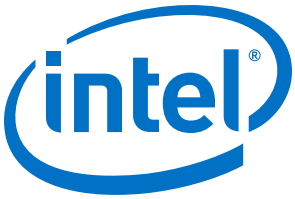


Intel® Quartus® Prime Standard Edition User Guide

Design Constraints

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

UG-20185 | 2019.01.10
Latest document on the web: PDF | HTML

Contents

1. Constraining Designs

The design constraints, assignments, and logic options that you specify influence how the Intel® Quartus® Prime Compiler implements your design. The Compiler attempts to synthesize and place logic in a manner than meets your constraints. In addition, design constraints also have an impact on how the Timing Analyzer and the Power Analyzer influence synthesis, placement, and routing.

You can specify design constraints in the GUI, with scripts, or directly in the files that store the constraints. The Intel Quartus Prime software preserves the constraints that you specify in the GUI in the following files:

- Intel Quartus Prime Settings file (`<project_directory>/<revision_name>.qsf`)—contains project-wide and instance-level assignments for the current revision of the project, in Tcl syntax. Each revision of a project has a separate `.qsf` file.
- Synopsys* Design Constraints file (`<project_directory>/<revision_name>.sdc`)—the Timing Analyzer uses industry-standard Synopsys Design Constraint format and stores those constraints in `.sdc` files.

By combining the syntax of the `.qsf` files and the `.sdc` files with procedural Tcl, you can automate iterations over several different settings, changing constraints and recompiling.

Related Information

- Intel Quartus Prime Standard Edition Settings File Reference Manual
  For information about all settings and constraints in the Intel Quartus Prime software.
- Tcl Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*
- Command Line Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*

1.1. Specifying Design Constraints Designs in the GUI

Intel Quartus Prime software provides tools that help you manually implement your project. These tools can also support design visualization, pre-filled parameters, and window cross probing, facilitating design exploration and debugging.

When you create or update a constraint in the Intel Quartus Prime software, the **System** tab of the **Messages** window displays the equivalent Tcl command. Utilize these commands as references for future scripted design definition and compilation.

1.1.1. Global Constraints and Assignments

Global constraints and project settings affect the entire Intel Quartus Prime project and all the applicable logic in the design. You often define global constraints in early project development; for example, when running the New Project Wizard. Intel Quartus Prime software stores global constraints in `.qsf` files, one for each project revision.

Table 1. Intel Quartus Prime Tools to Set Global Constraints

Assignment Type	Example	New Project Wizard	Device Dialog Box	Settings Dialog Box	Options Dialog Box
Project-wide	Project files	X		X	
Synthesis	• Device Family • Top-level Entity	X	X	X	
Fitter	• Device • Fitter Effort • IO Standard		X	X	
Simulation	Vector input source			X	
Third-party Tools	External Logic Analyzer				X
IP Settings	Maximum Platform Designer (Standard) Memory Usage				X

Related Information

Managing Project Settings
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

1.1.2. Node, Entity, and Instance-Level Constraints

Node, entity, and instance-level constraints apply to a subset of the design hierarchy. These constraints take precedence over any global assignment that affects the same sections of the design hierarchy.

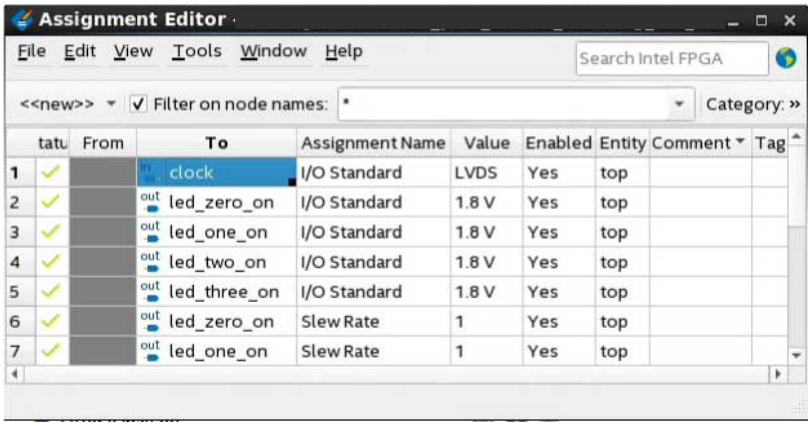
Table 2. Intel Quartus Prime Standard Edition Tools to Set Node, Entity and Instance Level Constraints

Assignment Type	Example	Assignment Editor	Chip Planner	Pin Planner
Pin	Project files	X		X
Location	• Device Family • Top-level Entity	X	X	
Routing	• Device • Fitter Effort • IO Standard	X	X	
Simulation	Vector input source	X	X	X

1.1.2.1. Specify Instance-Specific Constraints in Assignment Editor

Intel Quartus Prime Assignment Editor (**Assignments ➤ Assignment Editor**) provides a spreadsheet-like interface for assigning all instance-specific settings and constraints. To help you explore your design, the Assignment Editor allows you to filter assignments by node name or category.

Figure 1. Intel Quartus Prime Assignment Editor



Use the Assignment Editor to:

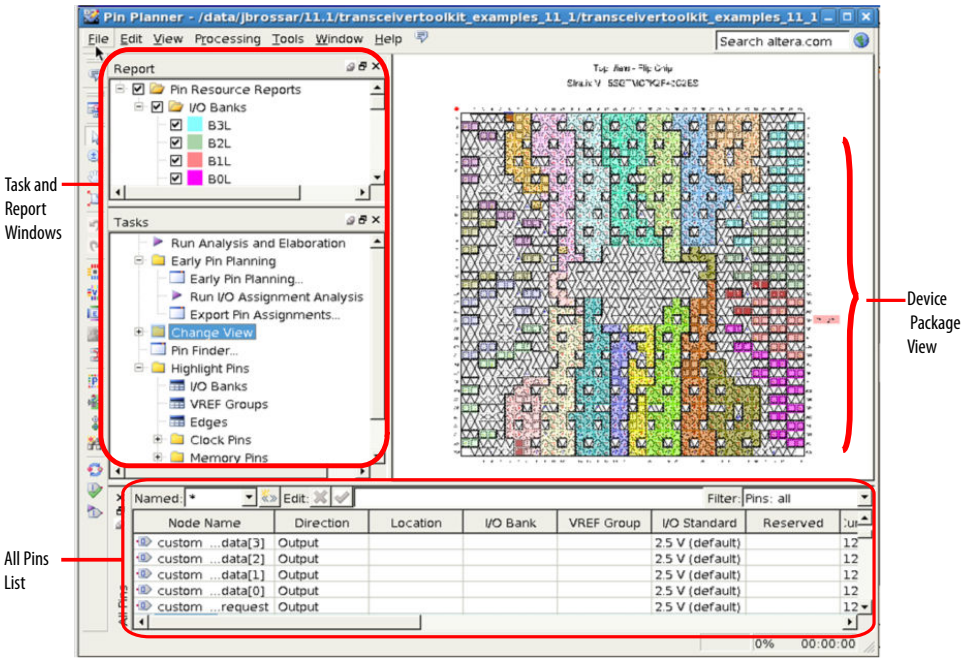
- Add, edit, or delete assignments for selected nodes
- Display information about specific assignments
- Enable or disable individual assignments
- Add comments to an assignment

Additionally, you can export assignments to a Comma-Separated Value File (`.csv`).

1.1.2.2. Specify I/O Constraints in Pin Planner

Intel Quartus Prime Pin Planner allows you to assign design elements to I/O pins. You can also plan and assign IP interface or user nodes not yet defined in the design.

Figure 2. Pin Planner GUI



Related Information

Managing Device I/O Pins on page 18

1.1.2.3. Adjust Constraints with the Chip Planner

With the Chip Planner you can adjust existing assignments to device resources, such as pins, logic cells, and LABs in a graphical representation of the device floorplan. You can also view equations and routing information and demote assignments by dragging and dropping to Logic Lock (Standard) regions in the **Logic Lock (Standard) Regions Window**.

Related Information

Design Floorplan Analysis in the Chip Planner
In *Intel Quartus Prime Standard Edition User Guide: Design Optimization*

1.1.2.4. Constraining Designs with the Design Partition Planner

The Design Partition Planner allows you to view design connectivity and hierarchy and can assist you in creating effective design partitions.

Additionally, the Design Partition Planner allows you to optimize design performance by isolating and resolving failing paths on a partition-by-partition basis.

Related Information

Creating Partitions and Logic Lock (Standard) Regions with the Design Partition Planner and the Chip Planner
In *Intel Quartus Prime Standard Edition User Guide: Design Optimization*

1.1.3. Probing Between Components of the Intel Quartus Prime GUI

Intel Quartus Prime software allows you to locate nodes and instances across windows and source files.

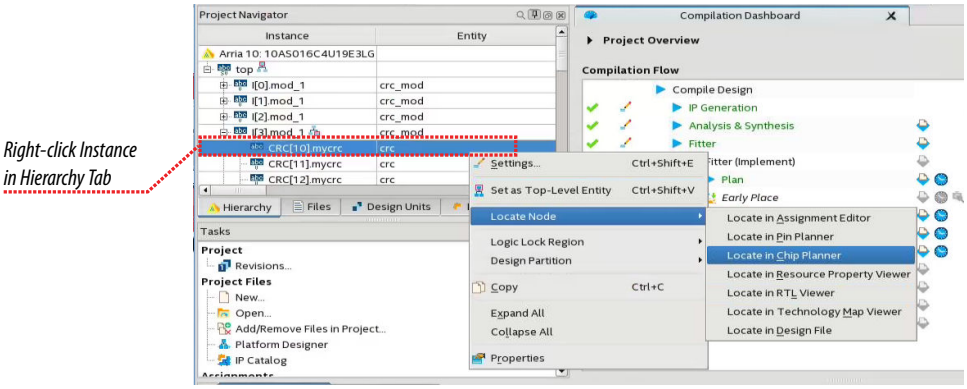
When you are in the Project Navigator, Assignment Editor, Chip Planner, or Pin Planner, and want to display a given resource in other Intel Quartus Prime tool:

1. Right-click the resource you want to display.
2. Click **Locate Node**, and then click one of the menu options.

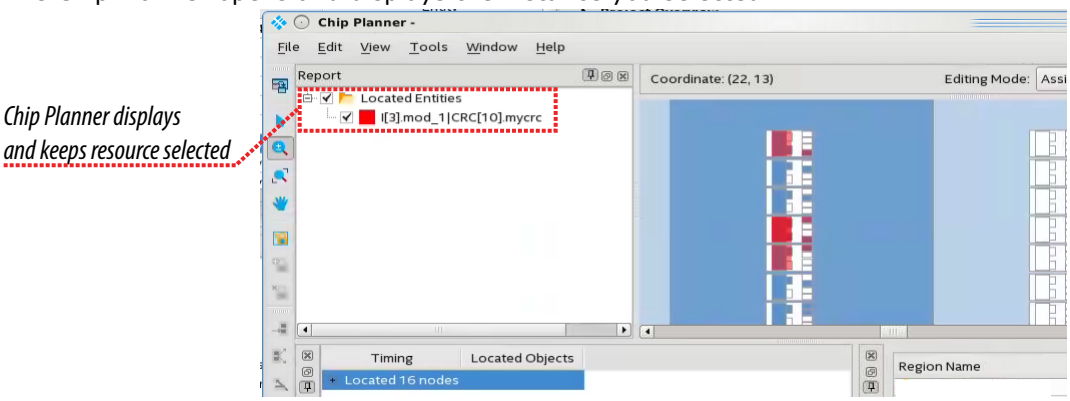
The corresponding window opens—or appears in the foreground if it is already open—and shows the element you clicked.

Example 1. Locate a Resource Selected in the Project Navigator

In the **Entity** list of the **Hierarchy** tab, right-click one object, and click **Locate ➤ Locate in Chip Planner**.



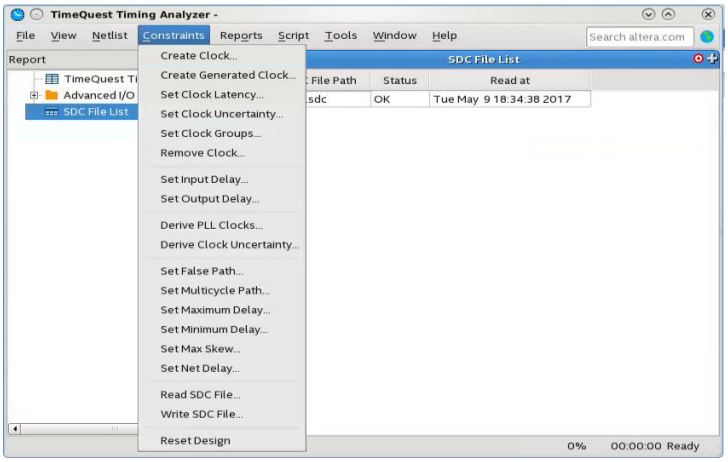
The Chip Planner opens and displays the instance you selected.



1.1.4. Specifying Timing Constraints in the GUI

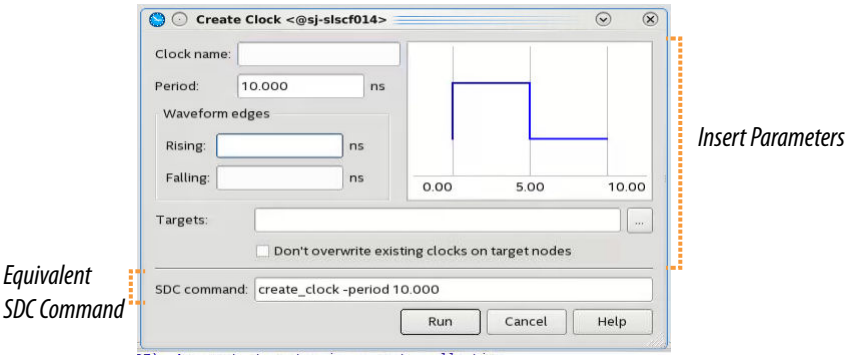
You can specify timing constraints in the Timing Analyzer GUI. Click the Constraints menu in the Timing Analyzer to specify timing constraints that you can apply to your project.

Figure 3. Constraint menu in Timing Analyzer



When you specify a constraint in the GUI, the dialog box displays the equivalent SDC command syntax.

Example 2. Create Clock Dialog Box



Individual timing assignments override project-wide requirements.

- To avoid reporting incorrect or irrelevant timing violations, you can assign timing exceptions to nodes and paths.
- The 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.

Related Information

Using the Timing Analyzer
In *Intel Quartus Prime Standard Edition User Guide: Timing Analyzer*

1.2. Constraining Designs with Tcl Scripts

You can perform all your design assignments using `.sdc` and `.qsf` setting files. To integrate these files in compilation and optimization flows, use Tcl scripts. Even though `.sdc` and `.qsf` files are written in Tcl syntax, they are not executable by themselves.

When you use Intel Quartus Prime Tcl packages, your scripts can open projects, make the assignments, compile the design, and compare compilation results against known goals and benchmarks. Furthermore, such a script can automate the iterative design process by modifying constraints and recompiling the design.

1.2.1. Create a Project and Apply Constraints

The command-line executables include options for common global project settings and commands. You can use a Tcl script to apply constraints such as pin locations and timing assignments. You can write a Tcl constraint file, or generate one for an existing project by clicking **Project ➤ 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 <Intel Quartus Prime *installation directory*>`/qdesigns/fir_filter/` directory.

```
project_new filtref -overwrite
# Assign family, device, and top-level file
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C12F256C6
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
#
project_close
```

Save the script in a file called `setup_proj.tcl` and type the commands illustrated in the example 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.

1.2.2. Assigning a Pin

To assign a signal to a pin or device location, use the Tcl command shown in this example:

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

Valid locations are pin location names. Some device families also support edge and I/O bank locations. Edge locations are EDGE_BOTTOM, EDGE_LEFT, EDGE_TOP, and EDGE_RIGHT. I/O bank locations include IOBANK_1 to IOBANK_n, where n is the number of I/O banks in a device.

1.2.3. Generating Intel Quartus Prime Settings Files

Intel Quartus Prime software allows you to generate .qsf files from your revision. You can embed these constraints in a scripted compilation flow, and even create sets of .qsf files for design optimization.

To generate a .qsf file from the Intel Quartus Prime software, click **Assignments ➤ Export Assignments**.

To organize the .qsf in a human readable form, **Project ➤ Organize Intel Quartus Prime Settings File**.

Example 3. Organized .qsf File

This example shows how .qsf files characterize a design revision. The set_global_assignment command makes all global constraints and software settings and set_location_assignment constrains each I/O node in the design to a physical pin on the device.

```
# Project-Wide Assignments
# ========================
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 9.1
set_global_assignment -name PROJECT_CREATION_TIME_DATE "10:37:10  MAY 7, 2009"
set_global_assignment -name LAST_QUARTUS_VERSION "17.0.0 Standard Edition"
set_global_assignment -name VERILOG_FILE mult.v
set_global_assignment -name VERILOG_FILE accum.v
set_global_assignment -name BDF_FILE filtref.bdf
set_global_assignment -name VERILOG_FILE hvalues.v
set_global_assignment -name VERILOG_FILE taps.v
set_global_assignment -name VERILOG_FILE state_m.v
set_global_assignment -name VERILOG_FILE acc.v
set_global_assignment -name SMART_RECOMPILE ON
set_global_assignment -name VECTOR_WAVEFORM_FILE fir.vwf

# Pin & Location Assignments
# ==========================
set_location_assignment PIN_F13 -to reset
set_location_assignment PIN_G10 -to d[2]
set_location_assignment PIN_F12 -to clk
set_location_assignment PIN_A10 -to clkx2
set_location_assignment PIN_G9 -to d[1]
set_location_assignment PIN_C12 -to d[7]
set_location_assignment PIN_F10 -to follow
set_location_assignment PIN_F9 -to yvalid
set_location_assignment PIN_E13 -to yn_out[2]
set_location_assignment PIN_E10 -to yn_out[3]
set_location_assignment PIN_C11 -to d[4]
set_location_assignment PIN_F11 -to d[0]
set_location_assignment PIN_C13 -to d[6]
set_location_assignment PIN_C8 -to yn_out[6]
```

```
set_location_assignment PIN_B13 -to d[5]
set_location_assignment PIN_B11 -to d[3]
set_location_assignment PIN_B10 -to yn_out[5]
set_location_assignment PIN_B8 -to yn_out[0]
set_location_assignment PIN_A13 -to yn_out[7]
set_location_assignment PIN_A11 -to yn_out[4]
set_location_assignment PIN_A12 -to yn_out[1]
set_location_assignment PIN_A9 -to newt

# Classic Timing Assignments
# ==========================
set_global_assignment -name FMAX_REQUIREMENT "85 MHz"

# Analysis & Synthesis Assignments
# ================================
set_global_assignment -name FAMILY "Cyclone IV GX"
set_global_assignment -name TOP_LEVEL_ENTITY filtref
set_global_assignment -name DEVICE_FILTER_PACKAGE FBGA
set_global_assignment -name DEVICE_FILTER_PIN_COUNT 256
set_global_assignment -name DEVICE_FILTER_SPEED_GRADE 6
set_global_assignment -name CYCLONE_OPTIMIZATION_TECHNIQUE SPEED
set_global_assignment -name MUX_RESTRUCTURE OFF

# Fitter Assignments
# ==================
set_global_assignment -name DEVICE EP4CGX15BF14C6
set_global_assignment -name FITTER_EFFORT "STANDARD FIT"
set_global_assignment -name PHYSICAL_SYNTHESIS_REGISTER_RETIMING ON
set_global_assignment -name PHYSICAL_SYNTHESIS_EFFORT EXTRA
set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "2.5 V"

# Simulator Assignments
# =====================
set_global_assignment -name VECTOR_INPUT_SOURCE fir.vwf

# start CLOCK(clockb)
# -------------------
# Classic Timing Assignments
    # ==========================
    set_global_assignment -name BASED_ON_CLOCK_SETTINGS clocka -section_id
clockb
    set_global_assignment -name DIVIDE_BASE_CLOCK_PERIOD_BY 2 -section_id
clockb
    set_global_assignment -name OFFSET_FROM_BASE_CLOCK "500 ps" -section_id
clockb

# end CLOCK(clockb)
# -----------------

# start CLOCK(clocka)
# -------------------
# Classic Timing Assignments
    # ==========================
    set_global_assignment -name FMAX_REQUIREMENT "100 MHz" -section_id clocka

# end CLOCK(clocka)
# -----------------

# ---------------------
# start ENTITY(filtref)
# Classic Timing Assignments
    # ==========================
    set_instance_assignment -name CLOCK_SETTINGS clocka -to clk
    set_instance_assignment -name CLOCK_SETTINGS clockb -to clkx2
    set_instance_assignment -name MULTICYCLE 2 -from clk -to clkx2
# Fitter Assignments
    # ==================
    set_instance_assignment -name SLEW_RATE 2 -to yvalid
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[0]
    set_instance_assignment -name SLEW_RATE 2 -to follow
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[7]
```

```
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[6]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[5]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[4]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[3]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[2]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[1]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
follow
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[7]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[6]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[5]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[4]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[3]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[2]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[1]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[0]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yvalid
# start DESIGN_PARTITION(Top)
    # ---------------------------
# Incremental Compilation Assignments
# ===================================
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
# end DESIGN_PARTITION(Top)
    # -------------------------

# end ENTITY(filtref)
# -------------------
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -
section_id Top
```

Related Information

Intel Quartus Prime Standard Edition Settings File Reference Manual
For information about all settings and constraints in the Intel Quartus Prime software.

1.2.4. Synopsys Design Constraint (.sdc) Files

Intel Quartus Prime software keeps timing constraints in `.sdc` files, which use Tcl syntax. You can embed these constraints in a scripted compilation flow, and even create sets of `.sdc` files for timing optimization.

Example 4. .sdc File

The example shows the timing constrains of a small design.

```
## PROGRAM "Quartus Prime"
## VERSION "Version 17.0.0 Build 595 04/25/2017 SJ Standard Edition"
## DATE    "Wed May 10 14:03:25 2017"
##
## DEVICE  "EP4CGX15BF14C6"
##
#**************************************************************
# Time Information
```

```
#**************************************************************
set_time_format -unit ns -decimal_places 3
#**************************************************************
# Create Clock
#**************************************************************
create_clock -name {clk} -period 4.000 -waveform { 0.000 2.000 } [get_ports
{clk}]
create_clock -name {clkx2} -period 4.000 -waveform { 0.000 2.000 } [get_ports
{clkx2}]
#**************************************************************
# Set Clock Uncertainty
#**************************************************************
set_clock_uncertainty -rise_from [get_clocks {clkx2}] -rise_to [get_clocks
{clkx2}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clkx2}] -fall_to [get_clocks
{clkx2}]  0.020
set_clock_uncertainty -fall_from [get_clocks {clkx2}] -rise_to [get_clocks
{clkx2}]  0.020
set_clock_uncertainty -fall_from [get_clocks {clkx2}] -fall_to [get_clocks
{clkx2}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks
{clkx2}]  0.040
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks
{clkx2}]  0.040
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks
{clk}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks
{clk}]  0.020
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks
{clkx2}]  0.040
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks
{clkx2}]  0.040
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks
{clk}]  0.020
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks
{clk}]  0.020
#**************************************************************
# Set False Path
#**************************************************************
set_false_path -from [get_clocks {clk clkx2}] -through [get_pins -
compatibility_mode *]  -to [get_clocks {clk clkx2}]
```

Related Information

Constraining and Analyzing with Tcl Commands
In *Intel Quartus Prime Standard Edition User Guide: Timing Analyzer*

1.2.5. Tcl-only Script Flows

As an alternative to `.sdc` and `.qsf` files, you can perform all design assignments and timing constraints inside the Tcl scripts. In this case, the script that automates compilation and custom results reporting also contains the design constraints.

You can export a design's contents to a procedural, executable Tcl (`.tcl`) file, and then use the generated script to restore settings after experimenting with other constraints.

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

Example 5. fir_filter_generated.tcl Tcl file

```
# Quartus Prime: Generate Tcl File for Project
# File: fir_filter_generated.tcl
# Generated on: Tue May  9 18:41:24 2017
# 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) "fir_filter"]} {
      puts "Project fir_filter is not open"
      set make_assignments 0
      }
} else {
    # Only open if not already open
    if {[project_exists fir_filter]} {
      project_open -revision filtref fir_filter
    } else {
      project_new -revision filtref fir_filter
    }
    set need_to_close_project 1
}

# Make assignments
if {$make_assignments} {
    set_global_assignment -name ORIGINAL_QUARTUS_VERSION 9.1
    set_global_assignment -name PROJECT_CREATION_TIME_DATE "10:37:10  MAY 7,
2009"
    set_global_assignment -name LAST_QUARTUS_VERSION "17.0.0 Standard Edition"
    set_global_assignment -name VERILOG_FILE mult.v
    set_global_assignment -name VERILOG_FILE accum.v
    set_global_assignment -name BDF_FILE filtref.bdf
    set_global_assignment -name VERILOG_FILE hvalues.v
    set_global_assignment -name VERILOG_FILE taps.v
    set_global_assignment -name VERILOG_FILE state_m.v
    set_global_assignment -name VERILOG_FILE acc.v
    set_global_assignment -name SMART_RECOMPILE ON
    set_global_assignment -name VECTOR_WAVEFORM_FILE fir.vwf
    set_global_assignment -name FMAX_REQUIREMENT "85 MHz"
    set_global_assignment -name FAMILY "Cyclone IV GX"
    set_global_assignment -name DEVICE_FILTER_PACKAGE FBGA
    set_global_assignment -name DEVICE_FILTER_PIN_COUNT 256
    set_global_assignment -name DEVICE_FILTER_SPEED_GRADE 6
    set_global_assignment -name CYCLONE_OPTIMIZATION_TECHNIQUE SPEED
    set_global_assignment -name MUX_RESTRUCTURE OFF
    set_global_assignment -name DEVICE EP4CGX15BF14C6
    set_global_assignment -name FITTER_EFFORT "STANDARD FIT"
    set_global_assignment -name PHYSICAL_SYNTHESIS_REGISTER_RETIMING ON
    set_global_assignment -name PHYSICAL_SYNTHESIS_EFFORT EXTRA
    set_global_assignment -name STRATIX_DEVICE_IO_STANDARD "2.5 V"
    set_global_assignment -name VECTOR_INPUT_SOURCE fir.vwf
    set_global_assignment -name BASED_ON_CLOCK_SETTINGS clocka -section_id
clockb
    set_global_assignment -name DIVIDE_BASE_CLOCK_PERIOD_BY 2 -section_id
clockb
    set_global_assignment -name OFFSET_FROM_BASE_CLOCK "500 ps" -section_id
clockb
    set_global_assignment -name FMAX_REQUIREMENT "100 MHz" -section_id clocka
    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_location_assignment PIN_F13 -to reset
    set_location_assignment PIN_G10 -to d[2]
    set_location_assignment PIN_F12 -to clk
    set_location_assignment PIN_A10 -to clkx2
    set_location_assignment PIN_G9 -to d[1]
    set_location_assignment PIN_C12 -to d[7]
    set_location_assignment PIN_F10 -to follow
    set_location_assignment PIN_F9 -to yvalid
    set_location_assignment PIN_E13 -to yn_out[2]
    set_location_assignment PIN_E10 -to yn_out[3]
    set_location_assignment PIN_C11 -to d[4]
```

```
    set_location_assignment PIN_F11 -to d[0]
    set_location_assignment PIN_C13 -to d[6]
    set_location_assignment PIN_C8 -to yn_out[6]
    set_location_assignment PIN_B13 -to d[5]
    set_location_assignment PIN_B11 -to d[3]
    set_location_assignment PIN_B10 -to yn_out[5]
    set_location_assignment PIN_B8 -to yn_out[0]
    set_location_assignment PIN_A13 -to yn_out[7]
    set_location_assignment PIN_A11 -to yn_out[4]
    set_location_assignment PIN_A12 -to yn_out[1]
    set_location_assignment PIN_A9 -to newt
    set_instance_assignment -name CLOCK_SETTINGS clocka -to clk
    set_instance_assignment -name CLOCK_SETTINGS clockb -to clkx2
    set_instance_assignment -name MULTICYCLE 2 -from clk -to clkx2
    set_instance_assignment -name SLEW_RATE 2 -to yvalid
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[0]
    set_instance_assignment -name SLEW_RATE 2 -to follow
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[7]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[6]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[5]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[4]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[3]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[2]
    set_instance_assignment -name SLEW_RATE 2 -to yn_out[1]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
follow
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[7]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[6]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[5]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[4]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[3]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[2]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[1]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yn_out[0]
    set_instance_assignment -name CURRENT_STRENGTH_NEW "MINIMUM CURRENT" -to
yvalid
    set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -
section_id Top

    # Commit assignments
    export_assignments

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

The example:

- Opens the project
- Assigns Constraints
- Writes assignments to QSF file
- Closes project

1.2.5.1. Tcl-only Timing Analysis

To avoid using a separated file to keep your timing constraints, copy and paste the .sdc file into your executable timing analysis script.

1.3. A Fully Iterative Scripted Flow

The **::quartus::flow** Tcl package in the Intel Quartus Prime Tcl API allows you to modify design constraints and recompile in an iterative flow.

Related Information

- ::quartus::flow
  In *Intel Quartus Prime Help*
- Command Line Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*

1.4. Constraining Designs Revision History

Document Version	Intel Quartus Prime Version	Changes
2019.01.04	18.1.0	• Clarified default location of .sdc and .qsf files in "Constraining Designs" topic. • Added two new "Assigning a Pin" and "Creating a Project and Applying Constraints" topics showing Tcl examples.
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.
2017.11.06	17.1.0	• Renamed topic: Constraining Designs with the GUI to Constraining Designs with Quartus Prime Tools. • Renamed topic: Global Constraints to Global Constraints and Assignments. • Added table: Quartus Prime Tools to Set Global Constraints. • Removed topic: Common Types of Global Constraints. • Removed topic: Settings That Direct Compilation and Analysis Flows. • Updated topic: Node, Entity and Instance-Level Constraints. • Added table: Quartus Prime Tools to Set Node, Entity and Instance Level Constraints. • Added topic: Assignment Editor. • Updated topic: Constraining Designs with the Pin Planner. • Updated topic: Constraining Designs with the Chip Planner. • Added topic: Constraining designs with the Design Partition Planner. • Updated topic: Probing Between Components of the Quartus Prime GUI. • Added example: Locate a Resource Selected in the Project Navigator. • Updated topic: SDC and the Timing Analyzer, and renamed to Specifying Individual Timing Constraints. • Added figure: Constraint Menu in Timing Analyzer. • Added example: Create Clock Dialog Box. • Updated topic: Constraining Designs with Tcl, and renamed to Constraining Designs with Tcl Scripts • Updated topic: Quartus Prime Settings Files and Tcl , and renamed to Generating Quartus Prime Settings Files. • Added example: blinking_led.qsf File. • Updated topic: Timing Analysis with Synopsys Design Constraints and Tcl, and renamed to Timing Analysis with .sdc Files and Tcl Scripts.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Added example: .sdc File with Timing Constraints. • Added topic: Tcl-only Script Flows. • Updated topic: A Fully Iterative Scripted Flow.
2015.11.02	15.1.0	• Changed instances of *Quartus II* to *Intel 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 II Help.
November 2009	9.1.0	• Added two notes. • Minor text edits.
March 2009	9.0.0	• Revised and reorganized the entire chapter. • Added section "Probing to Source Design Files and Other Quartus 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 II software 8.0 revision and date.

Related Information

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

2. Managing Device I/O Pins

This chapter describes efficient planning and assignment of I/O pins in your target device. Consider I/O standards, pin placement rules, and your PCB characteristics early in the design phase.

Figure 4. Pin Planner GUI

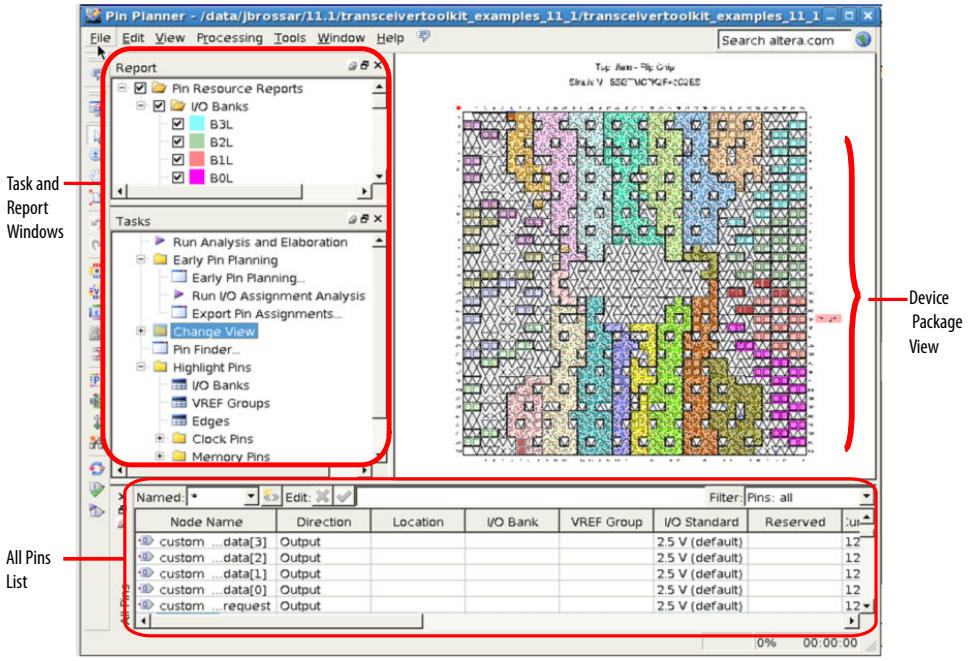


Table 3. Intel Quartus Prime I/O Pin Planning Tools

I/O Planning Task	Click to Access
Edit, validate, or export pin assignments	**Assignments ➤ Pin Planner**
View tailored pin planning advice	**Tools ➤ Advisors ➤ Pin Advisor**
Validate pin assignments against design rules	**Processing ➤ Start ➤ Start I/O Assignment Analysis**

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

Related Information

Instantiating the HPS Component
In *Intel Arria 10 Hard Processor System Technical Reference Manual*

2.1. I/O Planning Overview

On FPGA design, I/O planning includes creating pin-related assignments and validating them against pin placement guidelines. This process ensures a successful fit in your target device. When you plan and assign I/O pins in the initial stages of your project, you design for compatibility with your target device and PCB characteristics. As a result, your design process goes through fewer iterations, and you develop an accurate PCB layout sooner.

You can plan your I/O pins even before defining design files. Assign expected nodes not yet defined in design files, including interface IP core signals, and then generate a top-level file. The top-level file instantiates the next level of design hierarchy and includes interface port information like memory, high-speed I/O, device configuration, and debugging tools.

Assign design elements, I/O standards, interface IP, and other properties to the device I/O pins by name or by dragging to cells. You can then generate a top-level design file for I/O validation.

Use I/O assignment validation to fully analyze I/O pins 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.

Intel Quartus Prime software provides the Pin Planner tool 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.

2.1.1. 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 I/O standards, voltage and power supply requirements, and available I/O pins.
2. Click **Assignments ➤ Pin Planner**.
3. To setup a top-level HDL wrapper file that defines early port and interface information for your design, click **Early Pin Planning** in the **Tasks** pane.
    a. Click **Import IP Core** to import any defined IP core, and then assign signals to the interface IP nodes.
    b. Click **Set Up Top-Level File** and assign user nodes to device pins. User nodes become virtual pins in the top-level file and are not assigned to device pins.
    c. Click **Generate Top-Level File**. Use top-level file to validate I/O assignments.
4. Assign I/O properties to match your device and PCB characteristics, including assigning logic, I/O standards, output loading, slew rate, and current strength.
5. Click **Run I/O Assignment Analysis** in the **Tasks** pane to validate assignments and generate a synthesized design netlist. Correct any problems reported.
6. Click **Processing ➤ Start Compilation**. During compilation, the Intel Quartus Prime software runs I/O assignment analysis.

2.1.2. Integrating PCB Design Tools

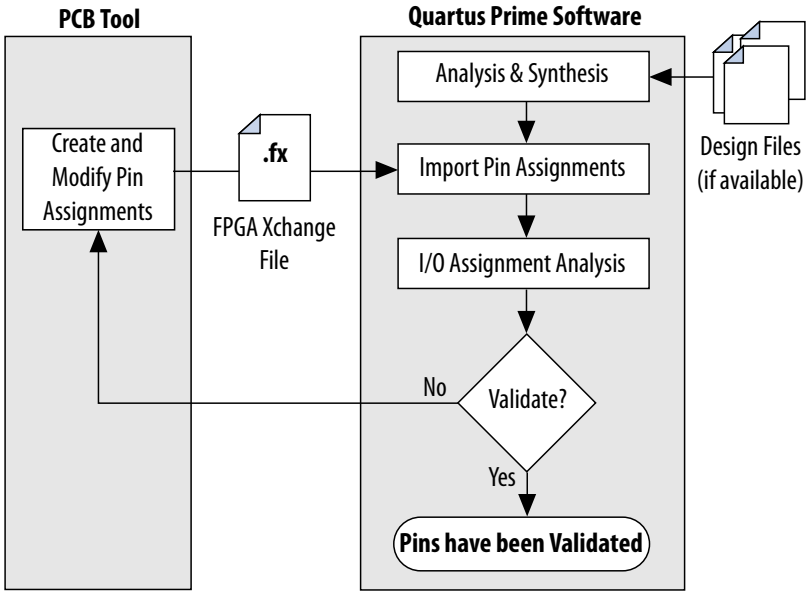
You can integrate PCB design tools into your work flow to map pin assignments to symbols in your system circuit schematics and board layout.

The Intel Quartus Prime software integrates with board layout tools by allowing import and export of pin assignment information in Intel Quartus Prime Settings Files (`.qsf`), Pin-Out File (`.pin`), and FPGA Xchange-Format File (`.fx`) files.

Table 4. 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 Designer Cadence Allegro
Define I/O assignments in your PCB tool, and then import the assignments into the Pin Planner for validation	Mentor Graphics I/O Designer Cadence Allegro

Figure 5. PCB Tool Integration

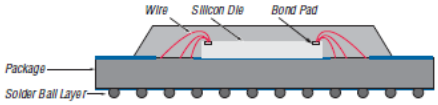
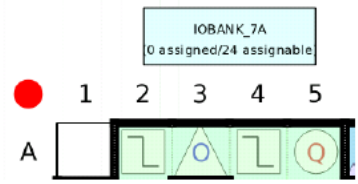
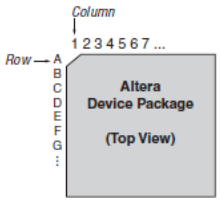
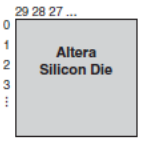
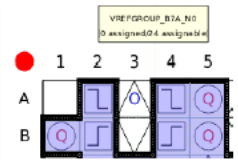


Related Information

Cadence PCB Design Tools Support
In *Intel Quartus Prime Standard Edition User Guide: PCB Design Tools*

2.1.3. Intel Device Terms

The following terms describe Intel 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..	

2.2. Assigning I/O Pins

Use the Pin Planner to visualize, modify, and validate I/O assignments in a graphical representation of the target device. 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 an Intel 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 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 dragging 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:
   a. Right-click an output or bidirectional pin, and click **Board Trace Model**. For differential I/O standards, the board trace model uses a differential pin pair with two symmetrical board trace models.
   b. 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.

2.2.1. 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.

2.2.2. 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.

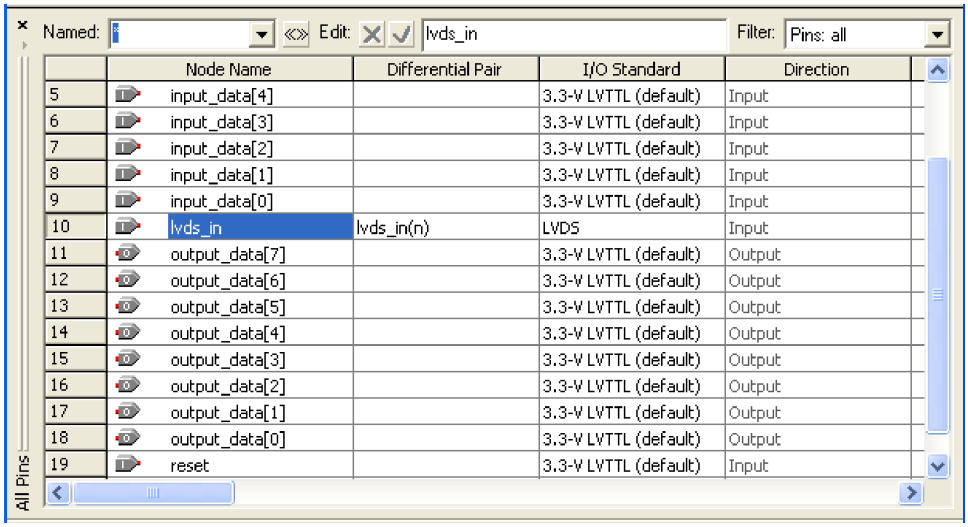
2.2.3. Assigning Differential Pins

When you assign a differential I/O standard to a single-ended top-level pin in your design, the Pin Planner automatically recognizes the negative pin as part of the differential pin pair assignment and creates the negative pin for you. The Intel 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.

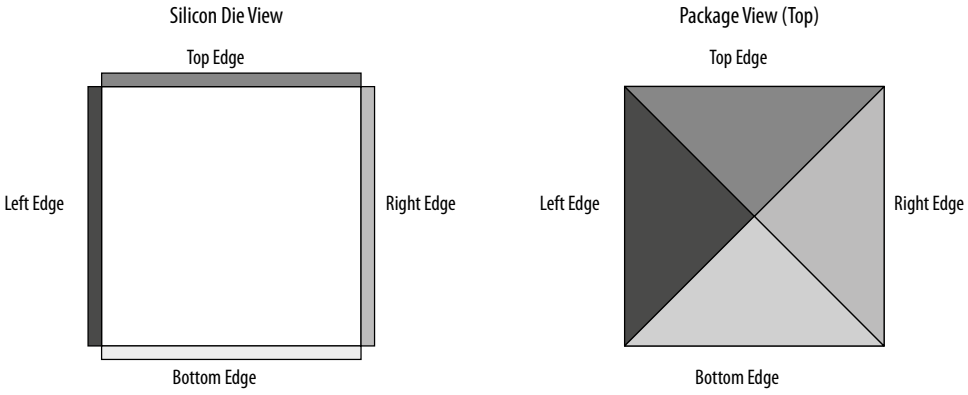
Figure 6. Creating a Differential Pin Pair in the Pin Planner



	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

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 figure shows Intel device package edges.

Figure 7. Die View and Package View of the Four Edges on an Intel Device



2.2.3.1. Overriding I/O Placement Rules on Differential Pins

I/O placement rules ensure that noisy signals do not corrupt neighboring signals. Each device family has predefined I/O placement rules.

I/O placement rules define, for example, the allowed placement of single-ended I/O with respect to differential pins, or how many output and bidirectional pins you can place within a VREF group when using voltage referenced input standards.

Use the **IO_MAXIMUM_TOGGLE_RATE** assignment to override I/O placement rules on pins, such as 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 you intend.

2.2.4. Entering Pin Assignments with Tcl Commands

You can apply pin assignments with Tcl scripts, by either entering individual Tcl commands in the Tcl Console, or creating a `.tcl` script and the typing the following in the command line:

Example 6. Applying Tcl Script Assignments

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

Example 7. Scripted Pin Assignment

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

```
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
In *Intel Quartus Prime Standard Edition User Guide: Scripting*

2.2.5. 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

2.2.5.1. Using Synthesis Attributes

The Intel Quartus Prime software translates synthesis attributes into standard assignments during compilation. These assignments appear in the Pin Planner. Intel Quartus Prime synthesis supports the `chip_pin`, `useioff`, and `altera_attribute` synthesis attributes.

If you modify or delete these assignments in the Pin Planner and then recompile your design, the Pin Planner changes override the synthesis attributes.

Use the `altera_attribute` synthesis attribute to create other pin-related assignments in your HDL code. The `altera_attribute` attribute supports all types of instance assignments. The following examples use the `altera_attribute` attribute to embed **Fast Input Register** logic option assignments and I/O standard assignments in both a Verilog HDL and a VHDL design file.

Example 8. altera_attribute Synthesis Attribute in Verilog HDL

```
input my_pin1 /* synthesis altera_attribute = "-name FAST_INPUT_REGISTER ON; -
name IO_STANDARD \"2.5 V\" " */ ;
```

Example 9. altera_attribute Synthesis Attribute in VHDL

```
entity my_entity is
        port(
                my_pin1: in std_logic
        );
end my_entity;
architecture rtl of my_entity is
begin

attribute altera_attribute : string;
attribute altera_attribute of my_pin1: signal is "-name FAST_INPUT_REGISTER
ON;
-- The architecture body
end rtl;
```

Use the `chip_pin` and `useioff` synthesis attributes to create pin location assignments and to assign **Fast Input Register**, **Fast Output Register**, and **Fast Output Enable Register** logic options. The following examples use the `chip_pin` and `useioff` attributes to embed location and **Fast Input Register** logic option assignments in Verilog HDL and VHDL design files.

Example 10. useioff and chip_pin Synthesis Attributes in VHDL

```
entity my_entity is
        port(
                    my_pin1: in std_logic
        );
end my_entity;

architecture rtl of my_entity is
attribute useioff : boolean;
attribute useioff of my_pin1 : signal is true;
attribute chip_pin : string;
attribute chip_pin of my_pin1 : signal is "C1";
begin -- The architecture body
end rtl;
```

Example 11. chip_pin Synthesis Attribute in Verilog HDL

```
input my_pin1 /* synthesis chip_pin = "C1" useioff = 1 */;
```

2.2.5.2. 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

2.3. Importing and Exporting I/O Pin Assignments

The Intel 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 5. 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 Intel Quartus Prime project with common constraints	• From Intel Quartus Prime project for optimization in a PCB design tool • From Intel Quartus Prime project for spreadsheet analysis or use in scripting assignments • From Intel Quartus Prime project for import into another Intel Quartus Prime project with similar constraints
Command	**Assignments ➤ Import Assignments**	**Assignments ➤ Export Assignments**
File formats	`.qsf`, `.esf`, `.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

2.3.1. 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 Intel Quartus Prime software requires only the **.fx** to import pin assignments from I/O Designer.

Table 6. 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 Intel Devices
- PCB Design Tools Support
  In *Intel Quartus Prime Standard Edition User Guide: PCB Design Tools*

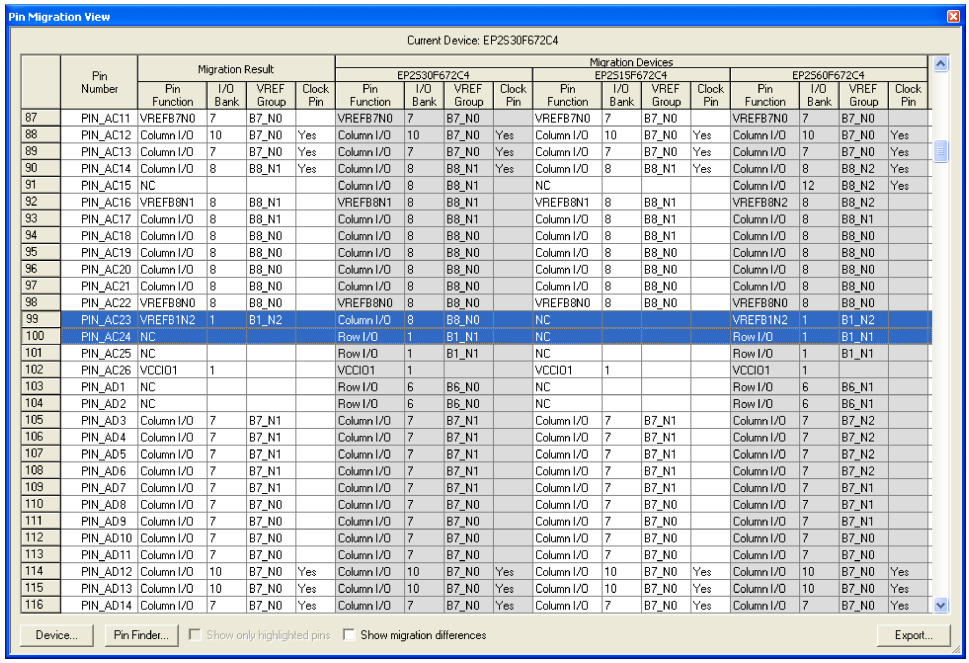
2.3.2. Migrating Assignments to Another Target Device

Click **View ➤ Pin Migration Window** to verify whether pin assignments are compatible with migration to a different Intel 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.

The Intel 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 8. Device Migration Compatibility (AC24 does not exist in migration device)



	Pin Number	Migration Result				EP2S30F672C4				EP2S15F672C4				EP2S60F672C4			
		Pin Function	I/O Bank	VREF Group	Clock Pin	Pin Function	I/O Bank	VREF Group	Clock Pin	Pin Function	I/O Bank	VREF Group	Clock Pin	Pin Function	I/O Bank	VREF Group	Clock Pin
87	PIN_AC11	VREFB7N0	7	B7_N0		VREFB7N0	7	B7_N0		VREFB7N0	7	B7_N0		VREFB7N0	7	B7_N0	
88	PIN_AC12	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes
89	PIN_AC13	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes
90	PIN_AC14	Column I/O	8	B8_N1	Yes	Column I/O	8	B8_N1	Yes	Column I/O	8	B8_N1	Yes	Column I/O	8	B8_N2	Yes
91	PIN_AC15	NC				Column I/O	8	B8_N1		NC				Column I/O	12	B8_N2	Yes
92	PIN_AC16	VREFB8N1	8	B8_N1		VREFB8N1	8	B8_N1		VREFB8N1	8	B8_N1		VREFB8N2	8	B8_N2	
93	PIN_AC17	Column I/O	8	B8_N1		Column I/O	8	B8_N1		Column I/O	8	B8_N1		Column I/O	8	B8_N1	
94	PIN_AC18	Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N1		Column I/O	8	B8_N0	
95	PIN_AC19	Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0	
96	PIN_AC20	Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0	
97	PIN_AC21	Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0		Column I/O	8	B8_N0	
98	PIN_AC22	VREFB8N0	8	B8_N0		VREFB8N0	8	B8_N0		VREFB8N0	8	B8_N0		VREFB8N0	8	B8_N0	
99	PIN_AC23	VREFB1N2	1	B1_N2		Column I/O	8	B8_N0		NC				VREFB1N2	1	B1_N2	
100	PIN_AC24	NC				Row I/O	1	B1_N1		NC				Row I/O	1	B1_N1	
101	PIN_AC25	NC				Row I/O	1	B1_N1		NC				Row I/O	1	B1_N1	
102	PIN_AC26	VCCIO1	1			VCCIO1	1			VCCIO1	1			VCCIO1	1		
103	PIN_AD1	NC				Row I/O	6	B6_N0		NC				Row I/O	6	B6_N1	
104	PIN_AD2	NC				Row I/O	6	B6_N0		NC				Row I/O	6	B6_N1	
105	PIN_AD3	Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N2	
106	PIN_AD4	Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N2	
107	PIN_AD5	Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N2	
108	PIN_AD6	Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N2	
109	PIN_AD7	Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1		Column I/O	7	B7_N1	
110	PIN_AD8	Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N1	
111	PIN_AD9	Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N1	
112	PIN_AD10	Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0	
113	PIN_AD11	Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0		Column I/O	7	B7_N0	
114	PIN_AD12	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes
115	PIN_AD13	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes	Column I/O	10	B7_N0	Yes
116	PIN_AD14	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes	Column I/O	7	B7_N0	Yes

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 displays the voltage reference. Even if you do not use that pin for a port connection in the 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 Intel 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 the original design because the pins are not physically connected to the smaller die.

Related Information

AN90: SameFrame PinOut Design for FineLine BGA Packages

2.4. Validating Pin Assignments

The Intel 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 7. 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**

2.4.1. I/O Assignment Validation Rules

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

Table 8. 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
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

continued...

Rule	Description	HDL Required?
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 9. Signal Switching Noise Rules

Rule	Description	HDL Required?
I/O bank cannot 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
Single-ended output must 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

2.4.2. Checking I/O Pin Assignments in Real-Time

Live I/O check validates I/O assignments against basic I/O buffer rules in real time. The Pin Planner immediately reports warnings or errors about assignments as you enter them. The Live I/O Check Status window displays the total number of errors and warnings. Use this analysis to quickly correct basic errors before proceeding. Run full I/O assignment analysis when you are ready to validate pin assignments against the complete set of I/O system rules.

Note: Live I/O check is supported only for Arria II, Cyclone® IV, MAX® II, and Stratix IV device families.

Live I/O check validates against the following basic I/O buffer rules:

- V_{CCIO} and V_{REF} voltage compatibility rules
- Electromigration (current density) rules
- Simultaneous Switching Output (SSO) rules
- I/O property compatibility rules, such as drive strength compatibility, I/O standard compatibility, `PCI_IO` clamp diode compatibility, and I/O direction compatibility
- Illegal location assignments:
  - An I/O bank or VREF group with no available pins
  - The negative pin of a differential pair if the positive pin of the differential pair is assigned with a node name with a differential I/O standard
  - Pin locations that do not support the I/O standard assigned to the selected node name
  - For HSTL- and SSTL-type I/O standards, VREF groups of a different V_{REF} voltage than the selected node name.

2.4.3. 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.

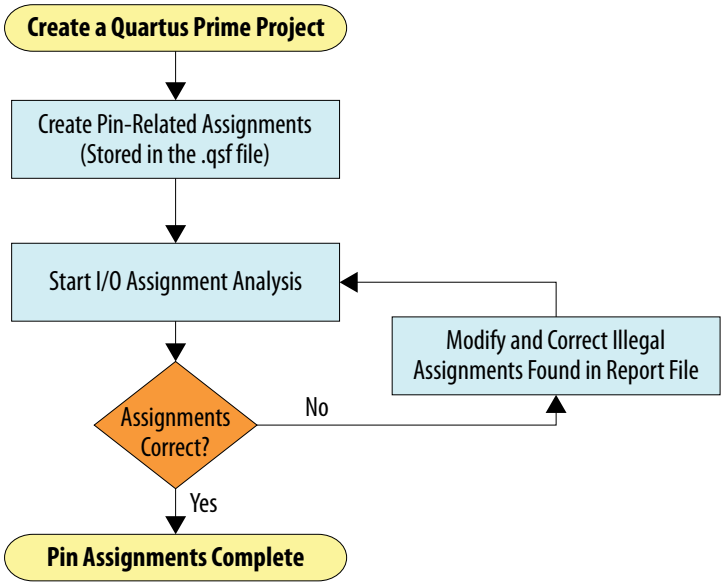
The Fitter assigns pins to accommodate your constraints. For example, if you assign an edge location to a group of LVDS pins, the Fitter assigns pin locations for each LVDS pin in the specified edge location and then performs legality checks. To display the Fitter-placed pins, click **Show Fitter Placements** 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, can assign the **Termination** to the pin. You cannot assign drive strength or slew rate settings when a pin has an OCT assignment.

2.4.3.1. 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 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 9. Assigning and Analyzing Pin-Outs without Design Files

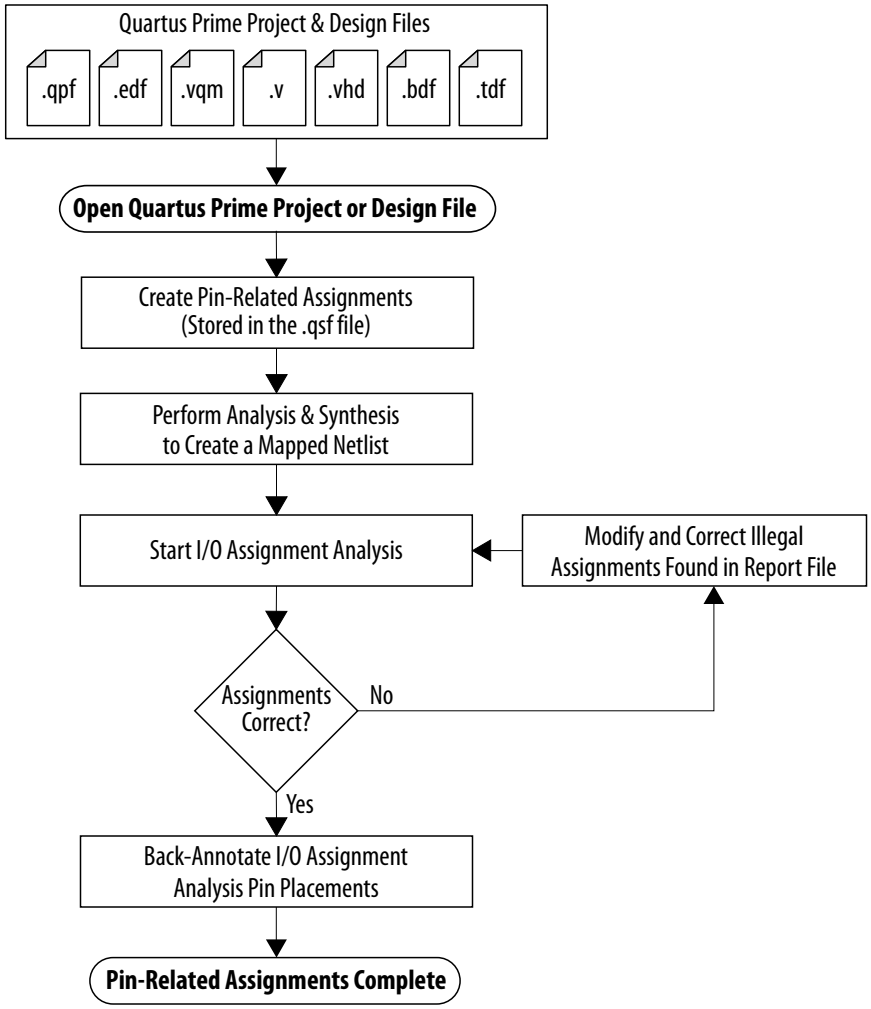


You must reserve all pins you intend to use as I/O pins, so that the Fitter can determine each pin type. After performing I/O assignment analysis, correct any errors reported by the Fitter and rerun I/O assignment analysis until all errors are corrected. A complete I/O assignment analysis requires all design files.

2.4.3.2. I/O Assignment Analysis With Design Files

I/O assignment analysis allows you 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 design, 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 10. I/O Assignment Analysis Flow

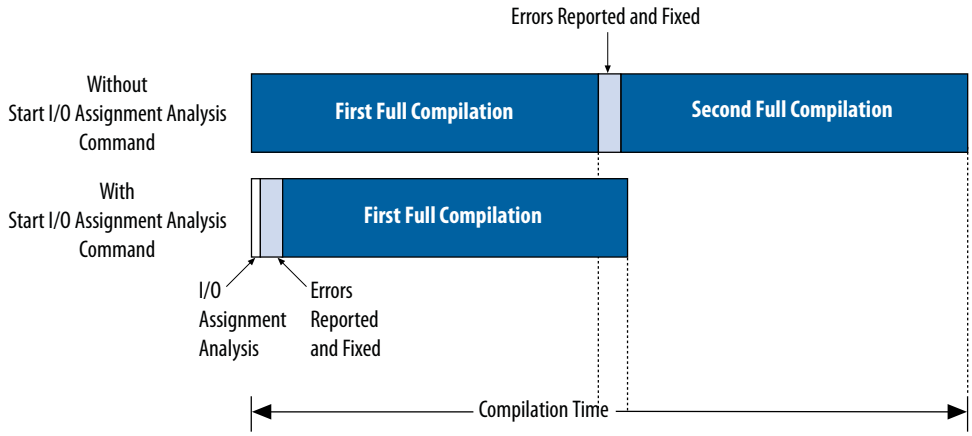


Even if I/O assignment analysis passes on incomplete design files, you may still encounter errors during full compilation. For example, you can assign a clock to a user I/O pin instead of assigning to a dedicated clock pin, or design the clock to drive a PLL that you have not yet instantiated in the design. This issues occur 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 11. I/O Assignment Analysis Reduces Compilation Time



2.4.3.3. Overriding Default I/O Pin Analysis

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

Each device contains VREF pins, each supporting one or more 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.

Related Information

Using the Timing Analyzer
In *Intel Quartus Prime Standard Edition User Guide: Timing Analyzer*

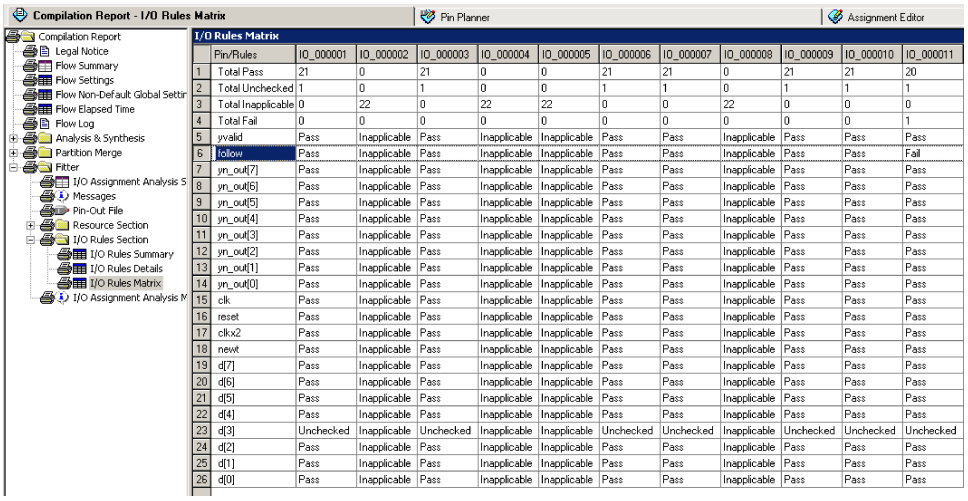
2.4.4. Understanding I/O Analysis Reports

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 were not checked. A severity rating indicates the rule's importance for effective analysis. "Inapplicable" rules do not apply to the target device family.

Figure 12. I/O Rules Matrix



I/O Rules Matrix

	Pin/Rules	IO_000001	IO_000002	IO_000003	IO_000004	IO_000005	IO_000006	IO_000007	IO_000008	IO_000009	IO_000010	IO_000011
1	Total Pass	21	0	21	0	0	21	21	0	21	21	20
2	Total Unchecked	1	0	1	0	0	1	1	0	1	1	1
3	Total Inapplicable	0	22	0	22	22	0	0	22	0	0	0
4	Total Fail	0	0	0	0	0	0	0	0	0	0	1
5	yvalid	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
6	follow	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Fail
7	yn_out[7]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
8	yn_out[6]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
9	yn_out[5]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
10	yn_out[4]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
11	yn_out[3]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
12	yn_out[2]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
13	yn_out[1]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
14	yn_out[0]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
15	clk	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
16	reset	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
17	clkx2	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
18	newt	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
19	d[7]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
20	d[6]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
21	d[5]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
22	d[4]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
23	d[3]	Unchecked	Inapplicable	Unchecked	Inapplicable	Inapplicable	Unchecked	Unchecked	Inapplicable	Unchecked	Unchecked	Unchecked
24	d[2]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
25	d[1]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass
26	d[0]	Pass	Inapplicable	Pass	Inapplicable	Inapplicable	Pass	Pass	Inapplicable	Pass	Pass	Pass

2.5. 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 ➤ 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 Intel Quartus Prime software supports three different methods of I/O timing analysis:

Table 10. 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. Timing Analyzer 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. Timing Analyzer reports tCO to an I/O pin using a default or user-specified value for a capacitive load.
Full board routing simulation	Use Intel-provided or Intel Quartus Prime software-generated IBIS or HSPICE I/O models for simulation in Mentor Graphics HyperLynx* and Synopsys HSPICE.

Note: Advanced I/O timing analysis is supported only for .28nm and larger device families. For devices that support advanced I/O timing, it is the default method of I/O timing analysis. For all other devices, you must use a default or user-specified capacitive load assignment to determine t_{CO} and power measurements.

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 Intel Quartus Prime software and integrating those models into HyperLynx and HSPICE simulations, refer to the *Signal Integrity Analysis with Third Party Tools* chapter.

Related Information

- Signal Integrity Analysis with Third-Party Tools
  In *Intel Quartus Prime Standard Edition User Guide: PCB Design Tools*
- Literature and Technical Documentation
- Intel Signal & Power Integrity Center

2.5.1. 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.

2.5.1.1. Board Trace Models

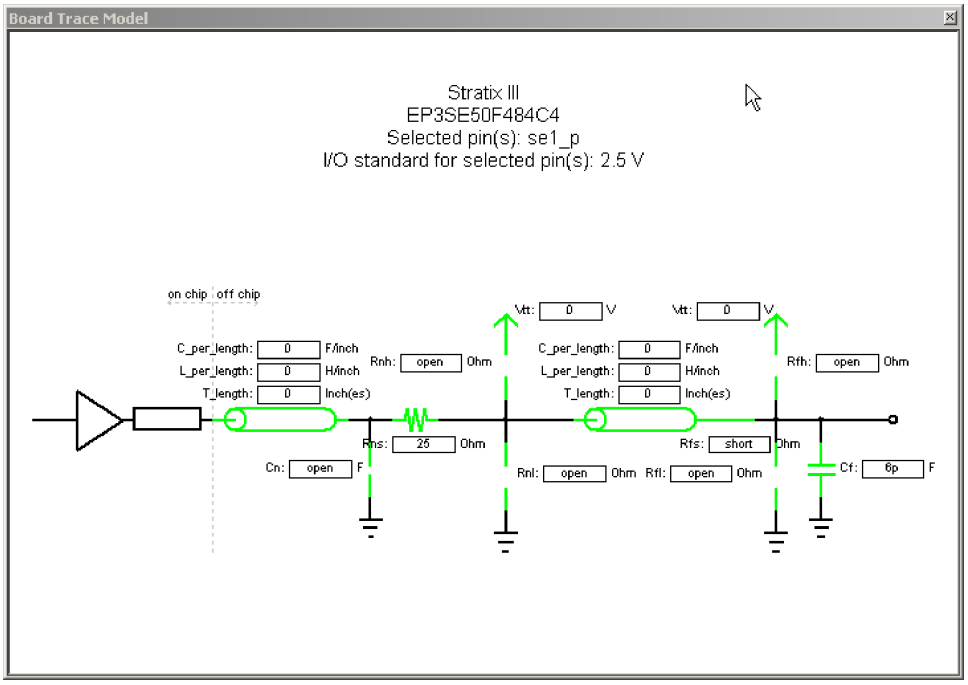
The Intel Quartus Prime software provides board trace model templates for various I/O standards.

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.

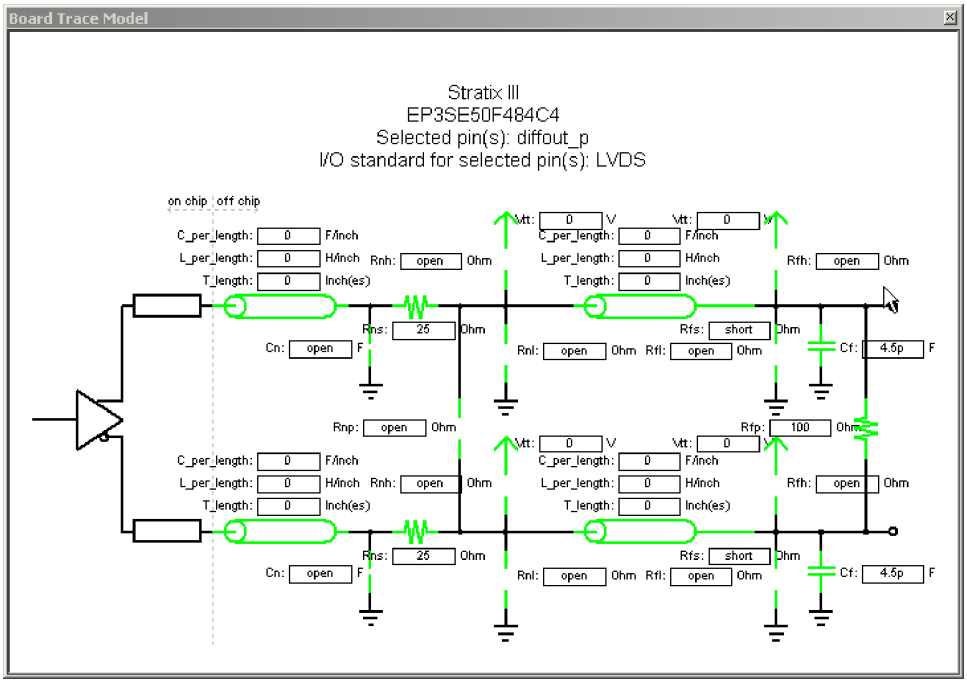
Figure 13. 2.5-V I/O Standard Board Trace Model



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 Intel Quartus Prime software models of transmission lines do not consider transmission-line resistance (lossless models). You only need to specify distributed inductance (L) and capacitance (C) values on a per-inch basis, which you can obtain from the PCB vendor or manufacturer, the CAD Design tool, or a signal integrity tool, such as the Mentor Graphics HyperLynx software.

Figure 14. LVDS Differential Board Trace Model



2.5.1.2. 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 uses 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 ➤ 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 12. 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 se1_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 se1_p
```

2.5.1.3. 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.

2.5.1.4. Specifying Near-End vs Far-End I/O Timing Analysis

You can select a near-end 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.

With far-end I/O timing analysis, the 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.

2.5.1.5. Advanced I/O Timing Analysis Reports

The following reports show advanced I/O timing analysis information:

Table 11. Advanced I/O Timing Reports

I/O Timing Report	Description
Timing Analyzer 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 Timing 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 ➤ Timing Analyzer**.

Related Information

Using the Timing Analyzer
In *Intel Quartus Prime Standard Edition User Guide: Timing Analyzer*

2.5.2. Adjusting I/O Timing and Power with Capacitive Loading

When calculating t_{CO} and power for output and bidirectional pins, the Timing Analyzer and the 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 Intel 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.

2.6. 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, Logic Lock (Standard) 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.

2.7. Analyzing Simultaneous Switching Noise

Click **Processing > Start > Start SSN Analyzer** to estimate the voltage noise for each pin in the design. The simultaneous switching noise (SSN) analysis accounts for the pin placement, I/O standard, board trace, output enable group, timing constraint, and PCB characteristics that you specify. The analysis produces a voltage noise estimate for each pin in the design. View the SSN results in the Pin Planner and adjust your I/O assignments to optimize signal integrity.

2.8. Scripting API

The Intel Quartus Prime software allows you to access I/O management functions through Tcl commands, rather than with the GUI. For detailed information about scripting command options and Tcl API packages, type the following at a system command prompt to view the Tcl API Help browser:

```
quartus_sh --qhelp
```

Related Information

- Tcl Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*
- Command Line Scripting
  In *Intel Quartus Prime Standard Edition User Guide: Scripting*

2.8.1. 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_map <project name>
```

2.8.2. 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.

2.8.3. 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`

2.8.4. Exclusive I/O Group

The following Tcl command creates an exclusive I/O group assignment:

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

2.8.5. Slew Rate and Current Strength

Use the following Tcl commands to create a 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

Package Information Datasheet for Mature Altera Devices

2.9. Managing Device I/O Pins Revision History

The following table shows the revision history for this chapter:

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.
2017.11.06	17.1.0	• Revised topic: I/O Planning Overview. • Revised topic: Basic I/O Planning Flow with the Pin Planner and renamed to Basic I/O Planning Flow with the Pin Planner.
2015.11.02	15.1.0	• 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
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

Documentation Archive

For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

A. Intel Quartus Prime Standard Edition User Guides

Refer to the following user guides for comprehensive information on all phases of the Intel Quartus Prime Standard Edition FPGA design flow.

Related Information

- Intel Quartus Prime Standard Edition User Guide: Getting Started
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Standard Edition software, including managing Intel Quartus Prime Standard Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- Intel Quartus Prime Standard Edition User Guide: Platform Designer
  Describes creating and optimizing systems using Platform Designer (Standard), a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer (Standard) automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Intel Quartus Prime Standard Edition User Guide: Design Recommendations
  Describes best design practices for designing FPGAs with the Intel Quartus Prime Standard Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Standard Edition synthesis optimally implements your design in hardware.
- Intel Quartus Prime Standard Edition User Guide: Design Compilation
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Standard Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- Intel Quartus Prime Standard Edition User Guide: Design Optimization
  Describes Intel Quartus Prime Standard Edition settings, tools, and techniques that you can use to achieve the highest design performance in Intel FPGAs. Techniques include optimizing the design netlist, addressing critical chains that limit retiming and timing closure, and optimization of device resource usage.
- Intel Quartus Prime Standard Edition User Guide: Programmer
  Describes operation of the Intel Quartus Prime Standard Edition Programmer, which allows you to configure Intel FPGA devices, and program CPLD and configuration devices, via connection with an Intel FPGA download cable.
- Intel Quartus Prime Standard Edition User Guide: Partial Reconfiguration
  Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.

- Intel Quartus Prime Standard Edition User Guide: Third-party Simulation
  Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Mentor Graphics, and Synopsys that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- Intel Quartus Prime Standard Edition User Guide: Third-party Synthesis
  Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- Intel Quartus Prime Standard Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Standard Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or "tapping") signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Standard Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Standard Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- Intel Quartus Prime Standard Edition User Guide: Design Constraints
  Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Standard Edition User Guide: PCB Design Tools
  Describes support for optional third-party PCB design tools by Mentor Graphics and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- Intel Quartus Prime Standard Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Standard Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.