the corresponding menus. Click **Next**. The **Place and route** page appears.
9. In the **FPGAX file name** field, type or browse to the backup copy of the **.fx** generated by the Quartus Prime software.
10. In the **Pin report file name** field, type or browse to the **.pin** generated by the Quartus Prime software. Click **Next**.
    You can also select a **.qsf** for update. The I/O Designer software can update the pin assignment information in the **.qsf** without affecting any other information in the file.

    Note: You can import a **.pin** without importing an **.fx**. The I/O Designer software does not generate a **.pin**. To transfer assignment information to the Quartus Prime software, select an additional file and file type. Altera recommends selecting an **.fx** in addition to a **.pin** for transferring all the assignment information in the **.fx** and **.pin** files. In some versions of the I/O Designer software, the standard file picker may incorrectly look for a **.pin** instead of an **.fx**. In this case, select **All Files (*.*)** from the **Save as type** list and select the file from the list.

11. On the **Synthesis** page, specify an external synthesis tool and a synthesis constraints file for use with the tool. If you do not use an external synthesis tool, click **Next**.
12. On the **PCB Flow** page, you can select an existing schematic project or create a new project as a symbol information destination.

    *   To select an existing project, select Choose existing project and click Browse after the Project Path field. The Select project dialog box appears. Select the project.
    *   To create a new project, in the Select project dialog box, select Create new empty project. Type the project file name in the Name field and browse to the location where you want to save the file. Click OK.

13. If you have not specified a design tool to which you can send symbol information in the I/O Designer software, click **Advanced** in the **PCB Flow** page and select your design tool. If you select the DxDesigner software, you have the option to specify a Hierarchical Occurrence Attributes (**.oat**) file to import into the I/O Designer software. Click **Next** and then click **Finish** to create the database.Updating

Updating I/O Designer with Quartus Prime Pin Assignments

As you fine tune your design in the Quartus Prime software, changes to design logic and pin assignments are likely. You must transfer any pin assignment changes made during design iterations for correct analysis in your circuit schematic and board layout tools. You transfer Quartus Prime pin assignment changes to I/O Designer by updating the **.fx** and the **.pin** files in the Quartus Prime software. When you update the **.fx** or the **.pin**, the I/O Designer database imports the changes automatically when configured according to the following instructions.

Before you begin

Figure 16-4: Updating Quartus Prime Pin Assignments in I/O Designer
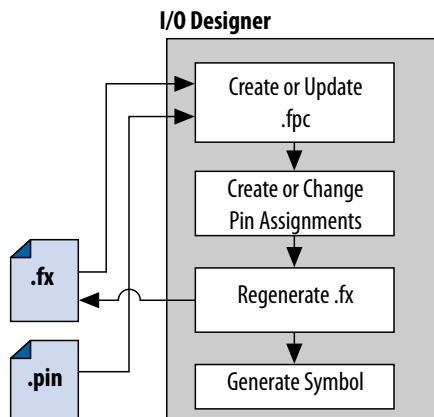
To update the **.fx** in your selected output directory and the **.pin** in your project directory after making changes to the design, perform the following tasks:

1. In the I/O Designer software, click **File** > **Properties**.
2. Under **FPGA Xchange**, specify the **.fx** file name and location.
3. Under **Place and Route**, specify the **.pin** file name and location.
   After you have set up these file locations, the I/O Designer software monitors these files for changes. If the specified **.fx** or **.pin** is modified during design processing, three indicators flash red in the lower right corner of the I/O Designer GUI. You can click the indicators to open the **I/O Designer Update Wizard** dialog box. The **I/O Designer Update Wizard** dialog box lists the updated files in the database.
4. Make logic or pin assignment changes in your design.
5. To preserve your changes an update the corresponding the **.fx** and **.pin** files, click **Processing** > **Start** > **Start EDA Netlist Writer** or **Processing** > **Start Compilation**.

   **Note:** Your I/O Designer database should us a backup copy of the **.fx** generated by the Quartus Prime software. Otherwise, updating the file in the Quartus Prime software overwrites any changes made to the file by the I/O Designer software. If there are I/O Designer assignments in the **.fx** that you want to preserve, create a backup copy of the file before updating it in the Quartus Prime software, and verify that your I/O Designer database points to the backup copy.

Updating Quartus Prime with I/O Designer Pin Assignments

As you fine tune your board design in I/O Designer, changes to signal routing and layout are likely. You must import any routing and layout changes into the Quartus Prime software for accurate place and route to match the new pin-out. The I/O Designer tool supports this flow.

Before you begin

Figure 16-5: Importing I/O Designer Pin Assignments
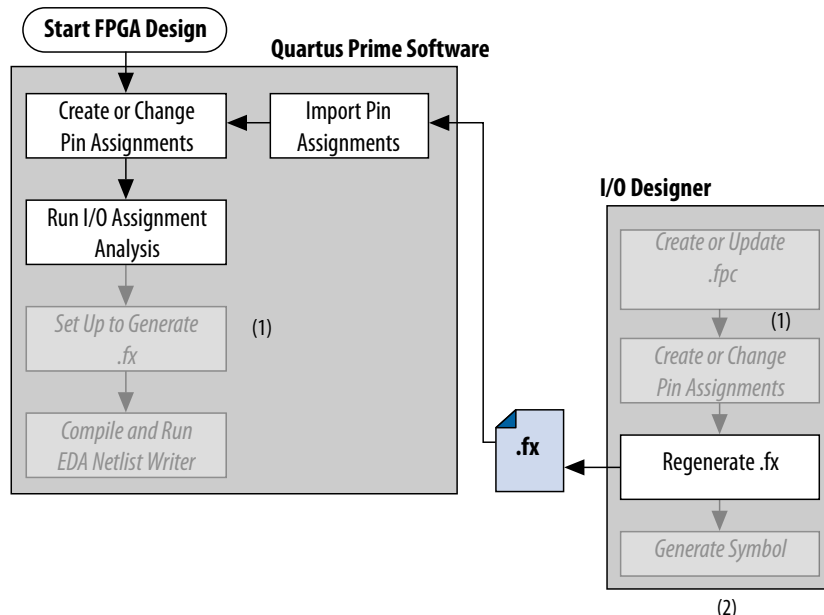
To import I/O Designer pin assignments, follow these steps:

1. Make pin assignment changes directly in the I/O Designer software, or the software can automatically update changes made in a board layout tool that are back-annotated to a schematic entry program such as the DxDesigner software.
2. To update the **.fx** with the changes, click **Generate** > **FPGA Xchange File**.
3. Open your Quartus Prime project.
4. Click **Assignments** > **Import Assignments**.
5. (Optional) To preserve original assignments before import, turn on **Copy existing assignments into <project name>.qsf.bak** before importing before importing the **.fx**.
6. Select the **.fx** and click **Open**.
7. Click **OK**.

Generating Schematic Symbols in I/O Designer

Circuit board schematic creation is one of the first tasks required in the design of a new PCB. You can use the I/O Designer software to generate schematic symbols for your Quartus Prime FPGA design for use in the DXDesigner schematic entry tools. The I/O Designer software can generate symbols for use in various Mentor Graphics schematic entry tools, and can import changes back-annotated by board layout tools to update the database and update the Quartus Prime software with the **.fx**

Most FPGA devices contain hundreds of pins, requiring large schematic symbols that may not fit on a single schematic page. Symbol designs in the I/O Designer software can be split or fractured into various functional blocks, allowing multiple part fractures on the same schematic page or across multiple pages. In the DxDesigner software, these part fractures join together with the use of the HETERO attribute.

You can use the I/O Designer **Symbol** wizard to quickly create symbols that you can subsequently refine. Alternatively, you can import symbols from another DXDesigner project, and then assign an FPGA to the symbol. To import symbols in the I/O Designer software, **File** > **Import Symbol**.

I/O Designer symbols are either functional, physical (PCB), or both. Signals imported into the database, usually from Verilog HDL or VHDL files, are the basis of a functional symbol. No physical device pins must be associated with the signals to generate a functional symbol. This section focuses on board-level PCB symbols with signals directly mapped to physical device pins through assignments in either the Quartus Prime Pin Planner or in the I/O Designer database.

Generating Schematic Symbols

To create a symbol based on a selected Altera FPGA device, follow these steps:

1. Start the I/O Designer software.
2. Click **Symbol** > **Symbol Wizard**.
3. In the **Symbol name** field, type the symbol name. The **DEVICE** and **PKG_TYPE** fields display the device and package information.

   Note: If **DEVICE** and **PKG_TYPE** are blank or incorrect, close the Symbol wizard and specify the correct device information (**File** > **Properties** > **FPGA Flow**).
4. Under **Symbol type**, click **PCB**. Under **Use signals**, click **All**, then click **Next**.
5. Select fracturing options for your symbol. If you are using the Symbol wizard to edit a previously created fractured symbol, you must turn on **Reuse existing fractures** to preserve your current fractures. Select other options on this page as appropriate for your symbol. Click **Next**.
6. Select additional fracturing options for your symbol. Click **Next**.
7. Select the options for the appearance of the symbols. Click **Next**.
8. Define the information you want to label for the entire symbol and for individual pins. Click **Next**.
9. Add any additional signals and pins to the symbol. Click **Finish**.
   You can view your symbol and any fractures you created with the Symbol Editor. You can edit parts of the symbol, delete fractures, or rerun the Symbol wizard. When you modify pin assignments in I/O Designer database, I/O Designer symbols automatically reflect these changes. Modify assignments in the I/O Designer software by supplying and updated **.fx** from the Quartus Prime software, or by back-annotating changes in your board layout tool.

Exporting Schematic Symbols to DxDesigner

You can export your I/O Designer schematic symbols for to DxDesigner for further design entry work. To generate all fractures of a symbol, click **Generate** > **All Symbols**. To generate only the currently displayed symbol, click **Generate** > **Current Symbol Only**. The DxDesigner project **/sym** directory preserves each symbol in the database as a separate file. You can instantiate the symbols in your DxDesigner schematics.
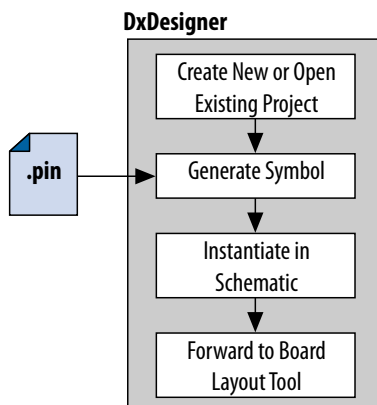
Integrating with DxDesigner

You can integrate the Mentor Graphics DxDesigner schematic capture tool into the Quartus Prime design flow. Use DxDesigner to create flat circuit schematics or to create hierarchical schematics that facilitate design reuse and a team-based design for all PCB types. Use DxDesigner in conjunction with I/O Designer software for a complete FPGA I/O and PCB design flow.

If you use DxDesigner without the I/O Designer software, the design flow is one-way, using only the **.pin** generated by the Quartus Prime software. You can only make signal and pin assignment changes in the

Quartus Prime software. You cannot back-annotate changes made in a board layout tool or in a
DxDesigner symbol to the Quartus Prime software.

Figure 16-6: DxDesigner-only Flow (without I/O Designer)

DxDesigner Project Settings

DxDesigner new projects automatically create FPGA symbols by default. However, if you are using the
I/O Designer with DxDesigner, you must enable DxBoardLink Flow options for integration with the I/O
Designer software. To enable the DxBoardLink flow design configuration when creating a new
DxDesigner project, follow these steps:

1. Start the DxDesigner software.
2. Click **File** > **New**, and then click the **Project** tab.
3. Click **More**. Turn on **DxBoardLink**. To enable the DxBoardLink Flow design configuration for an
   existing project, click **Design Configurations** in the Design Configuration toolbar and turn on
   DxBoardLink.

Creating Schematic Symbols in DxDesigner

You can create schematic symbols in the DxDesigner software manually or with the Symbol wizard. The
DxDesigner Symbol wizard is similar to the I/O Designer Symbol wizard, but with fewer fracturing
options. The DxDesigner Symbol wizard creates, fractures, and edits FPGA symbols based on the
specified Altera device. To create a symbol with the Symbol wizard, follow these steps;

1. Start the DxDesigner software.
2. Click **Symbol Wizard** in the toolbar.
3. Type the new symbol name in the name field and click **OK**.
4. Specify creation of a new symbol or modification of an existing symbol. To modify an existing symbol,
   specify the library path or alias, and select the existing symbol. To create a new symbol, select
   DxBoardLink for the symbol source. The DxDesigner block type defaults to Module because the FPGA
   design does not have an underlying DxDesigner schematic. Choose whether or not to fracture the
   symbol. Click **Next**.
5. Type a name for the symbol, an overall part name for all the symbol fractures, and a library name for
   the new library created for this symbol. By default, the part and library names are the same as the
   symbol name. Click **Next**.

6. Specify the appearance of the generated symbol and how itthe grid you have set in your DxDesigner project schematic. After making your selections. Click **Next**.

7. In the **FPGA vendor** list, select **Altera Quartus**. In the **Pin-Out file to import** field, select the **.pin** from your Quartus Prime project directory. You can also specify Fracturing Scheme, Bus pin, and Power pin options. Click **Next**.

8. Select to create or modify symbol attributes for use in the DxDesigner software. Click **Next**.

9. On the **Pin Settings** page, make any final adjustments to pin and label location and information. Each tabbed spreadsheet represents a fracture of your symbol. Click **Save Symbol**.
   After creating the symbol, you can examine and place any fracture of the symbol in your schematic. You can locate separate files of all the fractures you created in the library you specified or created in the **/sym** directory in your DxDesigner project. You can add the symbols to your schematics or you can manually edit the symbols or with the Symbol wizard.

Analyzing FPGA Simultaneous Switching Noise (SSN)

With the Quartus Prime software, you can extract pin assignment data and perform SSN analysis of your design. Perform SSN analysis early in the board layout stage as part of your overall pin planning process. Use the Quartus Prime SSN Analyzer to optimize the pin assignments for better SSN performance.

Scripting API

The I/O Designer software includes a command line Tcl interpreter. All commands input through the I/O Designer GUI translate into Tcl commands run by the tool. You can run individual Tcl commands or scripts in the I/O Designer Console window, rather than using the GUI.

You can use the following Tcl commands to control I/O Designer.

- `set_fpga_xchange_file` *<file name>*—specifies the **.fx** from which the I/O Designer software updates assignments.
- `update_from_fpga_xchange_file`—updates the I/O Designer database with assignment updates from the currently specified **.fx**.
- `generate_fpga_xchange_file`—updates the **.fx** with I/O Designer software changes for transfer back into the Quartus Prime software.
- `set_pin_report_file -quartus_pin` *<file name>*—imports assignment data from a Quartus Prime software .pin file.
- `symbolwizard`—runs the I/O Designer Symbol wizard.
- `set_dx_designer_project -path` *<path>*

Document Revision History

Table 16-1: Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Quartus Prime*.

Date	Version	Changes
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor.
June 2012	12.0.0	• Removed survey link.
December 2010	10.1.0	• Changed to new document template.

Related Information

Quartus Handbook Archive

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