



Intel[®] Quartus[®] Prime Standard Edition Handbook Volume 2

Design Implementation and Optimization

Updated for Intel[®] Quartus[®] Prime Design Suite: **17.1**



[Subscribe](#)

[Send Feedback](#)

QPS5V2 | 2017.11.06

Latest document on the web: [PDF](#) | [HTML](#)



Contents

1 Constraining Designs.....	11
1.1 Constraining Designs with Intel Quartus Prime Tools.....	11
1.1.1 Global Constraints and Assignments.....	11
1.1.2 Node, Entity, and Instance-Level Constraints.....	12
1.1.3 Probing Between Components of the Intel Quartus Prime GUI.....	14
1.1.4 Specifying Individual Timing Constraints.....	14
1.2 Constraining Designs with Tcl Scripts.....	16
1.2.1 Generating Intel Quartus Prime Settings Files.....	16
1.2.2 Timing Analysis with .sdc Files and Tcl Scripts.....	18
1.2.3 Using Tcl-only Script Flows.....	19
1.3 A Fully Iterative Scripted Flow.....	22
1.4 Document Revision History.....	22
2 Managing Device I/O Pins.....	24
2.1 I/O Planning Overview.....	25
2.1.1 Basic I/O Planning Flow.....	25
2.1.2 Integrating PCB Design Tools.....	26
2.1.3 Intel Device Terms.....	27
2.2 Assigning I/O Pins.....	27
2.2.1 Assigning to Exclusive Pin Groups.....	28
2.2.2 Assigning Slew Rate and Drive Strength.....	28
2.2.3 Assigning Differential Pins.....	28
2.2.4 Entering Pin Assignments with Tcl Commands.....	30
2.2.5 Entering Pin Assignments in HDL Code.....	30
2.3 Importing and Exporting I/O Pin Assignments.....	32
2.3.1 Importing and Exporting for PCB Tools.....	32
2.3.2 Migrating Assignments to Another Target Device.....	33
2.4 Validating Pin Assignments.....	34
2.4.1 I/O Assignment Validation Rules.....	34
2.4.2 Checking I/O Pin Assignments in Real-Time.....	35
2.4.3 Running I/O Assignment Analysis.....	36
2.4.4 Understanding I/O Analysis Reports.....	40
2.5 Verifying I/O Timing.....	40
2.5.1 Running Advanced I/O Timing.....	41
2.5.2 Adjusting I/O Timing and Power with Capacitive Loading.....	45
2.6 Viewing Routing and Timing Delays.....	45
2.7 Analyzing Simultaneous Switching Noise.....	45
2.8 Scripting API.....	45
2.8.1 Generate Mapped Netlist.....	45
2.8.2 Reserve Pins.....	46
2.8.3 Set Location.....	46
2.8.4 Exclusive I/O Group.....	46
2.8.5 Slew Rate and Current Strength.....	46
2.9 Document Revision History.....	47
3 Simultaneous Switching Noise (SSN) Analysis and Optimizations.....	48
3.1 Simultaneous Switching Noise (SSN) Analysis and Optimizations.....	48
3.2 Definitions.....	48



3.3 Understanding SSN.....	49
3.4 SSN Estimation Tools.....	51
3.5 SSN Analysis Overview.....	52
3.5.1 Performing Early Pin-Out SSN Analysis.....	53
3.5.2 Performing Final Pin-Out SSN Analysis.....	54
3.6 Design Factors Affecting SSN Results.....	54
3.7 Optimizing Your Design for SSN Analysis.....	54
3.7.1 Optimizing Pin Placements for Signal Integrity.....	55
3.7.2 Specifying Board Trace Model Settings.....	56
3.7.3 Defining PCB Layers and PCB Layer Thickness.....	57
3.7.4 Specifying Signal Breakout Layers.....	59
3.7.5 Creating I/O Assignments.....	60
3.7.6 Decreasing Pessimism in SSN Analysis.....	60
3.7.7 Excluding Pins as Aggressor Signals.....	61
3.8 Performing SSN Analysis and Viewing Results.....	61
3.8.1 Understanding the SSN Reports.....	61
3.8.2 Viewing SSN Analysis Results in the Pin Planner.....	62
3.9 Decreasing Processing Time for SSN Analysis.....	63
3.10 Scripting Support.....	63
3.10.1 Optimizing Pin Placements for Signal Integrity.....	64
3.10.2 Defining PCB Layers and PCB Layer Thickness.....	64
3.10.3 Specifying Signal Breakout Layers.....	65
3.10.4 Decreasing Pessimism in SSN Analysis.....	65
3.10.5 Performing SSN Analysis.....	65
3.11 Document Revision History.....	66
4 Command Line Scripting.....	67
4.1 Benefits of Command-Line Executables.....	67
4.2 Introductory Example.....	67
4.3 Command-Line Scripting Help.....	68
4.4 Project Settings with Command-Line Options.....	69
4.4.1 Option Precedence.....	69
4.5 Compilation with quartus_sh --flow.....	71
4.6 Text-Based Report Files.....	72
4.7 Using Command-Line Executables in Scripts.....	73
4.8 Common Scripting Examples.....	74
4.8.1 Create a Project and Apply Constraints.....	74
4.8.2 Check Design File Syntax.....	75
4.8.3 Create a Project and Synthesize a Netlist Using Netlist Optimizations.....	75
4.8.4 Archive and Restore Projects.....	76
4.8.5 Perform I/O Assignment Analysis.....	76
4.8.6 Update Memory Contents Without Recompiling.....	76
4.8.7 Create a Compressed Configuration File.....	77
4.8.8 Fit a Design as Quickly as Possible.....	77
4.8.9 Fit a Design Using Multiple Seeds.....	78
4.9 The QFlow Script.....	78
4.10 Document Revision History.....	79
5 Tcl Scripting.....	81
5.1 Tcl Scripting.....	81
5.2 Tool Command Language.....	81



- 5.3 Intel Quartus Prime Tcl Packages..... 82
 - 5.3.1 Loading Packages.....83
- 5.4 Intel Quartus Prime Tcl API Help..... 83
 - 5.4.1 Command-Line Options.....85
 - 5.4.2 The Intel Quartus Prime Tcl Console Window.....86
- 5.5 End-to-End Design Flows..... 87
- 5.6 Creating Projects and Making Assignments..... 87
- 5.7 Compiling Designs..... 88
 - 5.7.1 The flow Package..... 88
 - 5.7.2 Compile All Revisions.....88
- 5.8 Reporting.....89
 - 5.8.1 Saving Report Data in csv Format..... 89
- 5.9 Timing Analysis..... 90
- 5.10 Automating Script Execution.....90
 - 5.10.1 Execution Example..... 91
 - 5.10.2 Controlling Processing.....92
 - 5.10.3 Displaying Messages.....92
- 5.11 Other Scripting Features..... 92
 - 5.11.1 Natural Bus Naming.....92
 - 5.11.2 Short Option Names..... 93
 - 5.11.3 Collection Commands..... 93
 - 5.11.4 The post_message Command..... 94
 - 5.11.5 Accessing Command-Line Arguments.....94
 - 5.11.6 The quartus() Array.....96
- 5.12 The Intel Quartus Prime Tcl Shell in Interactive Mode Example..... 96
- 5.13 The tclsh Shell.....97
- 5.14 Tcl Scripting Basics.....97
 - 5.14.1 Hello World Example.....98
 - 5.14.2 Variables.....98
 - 5.14.3 Substitutions..... 98
 - 5.14.4 Arithmetic..... 99
 - 5.14.5 Lists..... 99
 - 5.14.6 Arrays..... 99
 - 5.14.7 Control Structures..... 100
 - 5.14.8 Procedures..... 101
 - 5.14.9 File I/O..... 102
 - 5.14.10 Syntax and Comments..... 102
 - 5.14.11 External References..... 103
- 5.15 Document Revision History..... 103
- 6 Signal Integrity Analysis with Third-Party Tools..... 105**
 - 6.1 Signal Integrity Analysis with Third-Party Tools..... 105
 - 6.1.1 Signal Integrity Simulations with HSPICE and IBIS Models..... 106
 - 6.2 I/O Model Selection: IBIS or HSPICE..... 107
 - 6.3 FPGA to Board Signal Integrity Analysis Flow..... 107
 - 6.3.1 Create I/O and Board Trace Model Assignments.....110
 - 6.3.2 Output File Generation..... 110
 - 6.3.3 Customize the Output Files..... 110
 - 6.3.4 Set Up and Run Simulations in Third-Party Tools.....111
 - 6.3.5 Interpret Simulation Results..... 111
 - 6.4 Simulation with IBIS Models..... 111



6.4.1 Elements of an IBIS Model.....	112
6.4.2 Creating Accurate IBIS Models.....	112
6.4.3 Design Simulation Using the Mentor Graphics HyperLynx Software.....	114
6.4.4 Configuring LineSim to Use Intel IBIS Models.....	116
6.4.5 Integrating Intel IBIS Models into LineSim Simulations.....	118
6.4.6 Running and Interpreting LineSim Simulations.....	119
6.5 Simulation with HSPICE Models.....	121
6.5.1 Supported Devices and Signaling.....	121
6.5.2 Accessing HSPICE Simulation Kits.....	121
6.5.3 The Double Counting Problem in HSPICE Simulations.....	122
6.5.4 HSPICE Writer Tool Flow.....	124
6.5.5 Running an HSPICE Simulation.....	126
6.5.6 Interpreting the Results of an Output Simulation.....	127
6.5.7 Interpreting the Results of an Input Simulation.....	127
6.5.8 Viewing and Interpreting Tabular Simulation Results.....	128
6.5.9 Viewing Graphical Simulation Results.....	128
6.5.10 Making Design Adjustments Based on HSPICE Simulations.....	129
6.5.11 Sample Input for I/O HSPICE Simulation Deck.....	131
6.5.12 Sample Output for I/O HSPICE Simulation Deck.....	135
6.5.13 Advanced Topics.....	141
6.6 Document Revision History.....	142
7 Mentor Graphics PCB Design Tools Support.....	144
7.1 FPGA-to-PCB Design Flow.....	145
7.2 Integrating with I/O Designer.....	147
7.2.1 Generating Pin Assignment Files.....	148
7.2.2 I/O Designer Settings.....	149
7.2.3 Transferring I/O Assignments.....	150
7.2.4 Updating I/O Designer with Intel Quartus Prime Pin Assignments.....	152
7.2.5 Updating Intel Quartus Prime with I/O Designer Pin Assignments.....	153
7.2.6 Generating Schematic Symbols in I/O Designer.....	154
7.2.7 Exporting Schematic Symbols to DxDesigner.....	155
7.3 Integrating with DxDesigner.....	155
7.3.1 DxDesigner Project Settings.....	156
7.3.2 Creating Schematic Symbols in DxDesigner.....	156
7.4 Analyzing FPGA Simultaneous Switching Noise (SSN).....	157
7.5 Scripting API.....	157
7.6 Document Revision History.....	157
8 Cadence PCB Design Tools Support.....	159
8.1 Cadence PCB Design Tools Support.....	159
8.2 Product Comparison.....	160
8.3 FPGA-to-PCB Design Flow.....	160
8.3.1 Integrating Intel FPGA Design.....	162
8.3.2 Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA.....	162
8.4 Setting Up the Intel Quartus Prime Software.....	162
8.4.1 Generating a .pin File.....	163
8.5 FPGA-to-Board Integration with the Cadence Allegro Design Entry HDL Software.....	163
8.5.1 Creating Symbols.....	164
8.5.2 Instantiating the Symbol in the Cadence Allegro Design Entry HDL Software...	169
8.6 FPGA-to-Board Integration with Cadence Allegro Design Entry CIS Software.....	170



- 8.6.1 Creating a Cadence Allegro Design Entry CIS Project.....171
- 8.6.2 Generating a Part..... 171
- 8.6.3 Generating Schematic Symbol.....172
- 8.6.4 Splitting a Part.....172
- 8.6.5 Instantiating a Symbol in a Design Entry CIS Schematic..... 174
- 8.6.6 Intel Libraries for the Cadence Allegro Design Entry CIS Software..... 174
- 8.7 Document Revision History..... 176
- 9 Reviewing Printed Circuit Board Schematics with the Intel Quartus Prime Software... 177**
 - 9.1 Reviewing Intel Quartus Prime Software Settings..... 177
 - 9.1.1 Device and Pins Options Dialog Box Settings..... 178
 - 9.2 Reviewing Device Pin-Out Information in the Fitter Report..... 179
 - 9.3 Reviewing Compilation Error and Warning Messages..... 181
 - 9.4 Using Additional Intel Quartus Prime Software Features.....181
 - 9.5 Using Additional Intel Quartus Prime Software Tools.....182
 - 9.5.1 Pin Planner.....182
 - 9.5.2 SSN Analyzer.....182
 - 9.6 Document Revision History..... 182
- 10 Design Optimization Overview..... 184**
 - 10.1 Initial Compilation: Required Settings..... 184
 - 10.1.1 Device Settings..... 184
 - 10.1.2 Device Migration Settings..... 184
 - 10.1.3 I/O Assignments..... 185
 - 10.1.4 Timing Requirement Settings..... 185
 - 10.1.5 Partitions and Floorplan Assignments for Incremental Compilation.....186
 - 10.2 Physical Implementation..... 187
 - 10.2.1 Trade-Offs and Limitations..... 187
 - 10.2.2 Preserving Results and Enabling Teamwork..... 187
 - 10.2.3 Reducing Area.....188
 - 10.2.4 Reducing Critical Path Delay.....188
 - 10.2.5 Reducing Power Consumption.....189
 - 10.2.6 Reducing Runtime..... 189
 - 10.3 Using Intel Quartus Prime Tools.....189
 - 10.3.1 Design Analysis 190
 - 10.3.2 Advisors.....190
 - 10.3.3 Design Space Explorer II..... 190
 - 10.4 Document Revision History..... 191
- 11 Reducing Compilation Time..... 192**
 - 11.1 Compilation Time Advisor..... 192
 - 11.2 Strategies to Reduce the Overall Compilation Time.....192
 - 11.2.1 Running Rapid Recompile..... 192
 - 11.2.2 Enabling Multi-Processor Compilation..... 193
 - 11.2.3 Using Incremental Compilation..... 194
 - 11.2.4 Using Block-Based Compilation.....195
 - 11.3 Reducing Synthesis Time and Synthesis Netlist Optimization Time.....195
 - 11.3.1 Settings to Reduce Synthesis Time and Synthesis Netlist Optimization Time.. 196
 - 11.3.2 Use Appropriate Coding Style to Reduce Synthesis Time..... 196
 - 11.4 Reducing Placement Time..... 196
 - 11.4.1 Fitter Effort Setting.....196
 - 11.4.2 Placement Effort Multiplier Settings.....197



11.4.3 Physical Synthesis Effort Settings.....	197
11.4.4 Preserving Placement with Incremental Compilation.....	197
11.5 Reducing Routing Time.....	197
11.5.1 Identifying Routing Congestion with the Chip Planner.....	198
11.6 Reducing Static Timing Analysis Time.....	199
11.7 Setting Process Priority.....	199
11.8 Document Revision History.....	200
12 Timing Closure and Optimization.....	201
12.1 About Timing Closure and Optimization.....	201
12.2 Optimize Multi-Corner Timing.....	201
12.3 Critical Paths.....	202
12.3.1 Viewing Critical Paths.....	202
12.4 Design Evaluation for Timing Closure.....	202
12.4.1 Review Compilation Results.....	202
12.4.2 Review Details of Timing Paths.....	211
12.4.3 Adjusting and Recompiling.....	214
12.5 Design Analysis.....	216
12.5.1 Ignored Timing Constraints.....	216
12.5.2 I/O Timing.....	216
12.5.3 Register-to-Register Timing Analysis.....	217
12.6 Timing Optimization.....	221
12.6.1 Displaying Timing Closure Recommendations for Failing Paths.....	221
12.6.2 Timing Optimization Advisor.....	222
12.6.3 Optional Fitter Settings.....	223
12.6.4 I/O Timing Optimization Techniques.....	225
12.6.5 Register-to-Register Timing Optimization Techniques.....	229
12.6.6 Logic Lock (Standard) Assignments.....	235
12.6.7 Location Assignments.....	237
12.6.8 Metastability Analysis and Optimization Techniques.....	237
12.7 Periphery to Core Register Placement and Routing Optimization	238
12.7.1 Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box.....	239
12.7.2 Setting Periphery to Core Optimizations in the Assignment Editor.....	239
12.7.3 Viewing Periphery to Core Optimizations in the Fitter Report.....	240
12.8 Scripting Support.....	241
12.8.1 Initial Compilation Settings.....	241
12.8.2 Resource Utilization Optimization Techniques.....	242
12.8.3 I/O Timing Optimization Techniques	243
12.8.4 Register-to-Register Timing Optimization Techniques.....	243
12.9 Document Revision History.....	244
13 Power Optimization.....	247
13.1 Power Dissipation.....	247
13.2 Design Space Explorer II.....	249
13.3 Power-Driven Compilation.....	250
13.3.1 Power-Driven Synthesis.....	250
13.3.2 Power-Driven Fitter.....	253
13.3.3 Area-Driven Synthesis.....	254
13.3.4 Gate-Level Register Retiming.....	254
13.4 Design Guidelines.....	255
13.4.1 Clock Power Management.....	255



- 13.4.2 Pipelining and Retiming..... 260
- 13.4.3 Architectural Optimization.....262
- 13.4.4 I/O Power Guidelines..... 262
- 13.4.5 Dynamically Controlled On-Chip Terminations.....263
- 13.4.6 Power Optimization Advisor..... 264
- 13.5 Document Revision History..... 265
- 14 Area Optimization..... 267**
 - 14.1 Resource Utilization..... 267
 - 14.2 Optimizing Resource Utilization..... 268
 - 14.2.1 Using the Resource Optimization Advisor..... 268
 - 14.2.2 Resource Utilization Issues Overview.....269
 - 14.2.3 I/O Pin Utilization or Placement..... 269
 - 14.2.4 Logic Utilization or Placement.....270
 - 14.2.5 Routing..... 274
 - 14.3 Scripting Support..... 277
 - 14.3.1 Initial Compilation Settings..... 277
 - 14.3.2 Resource Utilization Optimization Techniques.....277
 - 14.4 Document Revision History..... 279
- 15 Analyzing and Optimizing the Design Floorplan..... 280**
 - 15.1 Design Floorplan Analysis in the Chip Planner.....280
 - 15.1.1 Starting the Chip Planner.....281
 - 15.1.2 Chip Planner GUI Components.....281
 - 15.1.3 Viewing Architecture-Specific Design Information.....283
 - 15.1.4 Viewing Available Clock Networks in the Device..... 284
 - 15.1.5 Viewing Routing Congestion..... 285
 - 15.1.6 Viewing I/O Banks..... 285
 - 15.1.7 Viewing High-Speed Serial Interfaces (HSSI).....285
 - 15.1.8 Generating Fan-In and Fan-Out Connections..... 286
 - 15.1.9 Generating Immediate Fan-In and Fan-Out Connections.....286
 - 15.1.10 Exploring Paths in the Chip Planner..... 286
 - 15.1.11 Viewing Assignments in the Chip Planner..... 289
 - 15.1.12 Viewing High-Speed and Low-Power Tiles in the Chip Planner..... 290
 - 15.1.13 Viewing Design Partition Placement..... 290
 - 15.2 Logic Lock (Standard) Regions..... 291
 - 15.2.1 Logic Lock (Standard) Regions Window.....291
 - 15.2.2 Creating Logic Lock (Standard) Regions.....292
 - 15.2.3 Defining Routing Regions.....295
 - 15.2.4 Placing Logic Lock (Standard) Regions.....296
 - 15.2.5 Placing Device Resources into Logic Lock (Standard) Regions..... 296
 - 15.2.6 Hierarchical (Parent and Child) Logic Lock (Standard) Regions..... 300
 - 15.2.7 Additional Intel Quartus Prime Logic Lock (Standard) Design Features..... 300
 - 15.3 Using Logic Lock (Standard) Regions in the Chip Planner..... 301
 - 15.3.1 Viewing Connections Between Logic Lock (Standard) Regions in the Chip Planner..... 301
 - 15.3.2 Using Logic Lock (Standard) Regions with the Design Partition Planner..... 301
 - 15.4 Scripting Support..... 302
 - 15.4.1 Initializing and Uninitializing a Logic Lock (Standard) Region..... 302
 - 15.4.2 Creating or Modifying Logic Lock (Standard) Regions.....302
 - 15.4.3 Obtaining Logic Lock (Standard) Region Properties.....303
 - 15.4.4 Assigning Logic Lock (Standard) Region Content.....303



15.4.5 Save a Node-Level Netlist for the Entire Design into a Persistent Source File..	303
15.4.6 Setting Logic Lock (Standard) Assignment Priority.....	303
15.4.7 Assigning Virtual Pins with a Tcl command.....	304
15.5 Document Revision History.....	304
16 Netlist Optimizations and Physical Synthesis.....	306
16.1 Physical Synthesis Optimizations.....	306
16.1.1 Enabling Physical Synthesis Optimization.....	307
16.1.2 Physical Synthesis Options.....	307
16.1.3 Perform Register Retiming for Performance.....	308
16.1.4 Preventing Register Movement During Retiming.....	309
16.2 Applying Netlist Optimizations.....	310
16.2.1 WYSIWYG Primitive Resynthesis.....	311
16.2.2 Saving a Node-Level Netlist.....	312
16.3 Viewing Synthesis and Netlist Optimization Reports.....	313
16.4 Isolating a Partition Netlist	313
16.5 Scripting Support.....	314
16.5.1 Synthesis Netlist Optimizations.....	315
16.5.2 Physical Synthesis Optimizations.....	315
16.5.3 Back-Annotating Assignments.....	316
16.6 Document Revision History.....	317
17 Engineering Change Orders with the Chip Planner.....	318
17.1 Engineering Change Orders.....	318
17.1.1 Performance Preservation.....	318
17.1.2 Compilation Time.....	319
17.1.3 Verification.....	319
17.1.4 Change Modification Record.....	320
17.2 ECO Design Flow.....	320
17.3 The Chip Planner Overview.....	322
17.3.1 Opening the Chip Planner.....	322
17.3.2 The Chip Planner Tasks and Layers.....	323
17.4 Performing ECOs with the Chip Planner (Floorplan View).....	323
17.4.1 Creating, Deleting, and Moving Atoms.....	323
17.4.2 Check and Save Netlist Changes.....	323
17.5 Performing ECOs in the Resource Property Editor.....	323
17.5.1 Logic Elements.....	324
17.5.2 Adaptive Logic Modules.....	326
17.5.3 FPGA I/O Elements.....	327
17.5.4 FPGA RAM Blocks.....	331
17.5.5 FPGA DSP Blocks.....	332
17.6 Change Manager.....	333
17.6.1 Complex Changes in the Change Manager.....	334
17.6.2 Managing Signal Probe Signals.....	334
17.6.3 Exporting Changes.....	334
17.7 Scripting Support.....	334
17.8 Common ECO Applications.....	334
17.8.1 Adjust the Drive Strength of an I/O with the Chip Planner.....	335
17.8.2 Modify the PLL Properties With the Chip Planner.....	336
17.8.3 PLL Properties.....	337
17.8.4 Modify the Connectivity between Resource Atoms.....	339



17.9 Post ECO Steps.....	340
17.10 Document Revision History.....	340



1 Constraining Designs

You use constraints, assignments and logic options to control how the Intel® Quartus® Prime software implements a design. Constraints are also central in the way that the Timing Analyzer and the Power Analyzer inform synthesis, placement, and routing.

Intel Quartus Prime software keeps user-created constraints in one of two files:

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

To manually add or modify design constraints, assignments, and logic options, you use the Intel Quartus Prime software tools. If you keep your design and simulation settings in text files, use Tcl scripts. By combining the syntax of the .qsf files and the .sdc files with procedural Tcl, you can automate iteration over several different settings, changing constraints and recompiling.

Related Links

- [Tcl Scripting](#) on page 81
- [Intel Quartus Prime Standard Edition Settings File Reference Manual](#)
For information about all settings and constraints in the Intel Quartus Prime software.

1.1 Constraining Designs with Intel Quartus Prime Tools

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	<ul style="list-style-type: none"> Device Family Top-level Entity 	X	X	X	
Fitter	<ul style="list-style-type: none"> 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 Links

[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 Tools to Set Node, Entity and Instance Level Constraints

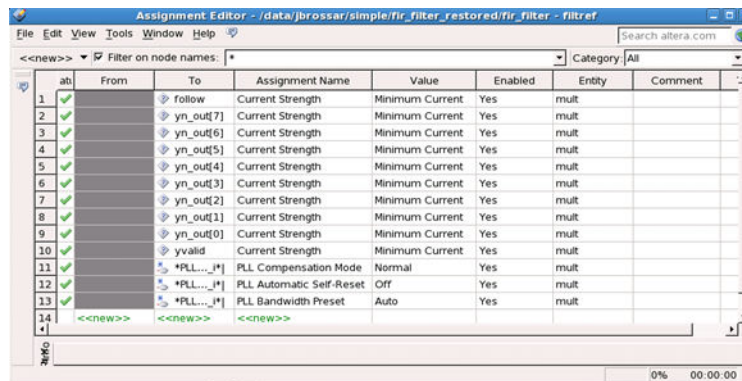
Assignment Type	Example	Assignment Editor	Chip Planner	Pin Planner
Pin	Project files	X		X
Location	<ul style="list-style-type: none"> Device Family Top-level Entity 	X	X	
Routing	<ul style="list-style-type: none"> Device Fitter Effort IO Standard 	X	X	
Simulation	Vector input source	X	X	X

1.1.2.1 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 Constraining Designs with the 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.

Related Links

[Managing Device I/O Pins](#) on page 24

1.1.2.3 Constraining Designs with the Chip Planner

With the Chip Planner, you can adjust existing assignments to device resources, such as pins, logic cells, and LABs using a graphical interface. 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 Links

[Design Floorplan Analysis in the Chip Planner](#) on page 280

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.

You can also use the Design Partition Planner to optimize design performance by isolating and resolving failing paths on a partition-by-partition basis.

Related Links

[Creating Partitions and Logic Lock \(Standard\) Regions with the Design Partition Planner and the Chip Planner](#)

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.

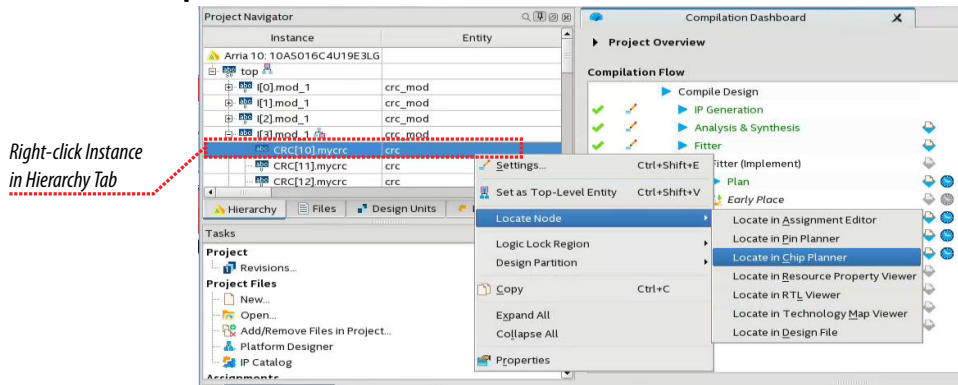
When you are in the Project Navigator, Assignment Editor, Chip Planner, or Pin Planner, and want to display a given resource in other Intel Quartus Prime tool:

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

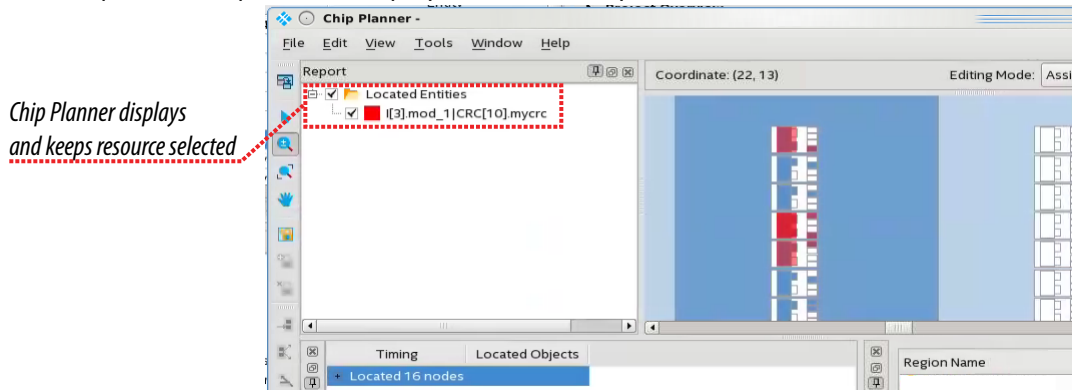
The corresponding window opens—or appears in the foreground if it is already open—and shows the element you clicked.

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**.



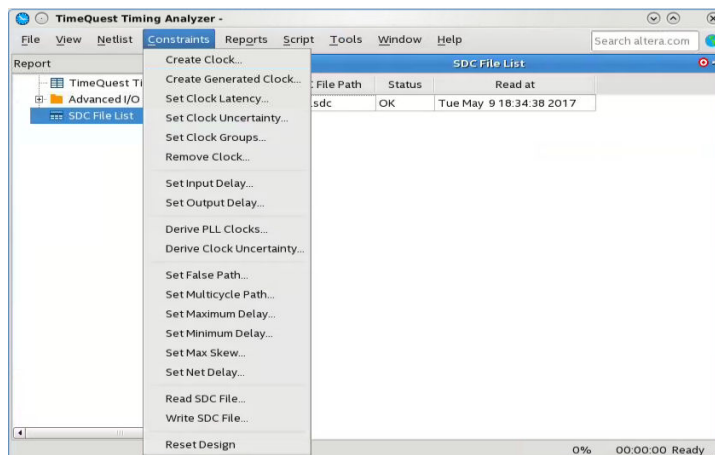
The Chip Planner opens and displays the instance you selected.



1.1.4 Specifying Individual Timing Constraints

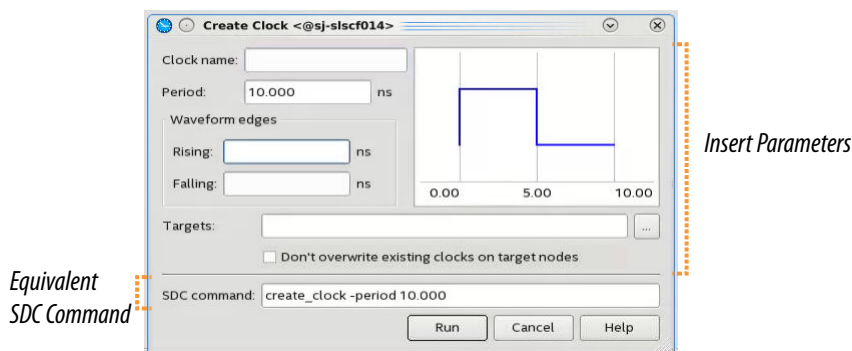
The Timing Analyzer GUI provides the **Constraint** menu for manual specification of timing constraints on individual entities, nodes, and pins.

Figure 2. Constraint menu in Timing Analyzer



When you specify commands and options using the **Constraint** menu, the dialog box displays the equivalent SDC command. This lets you familiarize with the Synopsys Design Constraint syntax, and enables the reuse for iterative scripting.

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 Links

[The Timing Analyzer](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 3*

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 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 Links

[Intel Quartus Prime Standard Edition Settings File Reference Manual](#)

For information about all settings and constraints in the Intel Quartus Prime software.

1.2.2 Timing Analysis with .sdc Files and Tcl Scripts

Intel Quartus Prime software keeps Timing Analyzer 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 constraints 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 Links

[Constraining and Analyzing with Tcl Commands](#)

In Intel Quartus Prime Standard Edition Handbook Volume 3

1.2.3 Using 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 your design's contents to a procedural, executable Tcl (.tcl) file. You can then use that generated script to restore certain 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



1.2.3.1 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 Links

- [::quartus::flow](#)
In *Intel Quartus Prime Help*
- [Tcl Scripting](#) on page 81

1.4 Document Revision History

Table 3. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> • 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. • 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	<ul style="list-style-type: none"> • Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
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.
<i>continued...</i>		



Date	Version	Changes
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 Intel Quartus Prime Help.
November 2009	9.1.0	<ul style="list-style-type: none">Added two notes.Minor text edits.
March 2009	9.0.0	<ul style="list-style-type: none">Revised and reorganized the entire chapter.Added section "Probing to Source Design Files and Other Intel Quartus Prime Windows" on page1-2.Added description of node type icons (Table1-3).Added explanation of wildcard characters.
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated Intel Quartus Prime software 8.0 revision and date.

Related Links

[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 3. Pin Planner GUI

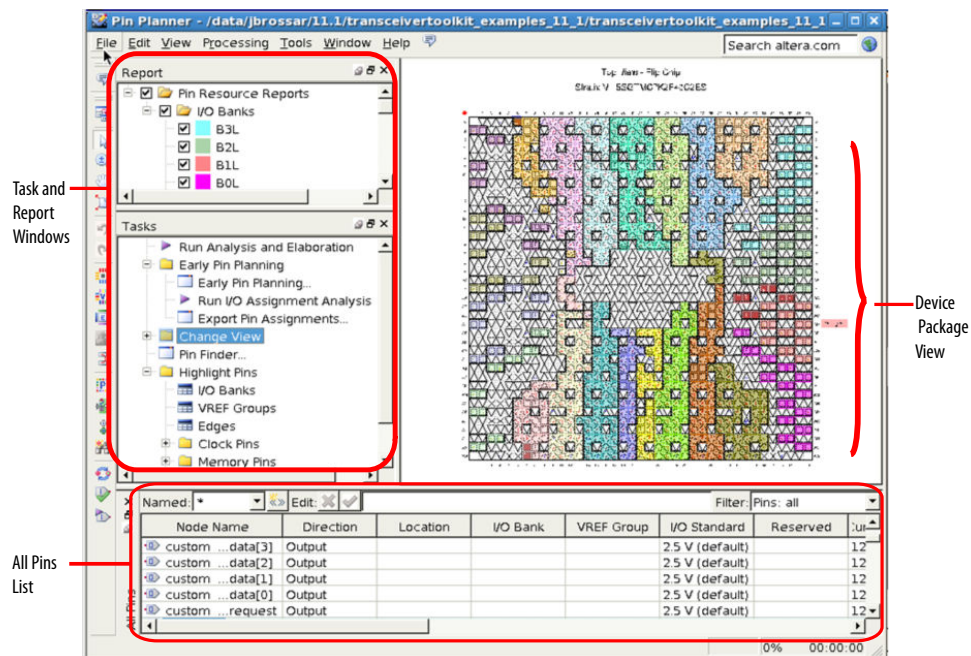


Table 4. Intel Quartus Prime I/O Pin Planning Tools

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 Links

[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

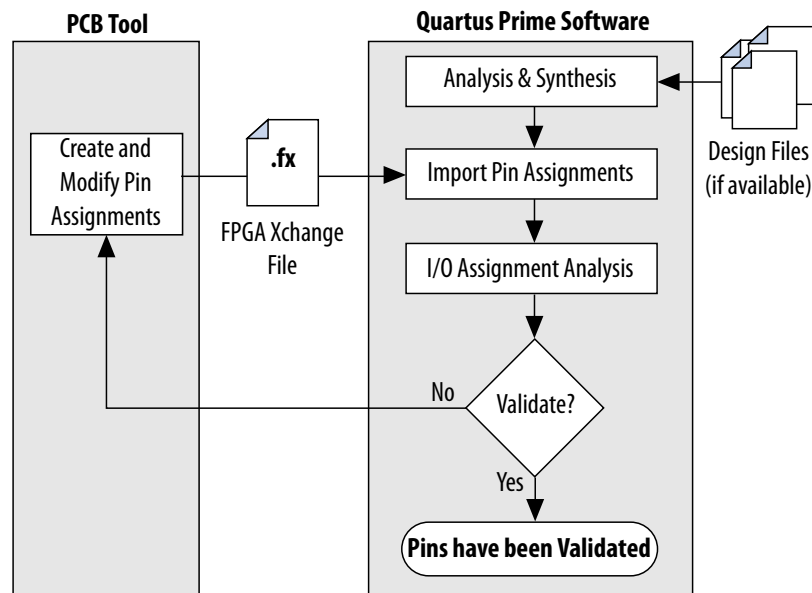
You can integrate PCB design tools into your work flow to map pin assignments to symbols in your system circuit schematics and board layout.

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 5. 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 4. PCB Tool Integration



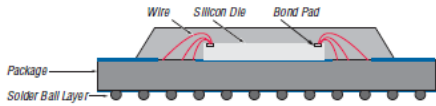
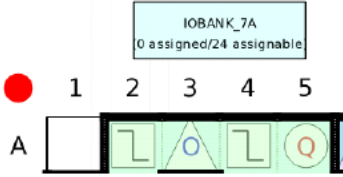
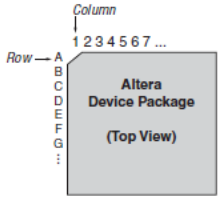
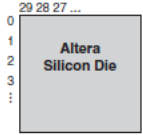
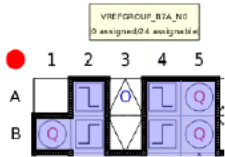
Related Links

- [Mentor Graphics PCB Design Tools Support](#) on page 144
- [Cadence PCB Design Tools Support](#) on page 159



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.

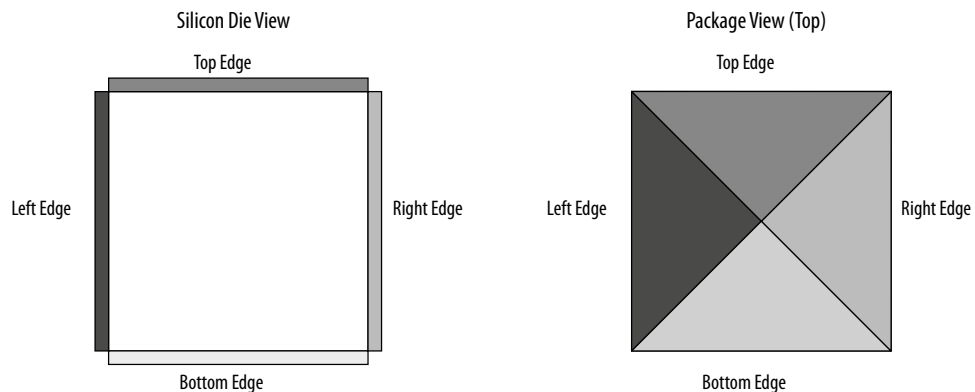
Note: If you have a single-ended clock that feeds a PLL, assign the pin only to the positive clock pin of a differential pair in the target device. Single-ended pins that feed a PLL and are assigned to the negative clock pin device cause the design to not fit.

Figure 5. Creating a Differential Pin Pair in the Pin Planner

	Node Name	Differential Pair	I/O Standard	Direction
5	input_data[4]		3.3-V LVTTTL (default)	Input
6	input_data[3]		3.3-V LVTTTL (default)	Input
7	input_data[2]		3.3-V LVTTTL (default)	Input
8	input_data[1]		3.3-V LVTTTL (default)	Input
9	input_data[0]		3.3-V LVTTTL (default)	Input
10	lvds_in	lvds_in(n)	LVDS	Input
11	output_data[7]		3.3-V LVTTTL (default)	Output
12	output_data[6]		3.3-V LVTTTL (default)	Output
13	output_data[5]		3.3-V LVTTTL (default)	Output
14	output_data[4]		3.3-V LVTTTL (default)	Output
15	output_data[3]		3.3-V LVTTTL (default)	Output
16	output_data[2]		3.3-V LVTTTL (default)	Output
17	output_data[1]		3.3-V LVTTTL (default)	Output
18	output_data[0]		3.3-V LVTTTL (default)	Output
19	reset		3.3-V LVTTTL (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 6. 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 use Tcl scripts to apply pin assignments rather than using the GUI. Enter individual Tcl commands in the Tcl Console, or type the following to apply the assignments contained in a Tcl script:

Example 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 Links

[Tcl Scripting](#) on page 81

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. The assignments appear in the Pin Planner. If you modify or delete these assignments in the Pin Planner and then recompile your design, the Pin Planner changes override the synthesis attributes. Intel Quartus Prime synthesis supports the `chip_pin`, `useioff`, and `altera_attribute` synthesis attributes.



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 8. Verilog HDL Synthesis Attribute

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

Example 9. VHDL Synthesis Attribute

```
VHDL Example
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;
```

Use the `altera_attribute` synthesis attribute to create other pin-related assignments in your HDL code. The `altera_attribute` attribute is understood only by Intel Quartus Prime integrated synthesis and 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.

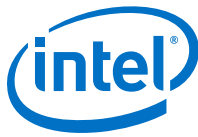
Example 10. Verilog HDL Synthesis Attribute

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

Example 11. VHDL Synthesis Attribute

```
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;
```



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 Links

[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 6. Importing and Exporting I/O Pin Assignments

	Import Assignments	Export Assignments
Scenario	<ul style="list-style-type: none"> 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 	<ul style="list-style-type: none"> 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 7. 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
<i>continued...</i>	



File Column Name	Description
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 Links

- [Pin-Out Files for Intel Devices](#)
- [Mentor Graphics PCB Design Tools Support on page 144](#)

2.3.2 Migrating Assignments to Another Target Device

Click **View > Pin Migration Window** to verify whether your 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 7. Device Migration Compatibility (AC24 does not exist in migration device)

Pin Number	Pin Function	Migration Result				Migration Devices												
		I/O Bank	VREF Group	Clock Pin	Dir	EP2530F672C4				EP2519F672C4				EP2560F672C4				
						I/O Bank	VREF Group	Clock Pin	Dir	I/O Bank	VREF Group	Clock Pin	Dir	I/O Bank	VREF Group	Clock Pin	Dir	
87	PIN_AC11	VREFB7N0	7	B7_N0	Yes	VREFB7N0	7	B7_N0	Yes	VREFB7N0	7	B7_N0	Yes	VREFB7N0	7	B7_N0	Yes	Yes
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	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	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	Yes
91	PIN_AC15	NC				Column I/O	8	B8_N1	NC					Column I/O	12	B8_N2	Yes	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	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	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	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 display the voltage reference. Even if you do not use that pin for a port connection in your design, you must use the VREF standard for I/O standards that require it on the actual board for the migration device.

If one of the migration devices has pins intended for connection to V_{CC} or GND and these same pins are I/O pins on a different device in the migration path, the 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 your original design because the pins are not physically connected to the smaller die.

Related Links

[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 8. 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 9. 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

continued...



Rule	Description	HDL Required?
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
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 10. 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 Running I/O Assignment Analysis

I/O assignment analysis validates I/O assignments against the complete set of I/O system and board layout rules. Full I/O assignment analysis validates blocks that directly feed or are fed by resources such as a PLL, LVDS, or gigabit transceiver blocks. In addition, the checker validates the legality of proper V_{REF} 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.

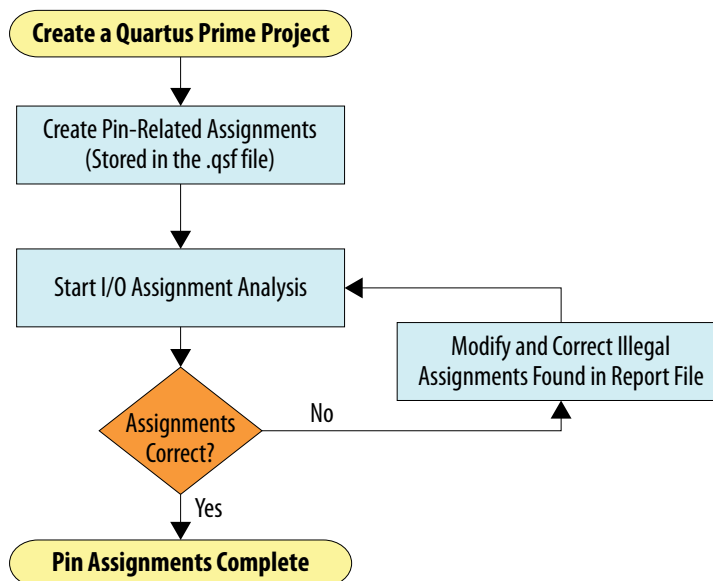
View the I/O Assignment Warnings report to view and resolve all assignment warnings. For example, a warning that some design pins have undefined drive strength or slew rate. The Fitter recognizes undefined, single-ended output and bidirectional pins as non-calibrated OCT. To resolve the warning, assign 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 Running Early I/O Assignment Analysis (Without Design Files)

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

Figure 8. Assigning and Analyzing Pin-Outs without Design Files

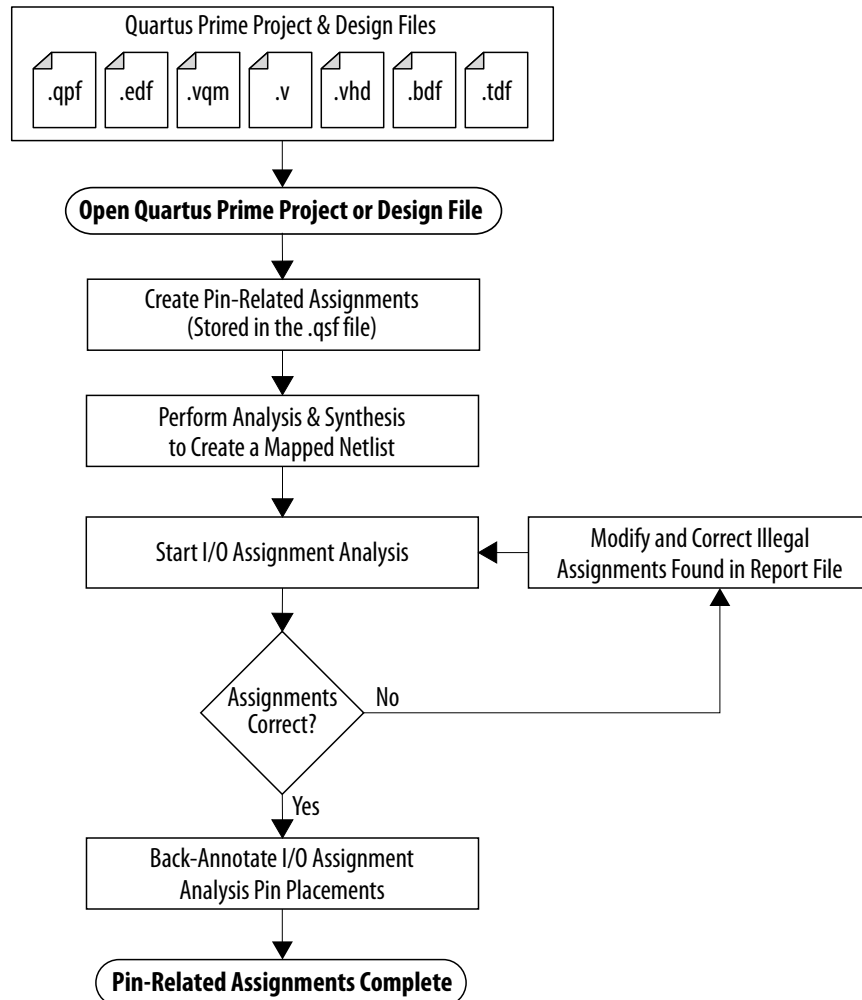


You must reserve all pins you intend to use as I/O pins, so that the Fitter can determine each pin type. After performing I/O assignment analysis, correct any errors reported by the Fitter and rerun I/O assignment analysis until all errors are corrected. A complete I/O assignment analysis requires all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

2.4.3.2 Running I/O Assignment Analysis (with Design Files)

Use I/O assignment analysis to perform full I/O legality checks after fully defining HDL design files. When you run I/O assignment analysis on a complete design, the tool verifies all I/O pin assignments against all I/O rules. When you run I/O assignment analysis on a partial 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 9. I/O Assignment Analysis Flow



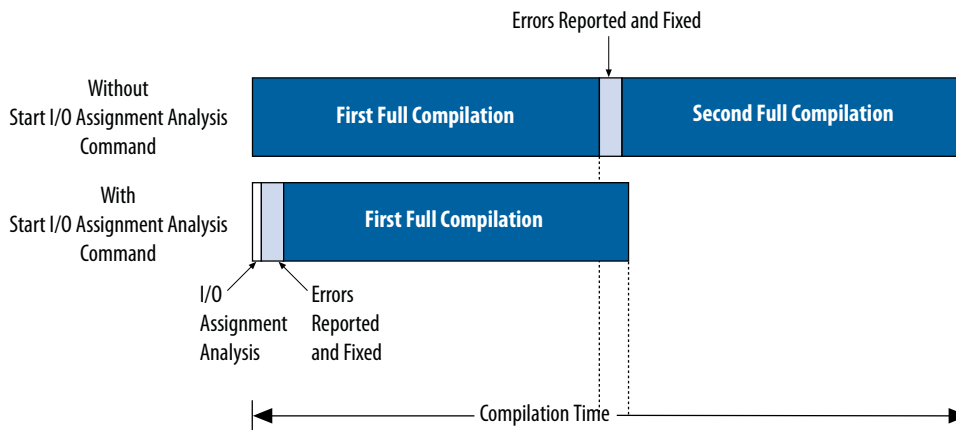
Even if I/O assignment analysis passes on incomplete design files, you may still encounter errors during full compilation. For example, you can assign a clock to a user I/O pin instead of assigning it to a dedicated clock pin, or design the clock to drive a PLL that you have not yet instantiated in the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

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

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



Figure 10. 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 various pins to accommodate I/O rule exceptions, such as for analyzing VREF or inactive pins.

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

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

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

Related Links

[The Timing Analyzer](#)

In Intel Quartus Prime Standard Edition Handbook Volume 3

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 11. 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 11. 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 t _{CO} 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 Links

- [Literature and Technical Documentation](#)
- [Intel Signal & Power Integrity Center](#)
- [Signal Integrity Analysis with Third-Party Tools](#) on page 105

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.

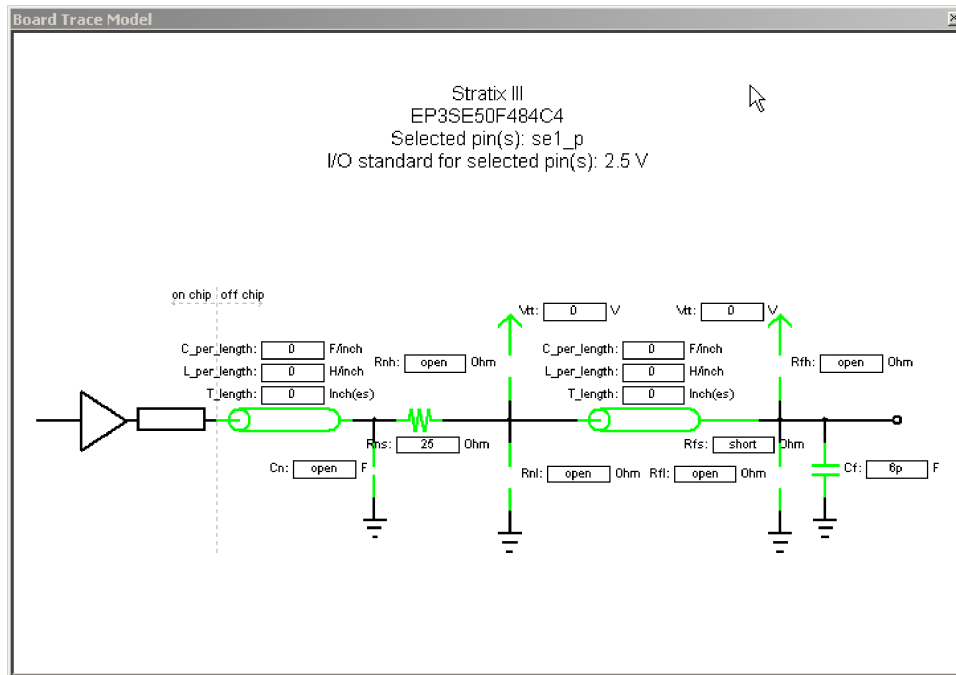
2.5.1.1 Understanding the Board Trace Models

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.

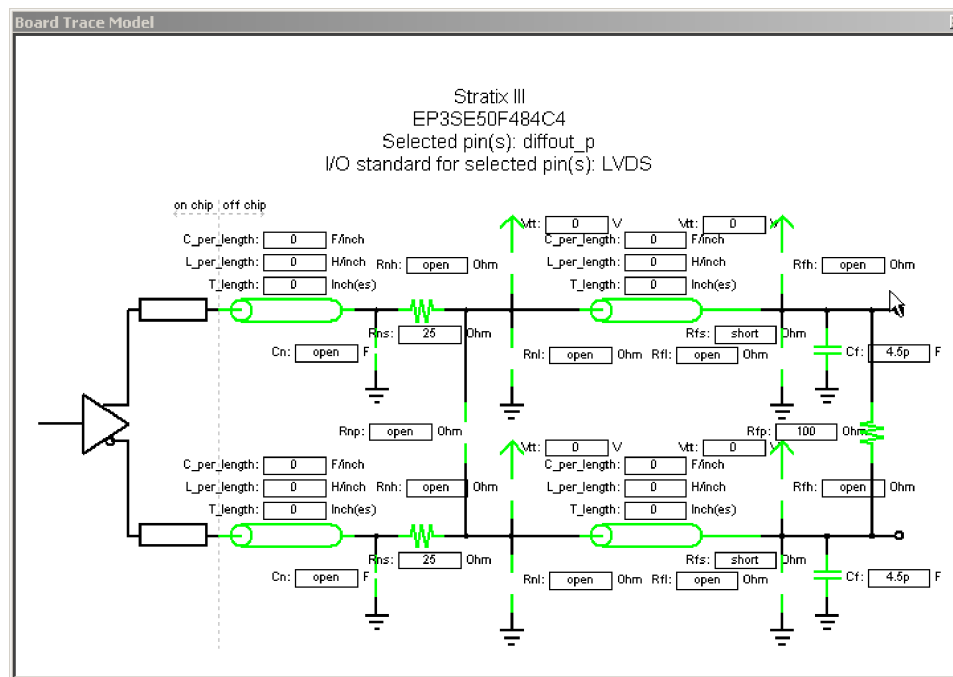
Figure 12. 2.5-V I/O Standard Board Trace Model



The following figure shows the template for the **LVDS** I/O standard. The far-end capacitance (C_f) 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 lossless transmission lines, and does not require a transmission-line resistance value. Only distributed inductance (L) and capacitance (C) values are needed. The distributed L and C values of transmission lines must be entered on a per-inch basis, and can be obtained from the PCB vendor or manufacturer, the CAD Design tool, or a signal integrity tool, such as the Mentor Graphics HyperLynx software.

Figure 13. 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 sel_p
```



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

Related Links

[Board Trace Model](#)

In *Intel Quartus Prime Help*

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 Understanding Advanced I/O Timing Analysis Reports

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

Table 12. 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 Links

- [The Timing Analyzer](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 3*



- [The 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

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

```
quartus_sh --qhelp
```

Related Links

- [Tcl Scripting](#) on page 81
- [Command Line Scripting](#) on page 67

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

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

```
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 Links

[Package Information Datasheet for Mature Altera Devices](#)

2.9 Document Revision History

The following table shows the revision history for this chapter.

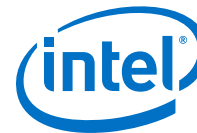
Table 13. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated Live I/O check device support to include only limited device families.
2014.08.30	14.0a10.0	<ul style="list-style-type: none"> Added link to information about special pin assignment features for Arria 10 SoC devices.
2014.06.30	14.0.0	<ul style="list-style-type: none"> Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2013	13.1.0	<ul style="list-style-type: none"> Reorganization and conversion to DITA.
May 2013	13.0.0	<ul style="list-style-type: none"> Added information about overriding I/O placement rules.
November 2012	12.1.0	<ul style="list-style-type: none"> Updated Pin Planner description for new task and report windows.
June 2012	12.0.0	<ul style="list-style-type: none"> Removed survey link.
November 2011	11.1.0	<ul style="list-style-type: none"> Minor updates and corrections. Updated the document template.
December 2010	10.0.1	Template update
July 2010	10.0.0	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Updated "Pad View Window" on page 5-20 Added new figures: <ul style="list-style-type: none"> 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 Links

Documentation Archive

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



3 Simultaneous Switching Noise (SSN) Analysis and Optimizations

3.1 Simultaneous Switching Noise (SSN) Analysis and Optimizations

FPGA design has evolved from small programmable circuits to designs that compete with multimillion-gate ASICs. At the same time, the I/O counts on FPGAs and logic density requirements of designs have increased exponentially.

The higher-speed interfaces in FPGAs, including high-speed serial interfaces and memory interfaces, require careful interface design on the PCB. Designers must address the timing and signal integrity requirements of these interfaces early in the design cycle. Simultaneous switching noise (SSN) often leads to the degradation of signal integrity by causing signal distortion, thereby reducing the noise margin of a system.

Today's complex FPGA system design is incomplete without addressing the integrity of signals coming in to and out of the FPGA. Altera recommends that you perform SSN analysis early in your FPGA design and prior to the layout of your PCB with complete SSN analysis of your FPGA in the Intel Quartus Prime software. This chapter describes the Intel Quartus Prime SSN Analyzer tool and covers the following topics:

3.2 Definitions

The terminology used in this chapter includes the following terms:

- **Aggressor:** An output or bidirectional signal that contributes to the noise for a victim I/O pin
- **PDN:** Power distribution network
- **QH:** Quiet high signal level on a pin
- **QHN:** Quiet high noise on a pin, measured in volts
- **QL:** Quiet low signal level on a pin
- **QLN:** Quiet low noise on a pin, measured in volts
- **SI:** Signal integrity (a superset of SSN, covering all noise sources)
- **SSN:** Simultaneous switching noise
- **SSO:** Simultaneous switching output (which are either the output or bidirectional pins)
- **Victim:** An input, output, or bidirectional pin that is analyzed during SSN analysis. During SSN analysis, each pin is analyzed as a victim. If a pin is an output or bidirectional pin, the same pin acts as an aggressor signal for other pins.



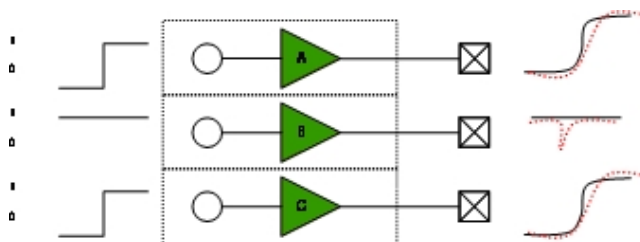
3.3 Understanding SSN

SSN is defined as a noise voltage induced onto a single victim I/O pin on a device due to the switching behavior of other aggressor I/O pins on the device. SSN can be divided into two types of noise: voltage noise and timing noise.

In a sample system with three pins, two of the pins (A and C) are switching, while one pin (B) is quiet. If the pins are driven in isolation, the voltage waveforms at the output of the buffers appear without noise interference, as shown by the solid curves at the left of the figure. However, when pins A and C are switching simultaneously, the noise generated by the switching is injected onto other pins. This noise manifests itself as a voltage noise on pin B and timing noise on pins A and C.

Figure 14. System with Three Pins

In this figure, the dotted curves show the voltage noise on pin B and timing noise on pins A and C.



Voltage noise is measured as the change in voltage of a signal due to SSN. When a signal is QH, it is measured as the change in voltage toward 0 V. When a signal is QL, it is measured as the change in voltage toward V_{CC} .

In the Intel Quartus Prime software, only voltage noise is analyzed. Voltage noise can be caused by SSOs under two worst-case conditions:

- The victim pin is high and the aggressor pins (SSOs) are switching from low to high
- The victim pin is low and the aggressor pins (SSOs) are switching from high to low

Figure 15. Quiet High Output Noise Estimation on Pin B

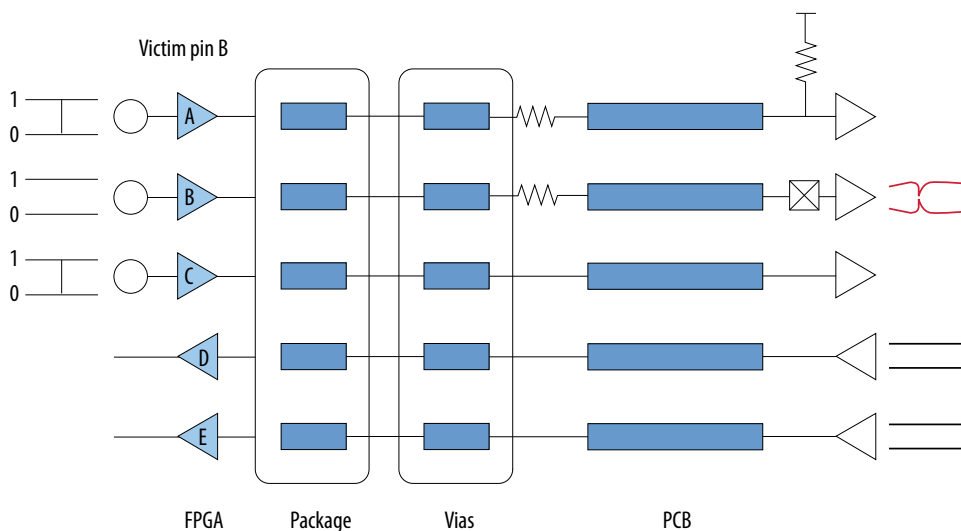
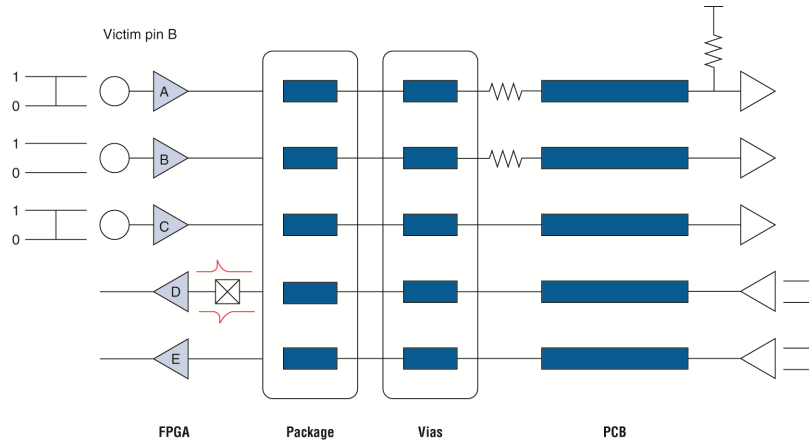
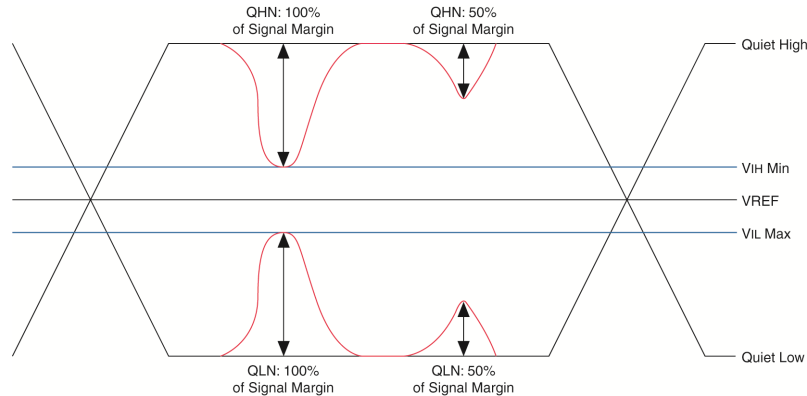


Figure 16. Quiet Low Input Noise Estimation for Pin D



SSN can occur in any system, but the induced noise does not always result in failures. Voltage functional errors are caused by SSN on quiet victim pins only when the voltage values on the quiet pins change by a large voltage that the logic listening to that signal reads a change in the logic value. For QH signals, a voltage functional error occurs when noise events cause the voltage to fall below V_{IH} . Similarly, for QL signals, a voltage functional error occurs when noise events cause the voltage to rise above V_{IL} . Because V_{IH} and V_{IL} of the Altera device are different for different I/O standards, and because signals have different quiet voltage values, the absolute amount of SSN, measured in volts, cannot be used to determine if a voltage failure occurs. Instead, to assess the level of impact by SSN in the SSN analysis, the Intel Quartus Prime software quantifies the SSN in terms of the percentage of signal margin in Intel devices.

Figure 17. Reporting Noise Margins

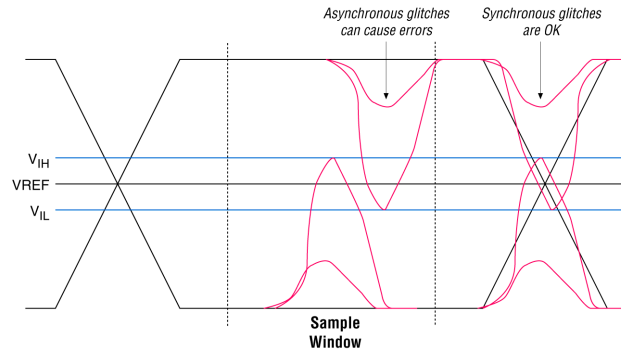


The figure shows four noise events, two on QH signals and two on QL signals. The two noise events on the right-side of the figure consume 50 percent of the signal margin and do not cause voltage functional errors. However, the two noise events on the left side of the figure consume 100 percent of the signal margin, which can cause a voltage functional error.

Noise caused by aggressor signals are synchronously related to the victim pin outside of the sampling window of a receiver. This noise affects the switching time of a victim pin, but are not considered an input threshold violation failure.



Figure 18. Synchronous Voltage Noise with No Functional Error



Related Links

[SSN Analysis Overview](#) on page 52

3.4 SSN Estimation Tools

Addressing SSN early in your FPGA design and PCB layout can help you avoid costly board respins and lost time, both of which can impact your time-to-market.

Intel provides many tools for SSN analysis and estimation, including the following tools:

- SSN characterization reports
- An early SSN estimation (ESE) tool
- The SSN Analyzer in the Intel Quartus Prime software

The ESE tool is useful for preliminary SSN analysis of your FPGA design; for more accurate results, however, you must use the SSN Analyzer in the Intel Quartus Prime software.

Table 14. Comparison of ESE Tool and SSN Analyzer Tool

ESE Tool	SSN Analyzer
Is not integrated with the Intel Quartus Prime software.	Integrated with the Intel Quartus Prime software, allowing you to perform preliminary SSN analysis while making I/O assignment changes in the Intel Quartus Prime software.
QL and QH levels are computed assuming a worst-case pattern of I/O placements.	QL and QH levels are computed based on the I/O placements in your design.
No support for entering board information.	Supports board trace models and board layer information, resulting in a more accurate SSN analysis.
No graphical representation.	Integrated with the Intel Quartus Prime Pin Planner, in which an SSN map shows the QL and QH levels on victim pins.
Good for doing an early SSN estimate. Does not require you to use the Intel Quartus Prime software.	Requires you to create a Intel Quartus Prime software project and provide the top-level port information.

Related Links

[Altera Signal Integrity Center](#)

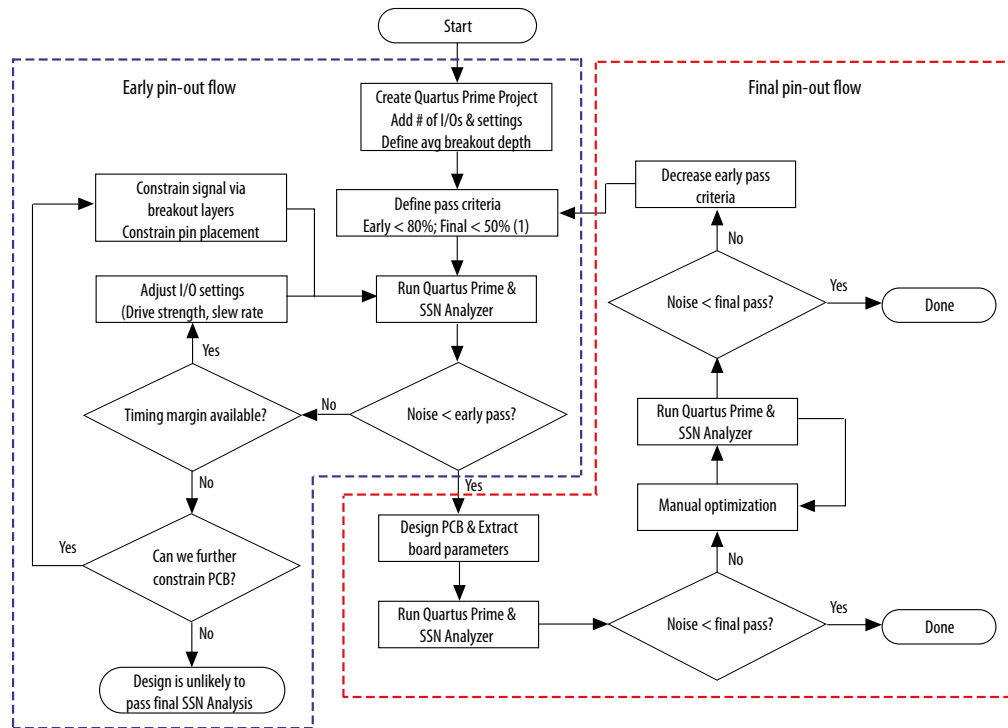
3.5 SSN Analysis Overview

You can run the SSN Analyzer at different stages in your design cycle to obtain SSN results. The accuracy of the results depends on the completeness of your design information.

Start SSN analysis early in the design cycle to obtain preliminary results and adjust your I/O assignments. Iterate through the design cycle to finally perform a fully constrained SSN analysis with complete information about your board.

The early pin-out flow assumes conservative design rules initially, and then lets you analyze the design and iteratively apply tighter design rules until SSN analysis indicates your design meets SSN constraints. You must define pass criteria for SSN analysis as a percentage of signal margin in both the early pin-out flow and the final pin-out flow. The pass criteria you define is specific to your design requirements. For example, a pass criterion you might define is a condition that verifies you have sufficient SSN margins in your design. You may require that the acceptable voltage noise on a pin must be below 70% of the voltage level for that pin. The pass criteria for the early-pin out flow may be higher than the final pin-out flow criteria, so that you do not spend too much time optimizing the on-FPGA portions of your design when the SSN metrics for the design may improve after the design is fully specified.

Figure 19. Early Flow and Final Pin-Out SSN Analysis



Note:

1. Pass criteria determined by customer requirements.



3.5.1 Performing Early Pin-Out SSN Analysis

In the early stages of your design cycle, before you create pin location for your design, use the early pin-out flow to obtain preliminary SSN analysis results.

To obtain useful SSN results, you must define the top-level ports of your design, but your design files do not have to be complete.

Performing Early Pin-Out SSN Analysis with the ESE Tool

If you know the I/O standards and signaling standards for your design, you can use the ESE tool to perform an initial SSN evaluation.

3.5.1.1 Performing Early Pin-Out SSN Analysis with the SSN Analyzer

In the early stages of your design cycle, you may not have complete board information, such as board trace parameters, layer information, and the signal breakout layers. If you run the SSN Analyzer without this specific information, it uses default board trace models and board layer information for SSN analysis, and as a result the SSN Analyzer confidence level is low. If the noise amounts are larger than the pass criteria for early pin-out SSN analysis, verify whether the SSN noise violations are true failures or false failures. For example, sometimes the SSN Analyzer can determine whether pins are switching synchronously and use that information to filter false positives; however, it may not be able to determine all the synchronous groups. You can improve the SSN analysis results by adjusting your I/O assignments and other design settings. After you optimize your design such that it meets the pass criteria for the early pin-out flow, you can then begin to design your PCB.

If you have complete information for the top-level ports of your design, you can use the SSN Analyzer to perform an initial SSN evaluation. Use the following steps to perform early pin-out SSN analysis:

1. Create a project in the Intel Quartus Prime software.
2. Specify your top-level design information either in schematic form or in HDL code.
3. Perform Analysis and Synthesis.
4. Create I/O assignments, such as I/O standard assignments, for the top-level ports in your design.

Note: Do not create pin location assignments. The Fitter automatically creates optimized pin location assignments.

5. If you do not have completed design files and timing constraints, run I/O assignment analysis.

Note: During I/O assignment analysis, the Fitter places all the unplaced pins on the device, and checks all the I/O placement rules.

6. Run the SSN Analyzer.

Related Links

- [Optimizing Your Design for SSN Analysis](#) on page 54
- [Managing Intel Quartus Prime Projects](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*
- [Managing Device I/O Pins](#) on page 24



3.5.2 Performing Final Pin-Out SSN Analysis

You perform final pin-out SSN analysis after you place all the pins in your design, or the Fitter places them for you, and you have complete information about the board trace models and PCB layers.

Even if your design achieves sufficient SSN results during early pin-out SSN analysis, you should run SSN analysis with the complete PCB information to ensure that SSN does not cause failures in your final design. You must specify the board parameters in the Intel Quartus Prime software, including the PCB layer thicknesses, the signal breakout layers, and the board trace models, before you can run SSN analysis on your final assignments.

If the SSN analysis results meet the pass criteria for final pin-out SSN analysis, SSN analysis is complete. If the SSN analysis results do not meet the pass criteria, you must further optimize your design by changing the board and design parameters and then rerun the SSN Analyzer. If the design still does not meet the pass criteria, reduce the pass criteria for early pin-out SSN analysis, and restart the process. By reducing the pass criteria for early pin-out SSN analysis, you place a greater emphasis on reducing SSN through I/O settings and I/O placement. Changing the drive strength and slew rate of output and bidirectional pins, as well as adjusting the placement of different SSOs, can affect SSN results. Adjusting I/O settings and placement allows the design to meet the pass criteria for final pin-out SSN analysis after you specify the actual PCB board parameters.

3.6 Design Factors Affecting SSN Results

There are many factors that affect the SSN levels in your design. The two main factors are the drive strength and slew rate settings of the output and bidirectional pins in your design.

Related Links

[Altera Signal Integrity Center](#)

3.7 Optimizing Your Design for SSN Analysis

The SSN Analyzer gives you flexibility to precisely define your system to obtain accurate SSN results.

The SSN Analyzer produces a voltage noise estimate for each input, output, and bidirectional pin in the design. It allows you to estimate the SSN levels, comprised of QLN and QHN levels, for your FPGA pins. Performing SSN analysis helps you optimize your design for SSN during compilation.

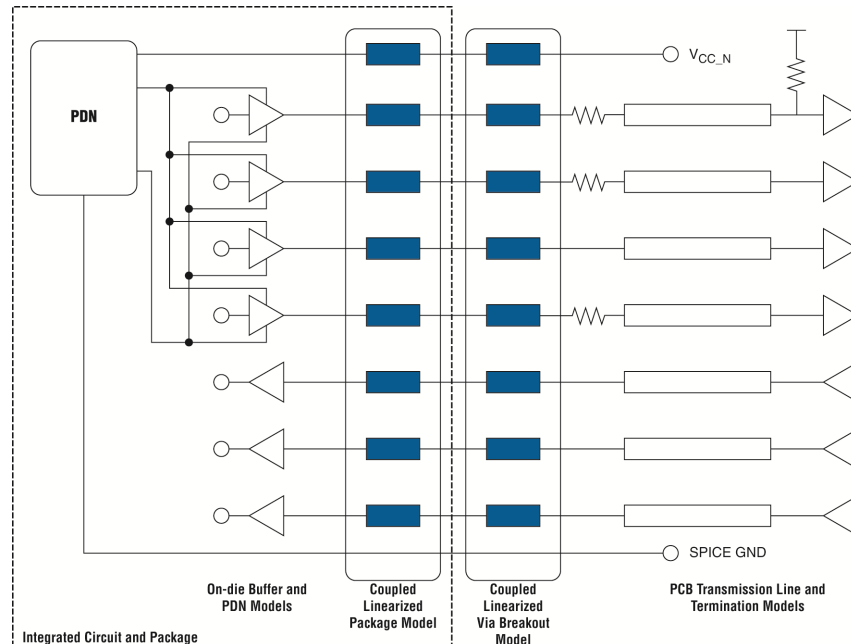
Because the SSN Analyzer is integrated into the Intel Quartus Prime software, it can automatically set up a system topology that matches your design. The SSN Analyzer accounts for different I/O standards and slew rate settings for each buffer in the design and models different board traces for each signal. Also, it correctly models the state of the unused pins in the design. The SSN Analyzer leverages any custom board trace assignments you set up for use by the advanced I/O timing feature.



The SSN Analyzer also models the package and vias in the design. Models for the different packages that Altera devices support are integrated into the Intel Quartus Prime software. In the Intel Quartus Prime software, you can specify different layers on which signals break out, each with its own thickness, and then specify which signal breaks out on which layer.

After constructing the circuit topology, the SSN Analyzer uses a simulation-based methodology to determine the SSN for each victim pin in the design.

Figure 20. Circuit Topology for SSN Analysis



3.7.1 Optimizing Pin Placements for Signal Integrity

The **SSN Optimization** logic option tells the Fitter to adjust the pin placement to reduce the SSN in the design.

The **SSN Optimization** logic option has three possible values: **Off**, **Normal compilation**, and **Extra Effort**.

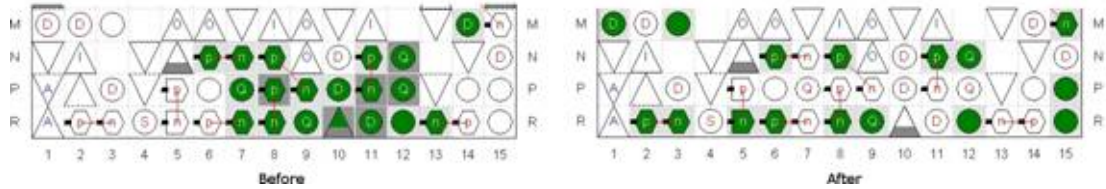
If you use the **SSN Optimization** logic option, avoid creating location assignments for your pins. Instead, let the Fitter place the pins during compilation to meet the timing performance of your design.

To display the Fitter-placed pins use the **Show Fitter Placements** feature in the Pin Planner. To accept these suggested pin locations, you must back-annotate your pin assignments.

Note: Setting the **SSN Optimization** option to **Extra effort** may impact your design f_{MAX} .

Figure 21. SSN Analysis Results Before and After Using the SSN Optimization Logic Option

The image shows the pin placement before and after turning on the **SSN Optimization** logic option.



Related Links

- [Show Commands \(View Menu/Task Window\) \(Pin Planner\)](#)
In Intel Quartus Prime Help
- [Area Optimization](#) on page 267
- [Timing Closure and Optimization](#) on page 201

3.7.2 Specifying Board Trace Model Settings

The SSN Analyzer uses circuit models to determine voltage noise during SSN analysis. The circuit topology is incomplete without board trace information and PCB layer information.

To accurately compute the SSN in your FPGA device, you must describe the board trace and PCB layer parameters in your design. If you do not specify some or all the board trace parameters and PCB layer information, the SSN Analyzer uses default parameters, which set the SSN confidence level to low.

The SSN Analyzer requires board trace models such as termination resistors, pin loads (capacitance), and transmission line parameters. You can define the board circuit models—or board trace models, in the Intel Quartus Prime software.

You can define an overall board trace model for each I/O standard in your design. This overall model is the default model for all pins that use a particular I/O standard. After configuring the overall board trace model, you can customize the model for specific pins.

Intel Quartus Prime software uses the parameters you specify for the board trace model during advanced I/O timing analysis with the Timing Analyzer.

If you already specified the board trace models as part of your advanced I/O timing assignments, Intel Quartus Prime software uses the same parameters during SSN analysis.

All the assignments for board trace models you specify are saved to the .qsf file. You can also use Tcl commands to create board trace model assignments.

Tcl Commands for Specifying Board Trace Models

```
set_instance_assignment -name BOARD_MODEL_TLINE_L_PER_LENGTH "3.041E-7" -to e[0]
set_instance_assignment -name BOARD_MODEL_TLINE_LENGTH 0.1391 -to e[0]
set_instance_assignment -name BOARD_MODEL_TLINE_C_PER_LENGTH "1.463E-10" -to e[0]
```




The best way to calculate transmission line parameters is to use a two-dimensional solver to estimate the inductance per inch and capacitance per inch for the transmission line. You can obtain the termination resistor topology information from the PCB schematics. The near-end and far-end pin load (capacitance) values can be obtained from the PCB schematic and other device data sheets. For example, if you know that an FPGA pin is driving a DIMM, you can obtain the far-end loading information in the data sheet for your target device.

Related Links

- [Managing Device I/O Pins](#) on page 24
- [Board Trace Model](#)
In Intel Quartus Prime Help.
- [Literature and Technical Documentation](#)

3.7.3 Defining PCB Layers and PCB Layer Thickness

Every PCB is fabricated using a number of layers. To remove some of the pessimism from your SSN results, Altera recommends that you create assignments describing your PCB layers in the Intel Quartus Prime software.

You can specify the number of layers on your PCB, and their thickness. The PCB layer information is used only during SSN analysis and is not used in other processes run by the Intel Quartus Prime software. If a custom PCB breakout region is not described you can select the default thickness, which directs the SSN Analyzer to use a single-layer PCB breakout region during SSN analysis.

All the assignments you create for the PCB layers are saved to the `.qsf`. You can also use Tcl commands to create PCB layer assignments. You can create any number of PCB layers, however, the layers must be consecutive.

Tcl Commands for Specifying PCB Layer Assignments

```
set_global_assignment -name PCB_LAYER_THICKNESS 0.00099822M -section_id 1
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 2
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 3
```

The cross-section shows the stackup information of a PCB, which tells you the number of layers used in your PCB. The PCB shown in this example consists of various signal and circuit layers on which FPGA pins are routed, as well as the power and ground layers.

Figure 22. Snapshot of Stackup of a PCB Shown in the Allegro Board Design Environment

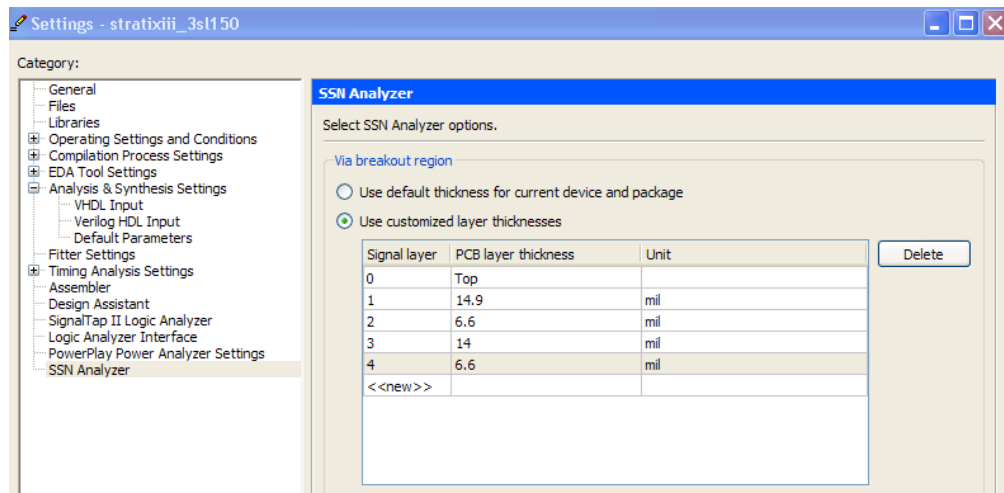
Layout Cross Section				
Cross Section				
	Subclass Name	Type	Material	Thickness (MIL)
1		SURFACE	AIR	
2	TOP	CONDUCTOR	PLATED_COPPER_FOIL	1.9
3		DIELECTRIC	FR-4	3.6
4	L2-PWR	PLANE	COPPER	1.2
5		DIELECTRIC	FR-4	3
6	L3-GND	PLANE	COPPER	1.2
7		DIELECTRIC	FR-4	4
8	L4-SIGNAL	CONDUCTOR	COPPER	0.6
9		DIELECTRIC	FR-4	6
10	L5-SIGNAL	CONDUCTOR	COPPER	0.6
11		DIELECTRIC	FR-4	4
12	L6-GND	PLANE	COPPER	1.2
13		DIELECTRIC	FR-4	3
14	L7-PWR	PLANE	COPPER	1.2
15		DIELECTRIC	FR-4	4
16	L8-SIGNAL	CONDUCTOR	COPPER	0.6
17		DIELECTRIC	FR-4	6
18	L9-SIGNAL	CONDUCTOR	COPPER	0.6
19		DIELECTRIC	FR-4	4
20	L10-GND	PLANE	COPPER	1.2
21		DIELECTRIC	FR-4	3



In this example, each of the four signal layers are a different thickness, with the depths shown in the **Thickness (MIL)** column. The layer thickness for each signal layer is computed as follows:

- Signal Layer 1 is the L4-SIGNAL, at thickness $(1.9+3.6+1.2+3+1.2+4=)$ 14.9 mils
- Signal Layer 2 is the L5-SIGNAL, at thickness $(0.6+6=)$ 6.6 mils
- Signal Layer 3 is the L8-SIGNAL, at thickness $(0.6+4+1.2+3+1.2+4=)$ 14 mils
- Signal Layer 4 is the L9-SIGNAL, at thickness $(0.6+6=)$ 6.6 mils

Figure 23. PCB Layers and Thickness Assignments Specified in the Intel Quartus Prime Software



3.7.4 Specifying Signal Breakout Layers

Each user I/O pin in your FPGA device can break out at different layers on your PCB. In the Pin Planner, you can specify on which layers the I/O pins in your design break out.

The breakout layer information is used only during SSN analysis and is not used in other processes run by the Intel Quartus Prime software. To assign a pin to PCB layer, follow these steps:

1. On the Assignments menu, click **Pin Planner**.
2. If necessary, perform Analysis & Elaboration, Analysis & Synthesis, or fully compile the design to populate the Pin Planner with the node names in the design.
3. Right-click anywhere in the **All Pins** or **Groups** list, and then click **Customize Columns**.
4. Select the **PCB layer** column and move it from the **Available columns** list to the **Show these columns in this order** list.
5. Click **OK**.
6. In the **PCB layer** column, specify the PCB layer to which you want to connect the signal.
7. On the File menu, click **Save Project** to save the changes.



Note: When you create PCB breakout layer assignments in the Pin Planner, you can assign the pin to any layer, even if you did not yet define the PCB layer.

3.7.5 Creating I/O Assignments

I/O assignments are required in FPGA design and are also used during SSN analysis to estimate voltage noise.

Each input, output, or bidirectional signal in your design is assigned a physical pin location on the device using pin location assignments. Each signal has a physical I/O buffer that has a specific I/O standard, pin location, drive strength, and slew rate. The SSN Analyzer supports most I/O standards in a device family, such as the **LVTTTL** and **LVC MOS** I/O standards.

Note: The SSN Analyzer does not support differential I/O standards, such as the **LVDS** I/O standard and its variations, because differential I/O standards contribute a small amount of SSN.

Related Links

- [Literature and Technical Documentation](#)
For more information on the Altera website about supported I/O standards.
- [I/O Management](#) on page 24
For more information about creating and managing I/O assignments, refer to the *Intel Quartus Prime Handbook*.

3.7.6 Decreasing Pessimism in SSN Analysis

In the absence of specific timing information, the SSN Analyzer analyzes your design under worst-case conditions.

Worst-case conditions include all pins acting as aggressor signals on all possible victim pins and all aggressor pins switching with the worst possible timing relationship. The results of SSN analysis under worst-case conditions are very pessimistic. You can improve the results of SSN Analysis by creating group assignments for specific types of pins. Use the following group assignments to decrease the pessimism in SSN analysis results:

- Assign pins to an output enable group—All pins in an output enable group must be either all input pins or all output pins. If all the pins in a group are always either all inputs or all outputs, it is impossible for an output pin in the group to cause SSN noise on an input pin in the group. You can assign pins to an output enable group with the **Output Enable Group** logic option.
- Assign pins to a synchronous group—I/O pins that are part of a synchronous group (signals that switch at the same time) may cause SSN, but do not result in any failures because the noise glitch occurs during the switching period of the signal. The noise, therefore, does not occur in the sampling window of that signal. You can assign pins to an output enable group with the **Synchronous Group** logic option. For example, in your design you have a bus with 32 pins that all belong to the same group. In a real operation, the bus switches at the same time, so any voltage noise induced by a pin on its groupmates does not matter, because it does not fall in the sampling window. If you do not assign the bus to a synchronous group, the other 31 pins can act as aggressors for the first pin in that group, leading to higher QL and QH noise levels during SSN analysis.



In some cases, the SSN Analyzer can detect the grouping for bidirectional pins by looking at the output enable signal of the bidirectional pins. However, Altera recommends that you explicitly specify the bidirectional groups and output groups in your design.

3.7.7 Excluding Pins as Aggressor Signals

The SSN Analyzer uses the following conditions to exclude pins as aggressor signals for a specific victim pin:

- A pin that is a complement of the victim pin. For example, any pin that is assigned a differential I/O standard cannot be an aggressor pin.
- A programming pin or JTAG pin because these pins are not active in user mode.
- Pins that have the same output enable signal as a bidirectional victim pin that the SSN Analyzer analyzes as an input pin. Pins with the same output enable signal also act as input pins and therefore cannot be aggressor pins at the same time.
- Pins in the same synchronous group as a victim output pin.
- A pin assigned the **I/O Maximum Toggle Rate** logic option with a frequency setting of zero. The SSN Analyzer does not consider pins with this setting as aggressor pins.

3.8 Performing SSN Analysis and Viewing Results

You can perform SSN analysis either on your entire design, or you can limit the analysis to specific I/O banks.

If you know the problem area for SSN is within one I/O bank and you are changing pin assignments only in that bank, you can run SSN analysis for just that one I/O bank to reduce analysis time.

Related Links

[Literature and Technical Documentation](#)

For more information about I/O bank numbering refer to the appropriate device handbook available on the Altera website.

3.8.1 Understanding the SSN Reports

When SSN analysis is complete, you can view detailed analysis reports. The detailed messages in the reports help you understand and resolve SSN problems.

The SSN Analyzer section of the Compilation report contains information generated during analysis, including the following reports:

[Summary Report](#) on page 62

[Output Pins Report and Input Pins Report](#) on page 62

[Unanalyzed Pins Report](#) on page 62

[Confidence Metric Details](#) on page 62



3.8.1.1 Summary Report

The Summary report summarizes the SSN Analyzer status and rates the SSN Analyzer confidence level as low, medium, or high.

The confidence level depends on the completeness of your board trace model assignments. The more assignments you complete, the higher the confidence level. However, the confidence level does not always contribute to the accuracy of the QL and QH noise levels on a victim pin. The accuracy of QH and QL noise levels depends on the accuracy of your board trace model assignments.

3.8.1.2 Output Pins Report and Input Pins Report

The Output Pins report lists all the output pins and bidirectional pins that the SSN analyzer considers as outputs. The Input Pins report lists all the input pins and bidirectional pins that the SSN analyzer considers as inputs. Both reports list:

- Location assignments for the pins.
- QL and QH noise in volts.
- For the I/O standard used for that signal, what percentage the QL and QH margins are.

The QH and QL noise margins that fall in the critical range (> 90%) are shown in red. The QH and QL noise margins that fall in the range of 70% to 90% are shown in gray.

3.8.1.3 Unanalyzed Pins Report

The SSN analyzer doesn't analyze all pins. The ignored pins are reported in the Unanalyzed Pins report:

- Pins assigned the **LVDS** I/O standard or any LVDS variations, such as the **mini-LVDS** I/O standard.
- Pins created in the migration flow that cover power and supply pins in other packages.
- The negative terminals of pseudo-differential I/O standards; the noise on differential standards is reported as the differential noise and is reported on the positive terminal.

3.8.1.4 Confidence Metric Details

The Confidence Metric Details Report lists the values the SSN analyzer uses for unspecified I/O, board, and PCB assignments.

3.8.2 Viewing SSN Analysis Results in the Pin Planner

After SSN analysis completes, you can analyze the results in the Pin Planner. In the Pin Planner you can identify the SSN hotspots in your device, as well as the QL and QH noise levels.

The QL and QH results for each pin are displayed with a different color that represents whether the pin is below the warning threshold, below the critical threshold, or above the critical threshold. This color representation is also referred to as the SSN map of your FPGA device.



When you view the SSN map, you can customize which details to display, including input pins, output pins, QH signals, QL signals, and noise levels. You can also adjust the threshold levels for QH and QL noise voltages. Adjusting the threshold levels in the Pin Planner does not change the threshold levels reported during SSN analysis and does not change the data in any of the SSN reports.

You can also you change I/O assignments and board trace information and rerun the SSN Analyzer to view the SSN analysis results based on those modified settings.

Related Links

[Show SSN Analyzer Results](#)

3.9 Decreasing Processing Time for SSN Analysis

FPGA designs are getting larger in density, logic, and I/O count. The time it takes to complete SSN analysis and other Intel Quartus Prime software processes affects your development time.

Faster processing times can reduce your design cycle time. Use the following guidelines to reduce processing time:

- Direct the Intel Quartus Prime software to use more than one processor for parallel executables, including the SSN Analyzer
- Perform SSN analysis after I/O assignment analysis if your design files and constraints are complete, and you are interested in generating the SSN results early in the design process and want to adjust I/O placements to see if you can obtain better results
- Perform SSN analysis after fitting if you want to view preliminary SSN results that do not take into account complete I/O assignment and I/O timing results
- Perform engineering change orders (ECOs) on your design, rather than recompiling the entire design, if you want to rerun SSN analysis after changing I/O assignments

Related Links

- [Compilation Process Settings Page](#)
For more information about using parallel processors, refer to Intel Quartus Prime Help.
- [Engineering Change Management with the Chip Planner](#) on page 318
For more information about performing ECOs on your design, refer to the *Intel Quartus Prime Handbook*.

3.10 Scripting Support

A Tcl script allows you to run procedures and determine settings. You can also run some of these procedures at a command prompt.

The Intel Quartus Prime software provides several packages to compile your design and create I/O assignments for analysis and fitting. You can create a custom Tcl script that maps the design and runs SSN analysis on your design.



For detailed information about specific scripting command options and Tcl API packages, type the following command at a system command prompt to run the Intel Quartus Prime Command-Line and Tcl API Help browser:

```
quartus_sh --qhelp
```

Related Links

- [Tcl Scripting](#) on page 81
- [Command-Line Scripting](#) on page 67
For more information about Intel Quartus Prime scripting support, including examples, refer to the *Intel Quartus Prime Handbook*.
- [API Functions for Tcl](#)
For more information about Intel Quartus Prime scripting support, including examples, refer to Intel Quartus Prime Help.

3.10.1 Optimizing Pin Placements for Signal Integrity

You can create an assignment that directs the Fitter to optimize pin placements for signal integrity with a Tcl command.

The following Tcl command directs the Fitter to optimize pin placement for signal integrity without affecting design f_{MAX} :

```
set_global_assignment -name OPTIMIZE_SIGNAL_INTEGRITY "Normal Compilation"
```

Related Links

[Optimizing Pin Placements for Signal Integrity](#) on page 55

3.10.2 Defining PCB Layers and PCB Layer Thickness

You can create PCB layer and thickness assignments with a Tcl command.

Tcl Commands for Specifying PCB Layer Assignments

```
set_global_assignment -name PCB_LAYER_THICKNESS 0.00099822M -section_id 1
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 2
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 3
set_global_assignment -name PCB_LAYER_THICKNESS 0.00055372M -section_id 4
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 5
set_global_assignment -name PCB_LAYER_THICKNESS 0.00034036M -section_id 6
set_global_assignment -name PCB_LAYER_THICKNESS 0.00082042M -section_id 7
```

These Tcl commands specify that there are seven PCB layers in the design, each with a different thickness. In each assignment, the letter M indicates the unit of measurement is millimeters. When you specify PCB layer assignments with Tcl commands, you must list the layers in consecutive order. For example, you would receive an error during SSN Analysis if your Tcl commands created the following assignments:

```
set_global_assignment -name PCB_LAYER_THICKNESS 0.00099822M -section_id 1
set_global_assignment -name PCB_LAYER_THICKNESS 0.00082042M -section_id 7
```




To create assignments with the unit of measurement in mils, refer to the syntax in the following Tcl commands.

```
set_global_assignment -name PCB_LAYER_THICKNESS 14.9MIL -section_id 1
set_global_assignment -name PCB_LAYER_THICKNESS 6.6MIL -section_id 2
set_global_assignment -name PCB_LAYER_THICKNESS 14MIL -section_id 3
set_global_assignment -name PCB_LAYER_THICKNESS 6.6MIL -section_id 4
```

Related Links

[Defining PCB Layers and PCB Layer Thickness](#) on page 57

3.10.3 Specifying Signal Breakout Layers

You can create signal breakout layer assignments with a Tcl command.:

```
set_instance_assignment -name PCB_LAYER 10 -to e[2] set_instance_assignment -
name PCB_LAYER 3 -to e[3]
```

When you create PCB breakout layer assignments with Tcl commands, if you do not specify a PCB layer, or if you specify a PCB layer that does not exist, the SSN Analyzer breaks out the signal at the bottommost PCB layer.

Note: If you create a PCB layer breakout assignment to a layer that does not exist, the SSN Analyzer will generate a warning message.

3.10.4 Decreasing Pessimism in SSN Analysis

You can create output enable group and synchronous group assignments to help decrease pessimism during SSN Analysis with a Tcl command.

The following Tcl command assigns the bidirectional bus DATAINOUT to an output enable group:

```
set_instance_assignment -name OUTPUT_ENABLE_GROUP 1 -to DATAINOUT
```

The following Tcl command assigns the bus PCI_ADD_io to a synchronous group:

```
set_instance_assignment -name SYNCHRONOUS_GROUP 1 -to PCI_AD_io
```

Related Links

[Decreasing Pessimism in SSN Analysis](#) on page 60

3.10.5 Performing SSN Analysis

You can perform SSN analysis with a command-line command. Use the `quartus_si` package that is provided with the Intel Quartus Prime software.

Type the following command at a system command prompt to start the SSN Analyzer:

```
quartus_si <project name>
```

To analyze just one I/O bank, type the following command at a system command prompt:

```
quartus_si <project revision> <--bank = bank id>
```



For example, to run analyze the I/O bank 2A type the following command:

```
quartus_si counter --bank=2A
```

For more information about the `quartus_si` package, type `quartus_si -h` at a system command prompt.

Related Links

[Performing SSN Analysis and Viewing Results](#) on page 61

3.11 Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
December 2014	14.1.0	<ul style="list-style-type: none">Minimal text edits for clarity in the topic about understanding SSN.
June 2014	14.0.0	Updated format.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update
December 2010	10.0.1	Template update
July 2010	10.0.0	<ul style="list-style-type: none">Reorganized and edited the chapterAdded links to Intel Quartus Prime Help for procedural information previously included in the chapter
November 2009	9.1.0	<ul style="list-style-type: none">Added "Figure 6–9 shows the layout cross-section of a PCB in the Cadence Allegro PCB tool. The cross-section shows the stackup information of a PCB, which tells you the number of layers used in your PCB. The PCB shown in this example consists of various signal and circuit layers on which FPGA pins are routed, as well as the power and ground layers." on page 6–12Updated for the Intel Quartus Prime software 9.1 release
March 2009	9.0.0	Initial release

Related Links

[Documentation Archive](#)

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



4 Command Line Scripting

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

The command-line executables are completely interchangeable with the Intel Quartus Prime GUI, allowing you to use the exact combination of tools that best suits your needs.

4.1 Benefits of Command-Line Executables

Intel Quartus Prime command-line executables give you precise control over each step of the design flow, reduce memory requirements, and improve performance.

You can group Intel Quartus Prime executable files into a script, batch file, or a makefile to automate design flows. These scripting capabilities facilitate the integration of Intel Quartus Prime software and other EDA synthesis, simulation, and verification software. Automatic design flows can perform on multiple computers simultaneously and easily archive and restore projects.

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

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

Related Links

[About Command-Line Executables](#)
in Intel Quartus Prime Help

4.2 Introductory Example

Create a new Intel Quartus Prime project, fit the design, and generate programming files with this example included with the Intel Quartus Prime software.

If installed, the tutorial design is located in the *<Intel Quartus Prime directory>/qdesigns/fir_filter* directory.



1. Ensure that *<Intel Quartus Prime directory>* /quartus/bin directory is in your PATH environment variable.
2. Copy the tutorial directory in a local folder.
3. In a console, type the four commands in the new project directory:

```
quartus_map filtref --source=filtref.bdf --family="Cyclone V"  
quartus_fit filtref --part=EP3C10F256C8 --pack_register=minimize_area  
quartus_asm filtref  
quartus_sta filtref
```

- a. With the first instruction you create a new Intel Quartus Prime project named **filtref**, set the top-level file as `filtref.bdf`, set Cyclone V as the target device family, and perform logic synthesis and technology mapping on the design files.
 - b. The second command performs place and route by fitting the **filtref** project into the specified device, and directs the Fitter to pack sequential and combinational functions into single logic cells to reduce device resource usage.
 - c. The third command creates a device programming image for the **filtref** project.
 - d. The last line performs basic timing analysis on the **filtref** project using the Intel Quartus Prime Timing Analyzer, reporting worst-case setup slack, worst-case hold slack, and other measurements.
4. Create a batch file or script file with the commands, like the UNIX shell script below:

```
#!/bin/sh  
PROJECT=filtref  
TOP_LEVEL_FILE=filtref.bdf  
FAMILY="Cyclone V"  
PART=EP3C10F256C8  
PACKING_OPTION=minimize_area  
quartus_map $PROJECT --source=$TOP_LEVEL_FILE --family=$FAMILY  
quartus_fit $PROJECT --part=$PART --pack_register=$PACKING_OPTION  
quartus_asm $PROJECT  
quartus_sta $PROJECT
```

5. Execute the script and compile your project.

Related Links

[Intel Quartus Prime Scripting Reference Manual](#)

4.3 Command-Line Scripting Help

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

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

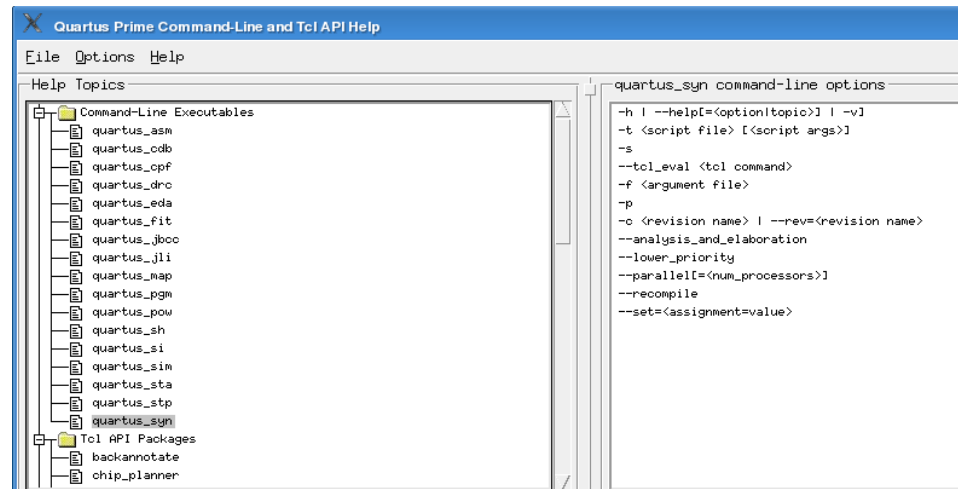
```
quartus_sh --qhhelp
```

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



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

Figure 24. Intel Quartus Prime Command-Line and Tcl API Help Browser



4.4 Project Settings with Command-Line Options

The Intel Quartus Prime software command-line executables accept arguments to set project variables and access common settings.

To make assignments to an individual entity you can use the Intel Quartus Prime Tcl scripting API. On existing projects, you can also open the project in the Intel Quartus Prime GUI, change the assignment, and close the project. The changed assignment is updated in the `.qsf`. Any command-line executables that are run after this update use the updated assignment.

Related Links

- [Tcl Scripting](#) on page 81
- [Intel Quartus Prime Settings File \(.qsf\) Definition](#) in Intel Quartus Prime Help
- [Intel Quartus Prime Standard Edition Settings File Reference Manual](#)
For information about all settings and constraints in the Intel Quartus Prime software.

4.4.1 Option Precedence

Project assignments follow a set of precedence rules. Assignments for a project can exist in three places:

- Intel Quartus Prime Settings File (`.qsf`)
- The compiler database
- Command-line options

The `.qsf` file contains all the project-wide and entity-level assignments and settings for the current revision for the project. The compiler database contains the result of the last compilation in the `/db` directory, and reflects the assignments at the moment when the project was compiled. Updated assignments first appear in the compiler database and later in the `.qsf` file.

Command-line options override any conflicting assignments in the `.qsf` file or the compiler database files. To specify whether the `.qsf` or compiler database files take precedence for any assignments not specified in the command-line, use the option `--read_settings_files`.

Table 15. Precedence for Reading Assignments

Option Specified	Precedence for Reading Assignments
<code>--read_settings_files = on</code> (Default)	<ol style="list-style-type: none"> 1. Command-line options 2. The <code>.qsf</code> for the project 3. Project database (db directory, if it exists) 4. Intel Quartus Prime software defaults
<code>--read_settings_files = off</code>	<ol style="list-style-type: none"> 1. Command-line options 2. Project database (db directory, if it exists) 3. Intel Quartus Prime software defaults

The `--write_settings_files` command-line option lists the locations to which assignments are written..

Table 16. Location for Writing Assignments

Option Specified	Location for Writing Assignments
<code>--write_settings_files = on</code> (Default)	<code>.qsf</code> file and compiler database
<code>--write_settings_files = off</code>	Compiler database

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

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

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

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

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

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



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

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

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

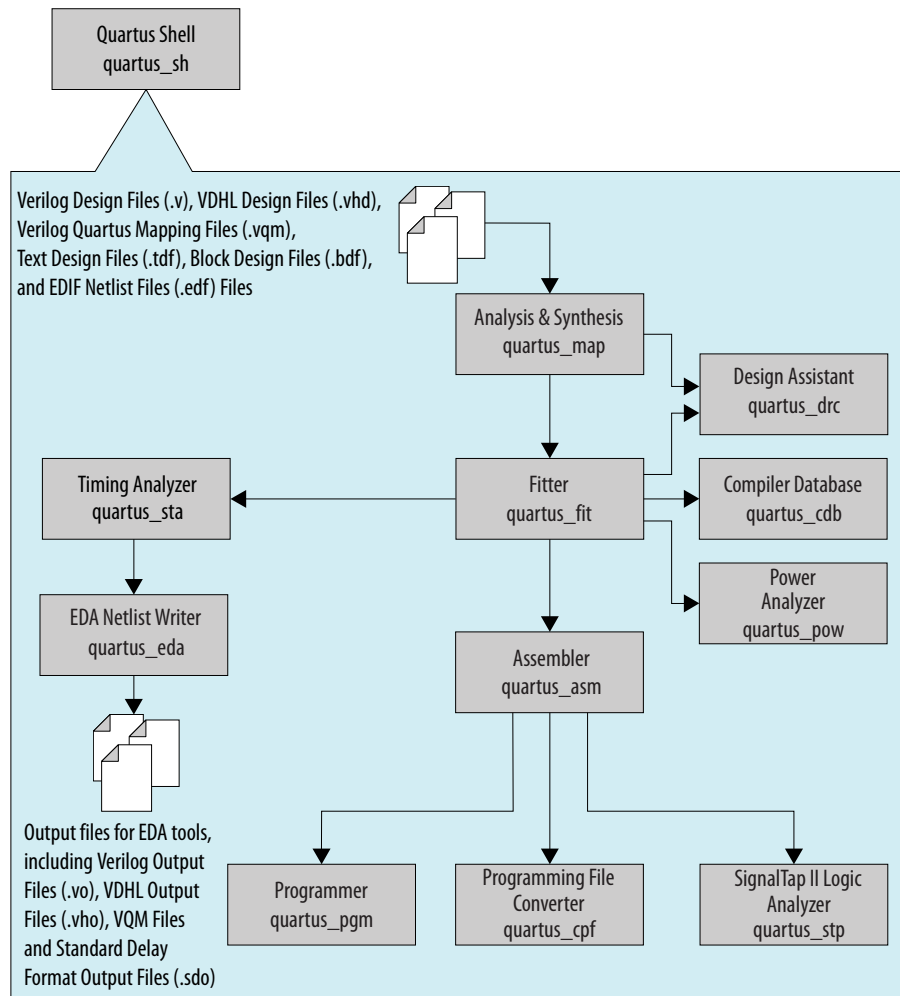
In this example, the `quartus_asm` executable does not read or write settings files:

```
quartus_map filtref --source=filtref --part=EP3C10F256C8
quartus_fit filtref --pack_register=off --read_settings_files=off
quartus_asm filtref --read_settings_files=off --write_settings_files=off
```

4.5 Compilation with `quartus_sh --flow`

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

Figure 25. Typical Design Flow



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

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

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

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

4.6 Text-Based Report Files

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



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

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

Related Links

- [Text-Format Report File \(.rpt\) Definition](#)
in Intel Quartus Prime Help
- [::quartus::report](#)
in Intel Quartus Prime Help

4.7 Using Command-Line Executables in Scripts

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

To set up a new project and apply individual constraints, such as pin location assignments and timing requirements, you must use a Tcl script or the Intel Quartus Prime GUI.

Command-line executables are very useful for working with existing projects, for making common global settings, and for performing common operations. For more flexibility in a flow, use a Tcl script. Additionally, Tcl scripts simplify passing data between different stages of the design flow.

For example, you can create a UNIX shell script to run a third-party synthesis software, place-and-route the design in the Intel Quartus Prime software, and generate output netlists for other simulation software.

This script shows a script that synthesizes a design with the Synopsys Synplify software, simulates the design using the Mentor Graphics ModelSim[®] software, and then compiles the design targeting a Cyclone V device.

```
#!/bin/sh
# Run synthesis first.
# This example assumes you use Synplify software
synplify -batch synthesize.tcl
# If your Quartus Prime project exists already, you can just
# recompile the design.
# You can also use the script described in a later example to
# create a new project from scratch
quartus_sh --flow compile myproject
# Use the quartus_sta executable to do fast and slow-model
# timing analysis
quartus_sta myproject --model=slow
quartus_sta myproject --model=fast
# Use the quartus_eda executable to write out a gate-level
# Verilog simulation netlist for ModelSim
quartus_eda my_project --simulation --tool=modelsim --format=verilog
```



```
# Perform the simulation with the ModelSim software
vlib cycloneV_ver
vlog -work cycloneV_ver /opt/quartusii/eda/sim_lib/cycloneV_atoms.v
vlib work
vlog -work work my_project.vo
vsim -L cycloneV_ver -t lps work.my_project
```

4.8 Common Scripting Examples

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

4.8.1 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.



4.8.2 Check Design File Syntax

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

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

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

```
#!/bin/sh
FILES_WITH_ERRORS=""
# Iterate over each file with a .bdf or .v extension
for filename in `ls *.bdf *.v`
do
# Perform a syntax check on the specified file
  quartus_map fir_filter --analyze_file=$filename
  # If the exit code is non-zero, the file has a syntax error
  if [ $? -ne 0 ]
  then
    FILES_WITH_ERRORS="$FILES_WITH_ERRORS $filename"
  fi
done
if [ -z "$FILES_WITH_ERRORS" ]
then
  echo "All files passed the syntax check"
  exit 0
else
  echo "There were syntax errors in the following file(s)"
  echo $FILES_WITH_ERRORS
  exit 1
fi
```

4.8.3 Create a Project and Synthesize a Netlist Using Netlist Optimizations

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

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

```
quartus_map top --source=top.edf --enable_register_retiming=on
--enable_wysiwyg_resynthesis=on --part=EP3C10F256C8
```

4.8.4 Archive and Restore Projects

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

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

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

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

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

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

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

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

Related Links

[Managing Intel Quartus Prime Projects](#)

In Intel Quartus Prime Standard Edition Handbook Volume 1

4.8.5 Perform I/O Assignment Analysis

You can perform I/O assignment analysis with a single command. I/O assignment analysis checks pin assignments to ensure they do not violate board layout guidelines. I/O assignment analysis does not require a complete place and route, so it can quickly verify that your pin assignments are correct.

```
quartus_fit --check_ios <project name> --rev=<revision name>
```

4.8.6 Update Memory Contents Without Recompiling

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

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



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

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

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

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

4.8.7 Create a Compressed Configuration File

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

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

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

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

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

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

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

4.8.8 Fit a Design as Quickly as Possible

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

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

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

4.8.9 Fit a Design Using Multiple Seeds

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

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

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

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

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

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

4.9 The QFlow Script

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



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

- quartus_map (Analysis and Synthesis)
- quartus_fit (Fitter)
- quartus_sta (Timing Analyzer)
- quartus_asm (Assembler)
- quartus_eda (EDA Netlist Writer)

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

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

```
quartus_sh -g
```

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

4.10 Document Revision History

Table 17. Document Revision History

Date	Version	Changes
2017.05.08	17.0.0	<ul style="list-style-type: none"> • Reorganized content on topics: Benefits of Command-Line Executables and Project Settings with Command-Line Options.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
2015.05.04	15.0.0	Remove descriptions of makefile support that was removed from software in 14.0.
December 2014	14.1.0	Updated DSE II commands.
June 2014	14.0.0	Updated formatting.
November 2013	13.1.0	Removed information about -silnet qmegawiz command
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	Corrected quartus_qpf example usage. Updated examples.
December 2010	10.1.0	Template update. Added section on using a script to regenerate megafunction variations. Removed references to the Classic Timing Analyzer (quartus_tan). Removed Qflow illustration.
July 2010	10.0.0	Updated script examples to use quartus_sta instead of quartus_tan, and other minor updates throughout document.
November 2009	9.1.0	Updated Table 2-1 to add quartus_jli and quartus_jbcc executables and descriptions, and other minor updates throughout document.
March 2009	9.0.0	No change to content.
November 2008	8.1.0	Added the following sections:

continued...



Date	Version	Changes
		<ul style="list-style-type: none">• "The MegaWizard Plug-In Manager" on page 2-11• "Command-Line Support" on page 2-12• "Module and Wizard Names" on page 2-13• "Ports and Parameters" on page 2-14• "Invalid Configurations" on page 2-15• "Strategies to Determine Port and Parameter Values" on page 2-15• "Optional Files" on page 2-15• "Parameter File" on page 2-16• "Working Directory" on page 2-17• "Variation File Name" on page 2-17• "Create a Compressed Configuration File" on page 2-21• Updated "Option Precedence" on page 2-5 to clarify how to control precedence• Corrected Example 2-5 on page 2-8• Changed Example 2-1, Example 2-2, Example 2-4, and Example 2-7 to use the EP1C12F256C6 device• Minor editorial updates• Updated entire chapter using 8½" × 11" chapter template
May 2008	8.0.0	<ul style="list-style-type: none">• Updated "Referenced Documents" on page 2-20.• Updated references in document.

Related Links

Documentation Archive

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



5 Tcl Scripting

5.1 Tcl Scripting

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

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

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

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

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

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

Related Links

[Tcl Design Examples](#)

5.2 Tool Command Language

Tcl (pronounced "tickle") stands for Tool Command Language, and is the industry-standard scripting language. Tcl supports control structures, variables, network socket access, and APIs.

With Tcl, you can work seamlessly across most development platforms. Synopsys, Mentor Graphics, and Intel software products support the Tcl language.



By combining Tcl commands and Intel Quartus Prime API functions, you can create your own procedures and automate your design flow. Run Intel Quartus Prime software in batch mode, or execute individual Tcl commands interactively in the Intel Quartus Prime Tcl shell.

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

Related Links

- [External References](#) on page 103
- [Tcl Scripting Basics](#) on page 97
- tcl.activestate.com

5.3 Intel Quartus Prime Tcl Packages

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

Table 18. Intel Quartus Prime Tcl Packages

Package Name	Package Description
backannotate	Back annotate assignments
chip_planner	Identify and modify resource usage and routing with the Chip Editor
database_manager	Manage version-compatible database files
device	Get device and family information from the device database
external_memif_toolkit	Interact with external memory interfaces and debug components
fif	Contains the set of Tcl functions for using the Fault Injection File (FIF) Driver
flow	Compile a project, run command-line executables, and other common flows
incremental_compilation	Manipulate design partitions and Logic Lock (Standard) regions, and settings related to incremental compilation
insystem_memory_edit	Read and edit memory contents in Intel devices
insystem_source_probe	Interact with the In-System Sources and Probes tool in an Intel device
interactive_synthesis	
iptclgen	Generate memory IP
jtag	Control the JTAG chain
logic_analyzer_interface	Query and modify the Logic Analyzer Interface output pin state
misc	Perform miscellaneous tasks such as enabling natural bus naming, package loading, and message posting
names	
partial_reconfiguration	Contain the set of Tcl functions for performing partial reconfiguration
project	Create and manage projects and revisions, make any project assignments including timing assignments
report	Get information from report tables, create custom reports
rtl	Traverse and query the RTL netlist of your design
sdc	Specify constraints and exceptions to the Timing Analyzer
<i>continued...</i>	



Package Name	Package Description
sd_c_ext	Intel-specific SDC commands
simulator	Configure and perform simulations
sta	Contain the set of Tcl functions for obtaining advanced information from the Timing Analyzer
stp	Run the Signal Tap Logic Analyzer
synthesis_report	Contain the set of Tcl functions for the Dynamic Synthesis Report tool
tdc	Obtain information from the Timing Analyzer

To keep memory requirements as low as possible, only the minimum number of packages load automatically with each Intel Quartus Prime executable. To run commands from other packages, load those packages beforehand.

Run your scripts with executables that include the packages you use in the scripts. For example, to use commands in the `sd_c_ext` package, you must use the `quartus_sta` executable because `quartus_sta` is the only executable with support for the `sd_c_ext` package.

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

```
<executable name> --tcl_eval help
```

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

```
quartus_fit --tcl_eval help
```

5.3.1 Loading Packages

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

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

This command is similar to `package require`, but it allows to alternate between different versions of an Intel Quartus Prime Tcl package.

Related Links

[Command Line Scripting](#) on page 67

5.4 Intel Quartus Prime Tcl API Help

You can access the Intel Quartus Prime Tcl API Help by typing the following at a system command prompt:

```
quartus_sh --qhelp
```

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



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

Tcl Help Output

```
tcl> help
-----
Available Intel Quartus Prime Tcl Packages:
-----
Loaded                Not Loaded
-----
::quartus::device    ::quartus::external_memif_toolkit
::quartus::misc      ::quartus::iptclgen
::quartus::project   ::quartus::design
                    ::quartus::rtm
                    ::quartus::partial_reconfiguration
                    ::quartus::report
                    ::quartus::names
                    ::quartus::incremental_compilation
                    ::quartus::flow

* Type "help -tcl"
to get an overview on Intel Quartus Prime Tcl usages.

* Type "help <package name>"
to view a list of Tcl commands available for
the specified Intel Quartus Prime Tcl package.
```

Table 19. Help Options Available in the Intel Quartus Prime Tcl Environment

Help Command	Description
<code>help</code>	To view the complete list of available Intel Quartus Prime Tcl packages.
<code>help -tcl</code>	To view how to load Tcl packages and access command-line help.
<code>help -pkg <package_name> [-version <version number>]</code>	To view the help of a given Intel Quartus Prime package, including the list of available Tcl commands. For convenience, you can omit the <code>::quartus::</code> package prefix, and type <code>help -pkg <package name></code> . If you do not specify the <code>-version</code> option, help for the currently loaded package appears by default. If the package is not loaded, help for the latest version of the package appears by default. Examples: <pre>help -pkg ::quartus::project</pre> <pre>help -pkg project help -pkg project -version 1.0</pre>
<code><command_name>-h</code> or <code><command_name> -help</code>	To view the short help of a Intel Quartus Prime Tcl command in a loaded package. Examples: <pre>project_open -h</pre> <pre>project_open -help</pre>
<code>package require ::quartus::<package name> [<version>]</code>	To load a specific version of an Intel Quartus Prime Tcl package. If <code><version></code> is not specified, the latest version of the package loads by default. Example: <pre>package require ::quartus::project 1.0</pre> <p>This command is similar to the <code>load_package</code> command.</p> <p>The advantage of the <code>load_package</code> command is that you can alternate freely between different versions of the same package.</p> <p>Type <code>load_package <package name> [-version <version number>]</code> to load an Intel Quartus Prime Tcl package with the specified version. If the <code>-version</code> option is not specified, the latest version of the package is loaded by default.</p>

continued...



Help Command	Description
	Example: load_package ::quartus::project -version 1.0
help -cmd<command_name> [-version <version>] or <command_name> -long_help	To view complete help text for an Intel Quartus Prime Tcl command. If you do not specify the -version option, help for the command in the currently loaded package version appears by default. If the package version for which you want help is not loaded, help for the latest version of the package appears by default. Examples: <pre>project_open -long_help</pre> <pre>help -cmd project_open</pre> <pre>help -cmd project_open -version 1.0</pre>
help -examples	To view examples of Intel Quartus Prime Tcl usage.
help -quartus	To view help on the predefined global Tcl array that contains project information and information about the Intel Quartus Prime executable that is currently running.
quartus_sh --qhelp	To launch the Tk viewer for Intel Quartus Prime command-line help and display help for the command-line executables and Tcl API packages.
help -timequestinfo	To view help on the predefined global "TimeQuestInfo" Tcl array that contains delay model information and speed grade information of a Timing Analyzer design that is currently running.

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

5.4.1 Command-Line Options

You can use any of the following command line options with executables that support Tcl:

Table 20. Command-Line Options Supporting Tcl Scripting

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

5.4.1.1 Run a Tcl Script

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

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



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

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

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

5.4.1.2 Interactive Shell Mode

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

```
quartus_sta -s
```

Commands you type in the Tcl shell are interpreted when you press Enter. To run a Tcl script in the interactive shell type:

```
source <script name>
```

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

5.4.1.3 Evaluate as Tcl

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

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

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

5.4.2 The Intel Quartus Prime Tcl Console Window

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

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

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



5.5 End-to-End Design Flows

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

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

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

There are no limitations on running flows from any executable. Flows include operations found in

Processing > Start in the Intel Quartus Prime GUI, and are also documented as options for the `execute_flow` Tcl command. If you can make settings in the Intel Quartus Prime software and run a flow to get your desired result, you can make the same settings and run the same flow in a Tcl script.

5.6 Creating Projects and Making Assignments

You can create a script that makes all the assignments for an existing project, and then use the script at any time to restore your project settings to a known state.

Click **Project > Generate Tcl File for Project** to automatically generate a `.tcl` file containing your assignments. You can source this file to recreate your project, and you can add other commands to this file, such as commands for compiling the design. This file is a good starting point to learn about project management and assignment commands.

To commit the assignments you create or modify to the `.qsf` file, you use the `export_assignments` or `project_close` commands. However, when you run the `execute_flow` command, Intel Quartus Prime software automatically commits the assignment changes to the `.qsf` file. To prevent this behavior, specify the `-dont_export_assignments` logic option.

Example 13. Create and Compile a Project

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

```
load_package flow
# Create the project and overwrite any settings
# files that exist
project_new fir_filter -revision filtref -overwrite
# Set the device, the name of the top-level BDF,
```

```
# and the name of the top-level entity
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C6F256C6
set_global_assignment -name BDF_FILE filtref.bdf
set_global_assignment -name TOP_LEVEL_ENTITY filtref
# Add other pin assignments here
set_location_assignment -to clk Pin_G1
# compile the project
execute_flow -compile
project_close
```

Related Links

- [Interactive Shell Mode](#) on page 86
- [Constraining Designs](#) on page 11
- [Intel Quartus Prime Settings File Reference Manual](#)

5.7 Compiling Designs

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

5.7.1 The `flow` Package

The `flow` package includes two commands for running Intel Quartus Prime command-line executables, either individually or together in standard compilation sequence.

- The `execute_module` command allows you to run an individual Intel Quartus Prime command-line executable.
- The `execute_flow` command allows you to run some or all the executables in commonly-used combinations.

Use the `flow` package instead of system calls to run Intel Quartus Prime executables from scripts or from the Intel Quartus Prime Tcl Console.

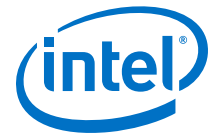
5.7.2 Compile All Revisions

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

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

Compile All Revisions

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

5.8 Reporting

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

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

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

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

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

Print All Report Panel Names

```
load_package report
project_open myproject
load_report
set panel_names [get_report_panel_names]
foreach panel_name $panel_names {
    post_message "$panel_name"
}
```

5.8.1 Saving Report Data in csv Format

You can create a Comma Separated Value (`.csv`) file from any Intel Quartus Prime report to open with a spreadsheet editor.

The following Tcl code shows a simple way to create a `.csv` file with data from the Fitter panel in a report.

Create .csv Files from Reports

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

```
puts $fh [join $row_data " , " ]  
}  
close $fh  
unload_report
```

You can modify the script to use command-line arguments to pass in the name of the project, report panel, and output file to use. You can run this script example with any executable that supports the report package.

5.9 Timing Analysis

The Intel Quartus Prime Timing Analyzer includes support for industry-standard SDC commands in the `sdc` package.

The Intel Quartus Prime software includes comprehensive Tcl APIs and SDC extensions for the Timing Analyzer in the `sta`, and `sdc_ext` packages. The Intel Quartus Prime software also includes a `tdc` package that obtains information from the Timing Analyzer.

Related Links

[Intel Quartus Prime Standard Edition Settings File Reference Manual](#)

For information about all settings and constraints in the Intel Quartus Prime software.

5.10 Automating Script Execution

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

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

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

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

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

1. Analysis and Synthesis (`quartus_map`)
2. Fitter (`quartus_fit`)
3. Assembler (`quartus_asm`)
4. Timing Analyzer (`quartus_sta`)

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



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

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

The Intel Quartus Prime software runs the scripts.

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

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

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

5.10.1 Execution Example

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

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

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

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

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

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

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

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

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

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

5.10.2 Controlling Processing

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

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

Restrict Processing to a Single Module

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

5.10.3 Displaying Messages

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

Related Links

- [The `post_message` Command](#) on page 94
- [Automating Script Execution](#) on page 90

5.11 Other Scripting Features

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

5.11.1 Natural Bus Naming

The Intel Quartus Prime software supports natural bus naming. Natural bus naming allows to use square brackets to specify bus indexes in HDL without including escape characters to prevent Tcl from interpreting the square brackets as containing commands. For example, one signal in a bus named `address` can be identified as `address[0]` instead of `address\[0\]`. You can take advantage of natural bus naming when making assignments.

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

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

```
enable_natural_bus_naming -h
```



5.11.2 Short Option Names

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

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

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

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

5.11.3 Collection Commands

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

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

5.11.3.1 The `foreach_in_collection` Command

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

`foreach_in_collection` Example

```
set all_instance_assignments [get_all_instance_assignments -name *]
foreach_in_collection asgn $all_instance_assignments {
    # Information about each assignment is
    # returned in a list. For information
    # about the list elements, refer to Help
    # for the get-all-instance-assignments command.
    set to [lindex $asgn 2]
    set name [lindex $asgn 3]
    set value [lindex $asgn 4]
    puts "Assignment to $to: $name = $value"
}
```

Related Links

[foreach_in_collection \(::quartus::misc\)](#)
In Intel Quartus Prime Help

5.11.3.2 The `get_collection_size` Command

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

`get_collection_size` Example

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

5.11.4 The `post_message` Command

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

The message type can be one of the following:

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

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

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

```
DISPLAY_COMMAND_LINE_MESSAGES_IN_COLOR = on
```

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

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

5.11.5 Accessing Command-Line Arguments

The global variable `quartus(args)` is a list of the arguments typed on the command-line following the name of the Tcl script.

Example 14. Simple Command-Line Argument Access

The following Tcl example prints all the arguments in the `quartus(args)` variable:

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



Example 15. Passing Command-Line Arguments to Scripts

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

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

5.11.5.1 The cmdline Package

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

cmdline Package

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

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

Passing Command-Line Arguments for Scripts

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

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

Full-Featured Method to Open Projects

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



```
} else {  
    puts "Project $optshash(project) does not exist"  
    exit 1  
}  
# The rest of your script goes here
```

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

Simple Method to Open Projects

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

5.11.6 The `quartus()` Array

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

```
help -quartus
```

5.12 The Intel Quartus Prime Tcl Shell in Interactive Mode Example

This section presents how to make project assignments and then compile the finite impulse response (FIR) filter tutorial project with the `quartus_sh` interactive shell.

This example assumes you already have the `fir_filter` tutorial design files in a project directory.

1. To run the interactive Tcl shell, type the following at the system command prompt:

```
quartus_sh -s
```

2. Create a new project called `fir_filter`, with a revision called `filtref` by typing:

```
project_new -revision filtref fir_filter
```

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

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

3. Set a global assignment for the device:

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

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



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

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

```
load_package flow
```

It returns:

```
1.1
```

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

```
execute_flow -compile
```

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

6. When you are finished with a project, close it with the `project_close` command.
7. To exit the interactive Tcl shell, type `exit` at a Tcl prompt.

Related Links

[set_global_assignment \(::quartus::project\)](#)
In Intel Quartus Prime Help

5.13 The tclsh Shell

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

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

5.14 Tcl Scripting Basics

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

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

5.14.1 Hello World Example

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

```
puts "Hello world"
```

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

5.14.2 Variables

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

```
set a 1
```

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

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

5.14.3 Substitutions

Tcl performs three types of substitution:

- Variable value substitution
- Nested command substitution
- Backslash substitution

5.14.3.1 Variable Value Substitution

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

5.14.3.2 Nested Command Substitution

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

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



5.14.3.3 Backslash Substitution

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

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

5.14.4 Arithmetic

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

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

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

5.14.5 Lists

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

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

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

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

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

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

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

```
lappend a 4 5 6
```

5.14.6 Arrays

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

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

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

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

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

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

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

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

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

5.14.7 Control Structures

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

If-Then-Else Structure

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

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

For Loop

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

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



foreach Loop

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

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

while Loop

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

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

5.14.8 Procedures

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

Simple Procedure

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

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

Using a Procedure

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

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

Accessing Global Variables

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

5.14.9 File I/O

Tcl includes commands to read from and write to files. You must open a file before you can read from or write to it, and close it when the read and write operations are done.

To open a file, use the `open` command; to close a file, use the `close` command. When you open a file, specify its name and the mode in which to open it. If you do not specify a mode, Tcl defaults to read mode. To write to a file, specify `w` for write mode.

Open a File for Writing

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

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

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

Write to a File

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

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

Read from a File

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

5.14.10 Syntax and Comments

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

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

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

Without the semicolon, the command is invalid because the `set` command does not terminate until the new line after the comment.



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

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

5.14.11 External References

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

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

Related Links

- [Intel Quartus Prime Tcl Examples](#)
For Intel Quartus Prime Tcl example scripts
- tcl.activestate.com
Tcl Developer Xchange

5.15 Document Revision History

Table 21. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	<ul style="list-style-type: none"> • Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>. • Updated the list of Tcl packages in the <i>Intel Quartus Prime Tcl Packages</i> section. • Updated the <i>Intel Quartus Prime Tcl API Help</i> section: <ul style="list-style-type: none"> — Updated the Tcl Help Output
June 2014	14.0.0	Updated the format.
June 2012	12.0.0	<ul style="list-style-type: none"> • Removed survey link.
November 2011	11.0.1	<ul style="list-style-type: none"> • Template update • Updated supported version of Tcl in the section "Tool Command Language." • minor editorial changes
May 2011	11.0.0	Minor updates throughout document.
December 2010	10.1.0	Template update Updated to remove tcl packages used by the Classic Timing Analyzer
July 2010	10.0.0	Minor updates throughout document.
November 2009	9.1.0	<ul style="list-style-type: none"> • Removed LogicLock example. • Added the incremental_compilation, insystem_source_probe, and rtl packages to Table 3-1 and Table 3-2. • Added quartus_map to table 3-2.
<i>continued...</i>		

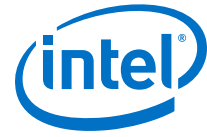


Date	Version	Changes
March 2009	9.0.0	<ul style="list-style-type: none">• Removed the "EDA Tool Assignments" section• Added the section "Compile All Revisions" on page 3-9• Added the section "Using the tcsh Shell" on page 3-20
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	Updated references.

Related Links

Documentation Archive

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



6 Signal Integrity Analysis with Third-Party Tools

6.1 Signal Integrity Analysis with Third-Party Tools

With the ever-increasing operating speed of interfaces in traditional FPGA design, the timing and signal integrity margins between the FPGA and other devices on the board must be within specification and tolerance before a single PCB is built.

If the board trace is designed poorly or the route is too heavily loaded, noise in the signal can cause data corruption, while overshoot and undershoot can potentially damage input buffers over time.

As FPGA devices are used in high-speed applications, signal integrity and timing margin between the FPGA and other devices on the printed circuit board (PCB) are important aspects to consider to ensure proper system operation. To avoid time-consuming redesigns and expensive board respins, the topology and routing of critical signals must be simulated. The high-speed interfaces available on current FPGA devices must be modeled accurately and integrated into timing models and board-level signal integrity simulations. The tools used in the design of an FPGA and its integration into a PCB must be “board-aware”—able to take into account properties of the board routing and the connected devices on the board.

The Intel Quartus Prime software provides methodologies, resources, and tools to ensure good signal integrity and timing margin between Intel FPGA devices and other components on the board. Three types of analysis are possible with the Intel Quartus Prime software:

- I/O timing with a default or user-specified capacitive load and no signal integrity analysis (default)
- The Intel Quartus Prime **Enable Advanced I/O Timing** option utilizing a user-defined board trace model to produce enhanced timing reports from accurate “board-aware” simulation models
- Full board routing simulation in third-party tools using Intel-provided or generated Input/Output Buffer Information Specification (IBIS) or HSPICE I/O models

I/O timing using a specified capacitive test load requires no special configuration other than setting the size of the load. I/O timing reports from the Intel Quartus Prime Timing Analyzer or the Intel Quartus Prime Classic Timing Analyzer are generated based only on point-to-point delays within the I/O buffer and assume the presence of the capacitive test load with no other details about the board specified. The default size of the load is based on the I/O standard selected for the pin. Timing is measured to the FPGA pin with no signal integrity analysis details.

The **Enable Advanced I/O Timing** option expands the details in I/O timing reports by taking board topology and termination components into account. A complete point-to-point board trace model is defined and accounted for in the timing analysis. This ability to define a board trace model is an example of how the Intel Quartus Prime software is “board-aware.”

In this case, timing and signal integrity metrics between the I/O buffer and the defined far end load are analyzed and reported in enhanced reports generated by the Intel Quartus Prime Timing Analyzer.

The information about signal integrity in this chapter refers to board-level signal integrity based on I/O buffer configuration and board parameters, not simultaneous switching noise (SSN), also known as ground bounce or V_{CC} sag. SSN is a product of multiple output drivers switching at the same time, causing an overall drop in the voltage of the chip's power supply. This can cause temporary glitches in the specified level of ground or V_{CC} for the device.

This chapter is intended for FPGA and board designers and includes details about the concepts and steps involved in getting designs simulated and how to adjust designs to improve board-level timing and signal integrity. Also included is information about how to create accurate models from the Intel Quartus Prime software and how to use those models in simulation software.

The information in this chapter is meant for those who are familiar with the Intel Quartus Prime software and basic concepts of signal integrity and the design techniques and components in good PCB design. Finally, you should know how to set up simulations and use your selected third-party simulation tool.

Related Links

[I/O Management](#) on page 24

For more information about defining capacitive test loads or how to use the **Enable Advanced I/O Timing** option to configure a board trace model.

6.1.1 Signal Integrity Simulations with HSPICE and IBIS Models

The Intel Quartus Prime software can export accurate HSPICE models with the built-in HSPICE Writer. You can run signal integrity simulations with these complete HSPICE models in Synopsys HSPICE. IBIS models of the FPGA I/O buffers are also created easily with the Intel Quartus Prime IBIS Writer.

You can run signal integrity simulations with these complete HSPICE models in Synopsys HSPICE.

You can integrate IBIS models into any third-party simulation tool that supports them, such as the Mentor Graphics HyperLynx software. With the ability to create industry-standard model definition files quickly, you can build accurate simulations that can provide data to help improve board-level signal integrity.

The I/O's IBIS and HSPICE model creation available in the Intel Quartus Prime software can help prevent problems before a costly board respin is required. In general, creating and running accurate simulations is difficult and time consuming. The tools in the Intel Quartus Prime software automate the I/O model setup and creation process by configuring the models specifically for your design. With these tools, you can set up and run accurate simulations quickly and acquire data that helps guide your FPGA and board design.

For a more information about SSN and ways to prevent it, refer to *AN 315: Guidelines for Designing High-Speed FPGA PCBs*.

For information about basic signal integrity concepts and signal integrity details pertaining to Intel FPGA devices, visit the Intel Signal & Power Integrity Center.



Related Links

- [AN 315: Guidelines for Designing High-Speed FPGA PCBs](#)
- [Intel Signal & Power Integrity Center](#)

6.2 I/O Model Selection: IBIS or HSPICE

The Intel Quartus Prime software can export two different types of I/O models that are useful for different simulation situations, IBIS models and HSPICE models.

IBIS models define the behavior of input or output buffers through voltage-current (V-I) and voltage-time (V-t) data tables. HSPICE models, or decks, include complete physical descriptions of the transistors and parasitic capacitances that make up an I/O buffer along with all the parameter settings that you require to run a simulation.

The Intel Quartus Prime software generates HSPICE decks, and adds preconfigured I/O standard, voltage, and pin loading settings for each pin in your design.

The choice of I/O model type is based on many factors.

Table 22. IBIS and HSPICE Model Comparison

Feature	IBIS Model	HSPICE Model
I/O Buffer Description	Behavioral —I/O buffers are described by voltage-current and voltage-time tables in typical, minimum, and maximum supply voltage cases.	Physical —I/O buffers and all components in a circuit are described by their physical properties, such as transistor characteristics and parasitic capacitances, as well as their connections to one another.
Model Customization	Simple and limited —The model completely describes the I/O buffer and does not usually have to be customized.	Fully customizable —Unless connected to an arbitrary board description, the description of the board trace model must be customized in the model file. All parameters of the simulation are also adjustable.
Simulation Set Up and Run Time	Fast —Simulations run quickly after set up correctly.	Slow —Simulations take time to set up and take longer to run and complete.
Simulation Accuracy	Good —For most simulations, accuracy is sufficient to make useful adjustments to the FPGA or board design to improve signal integrity.	Excellent —Simulations are highly accurate, making HSPICE simulation almost a requirement for any high-speed design where signal integrity and timing margins are tight.
Third-Party Tool Support	Excellent —Almost all third-party board simulation tools support IBIS.	Good —Most third-party tools that support SPICE support HSPICE. However, Synopsys HSPICE is required for simulations of Intel's encrypted HSPICE models.

For more information about IBIS files created by the Intel Quartus Prime IBIS Writer and IBIS files in general, as well as links to websites with detailed information, refer to *AN 283: Simulating Intel Devices with IBIS Models*.

Related Links

- [AN 283: Simulating Intel Devices with IBIS Models](#)

6.3 FPGA to Board Signal Integrity Analysis Flow

Board signal integrity analysis can take place at any point in the FPGA design process and is often performed before and after board layout. If it is performed early in the process as part of a pre-PCB layout analysis, the models used for simulations can be more generic.



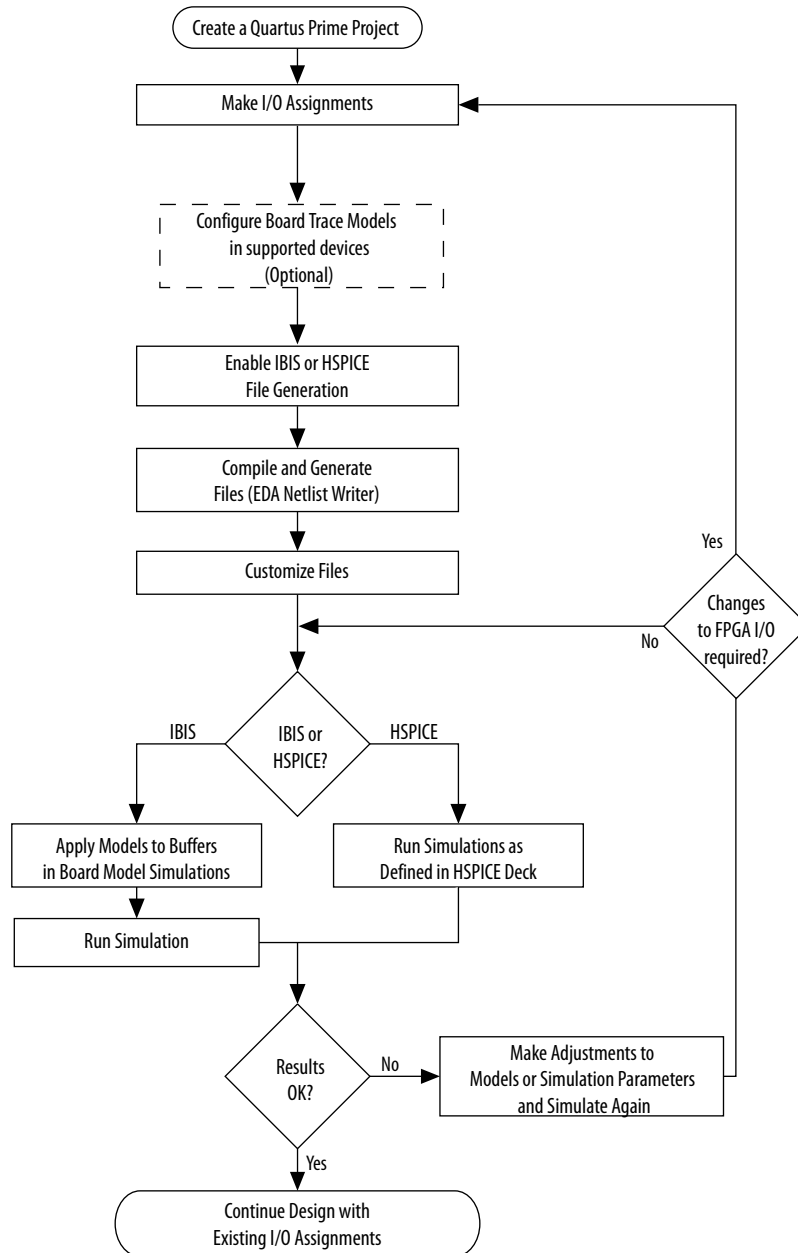
These models can be changed as much as required to see how adjustments improve timing or signal integrity and help with the design and routing of the PCB. Simulations and the resulting changes made at this stage allow you to analyze “what if” scenarios to plan and implement your design better. To assist with early board signal integrity analysis, you can download generic IBIS model files for each device family and obtain HSPICE buffer simulation kits from the “Board Level Tools” section of the EDA Tool Support Resource Center.

Typically, if board signal integrity analysis is performed late in the design, it is used for a post-layout verification. The inputs and outputs of the FPGA are defined, and required board routing topologies and constraints are known. Simulations can help you find problems that might still exist in the FPGA or board design before fabrication and assembly. In either case, a simple process flow illustrates how to create accurate IBIS and HSPICE models from a design in the Intel Quartus Prime software and transfer them to third-party simulation tools.

Your design depends on the type of model, IBIS or HSPICE, that you use for your simulations. When you understand the steps in the analysis flow, refer to the section of this chapter that corresponds to the model type you are using.



Figure 26. Third-Party Board Signal Integrity Analysis Flow



Related Links

[EDA Tool Support Resource Center](#)

For more information, generic IBIS model files for each device family, and to obtain HSPICE buffer simulation kits.

6.3.1 Create I/O and Board Trace Model Assignments

You can configure a board trace model for output signals or for bidirectional signals in output mode. You can then automatically transfer its description to HSPICE decks generated by the HSPICE Writer. This helps improve simulation accuracy.

To configure a board trace model, in the **Settings** dialog box, in the **Timing Analyzer** page, turn on the **Enable Advanced I/O Timing** option and configure the board trace model assignment settings for each I/O standard used in your design. You can add series or parallel termination, specify the transmission line length, and set the value of the far-end capacitive load. You can configure these parameters either in the Board Trace Model view of the Pin Planner, or click **SettingsDeviceDevice and Pin Options**.

The Intel Quartus Prime software can generate IBIS models and HSPICE decks without having to configure a board trace model with the **Enable Advanced I/O Timing** option. In fact, IBIS models ignore any board trace model settings other than the far-end capacitive load. If any load value is set other than the default, the delay given by IBIS models generated by the IBIS Writer cannot be used to account correctly for the double counting problem. The load value mismatch between the IBIS delay and the t_{CO} measurement of the Intel Quartus Prime software prevents the delays from being safely added together. Warning messages displayed when the EDA Netlist Writer runs indicate when this mismatch occurs.

Related Links

[I/O Management](#) on page 24

For information about how to use the **Enable Advanced I/O Timing** option and configure board trace models for the I/O standards used in your design.

6.3.2 Output File Generation

IBIS and HSPICE model files are not generated by the Intel Quartus Prime software by default. To generate or update the files automatically during each project compilation, select the type of file to generate and a location where to save the file in the project settings.

The IBIS and HSPICE Writers in the Intel Quartus Prime software are run as part of the EDA Netlist Writer during normal project compilation. If either writer is turned on in the project settings, IBIS or HSPICE files are created and stored in the specified location. For IBIS, a single file is generated containing information about all assigned pins. HSPICE file generation creates separate files for each assigned pin. You can run the EDA Netlist Writer separately from a full compilation in the Intel Quartus Prime software or at the command line.

Note: You must fully compile the project or perform I/O Assignment Analysis at least once for the IBIS and HSPICE Writers to have information about the I/O assignments and settings in the design.

6.3.3 Customize the Output Files

The files generated by either the IBIS or HSPICE Writer are text files that you can edit and customize easily for design or experimentation purposes.



IBIS files downloaded from the Altera website must be customized with the correct RLC values for the specific device package you have selected for your design. IBIS files generated by the IBIS Writer do not require this customization because they are configured automatically with the RLC values for your selected device. HSPICE decks require modification to include a detailed description of your board. With **Enable Advanced I/O Timing** turned on and a board trace model defined in the Intel Quartus Prime software, generated HSPICE decks automatically include that model's parameters. However, Intel recommends that you replace that model with a more detailed model that describes your board design more accurately. A default simulation included in the generated HSPICE decks measures delay between the FPGA and the far-end device. You can make additions or adjustments to the default simulation in the generated files to change the parameters of the default simulation or to perform additional measurements.

6.3.4 Set Up and Run Simulations in Third-Party Tools

When you have generated the files, you can use them to perform simulations in your selected simulation tool.

With IBIS models, you can apply them to input, output, or bidirectional buffer entities and quickly set up and run simulations. For HSPICE decks, the simulation parameters are included in the files. Open the files in Synopsys HSPICE and run simulations for each pin as required.

With HSPICE decks generated from the HSPICE Writer, the double counting problem is accounted for, which ensures that your simulations are accurate. Simulations that involve IBIS models created with anything other than the default loading settings in the Intel Quartus Prime software must take the change in the size of the load between the IBIS delay and the Intel Quartus Prime t_{CO} measurement into account. Warning messages during compilation alert you to this change.

6.3.5 Interpret Simulation Results

If you encounter timing or signal integrity issues with your high-speed signals after running simulations, you can make adjustments to I/O assignment settings in the Intel Quartus Prime software.

You can adjust drive strength or I/O standard, or make changes to the board routing or topology. After regenerating models in the Intel Quartus Prime software based on the changes you have made, rerun the simulations to check whether your changes corrected the problem.

6.4 Simulation with IBIS Models

IBIS models provide a way to run accurate signal integrity simulations quickly. IBIS models describe the behavior of I/O buffers with voltage-current and voltage-time data curves.

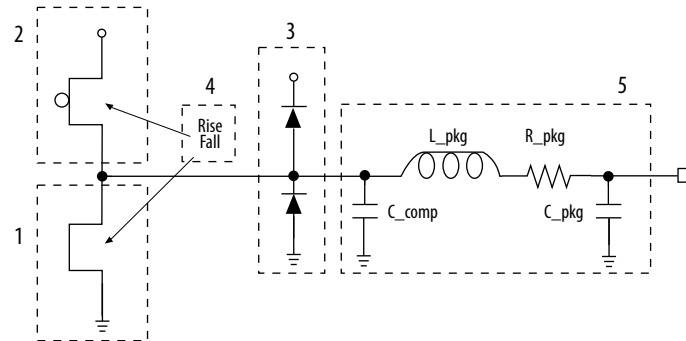
Because of their behavioral nature, IBIS models do not have to include any information about the internal circuit design of the I/O buffer. Most component manufacturers, including Intel, provide IBIS models for free download and use in signal integrity analysis simulation tools. You can download generic device family IBIS models from the Altera website for early design simulation or use the IBIS Writer to create custom IBIS models for your existing design.

6.4.1 Elements of an IBIS Model

An IBIS model file (.**ibs**) is a text file that describes the behavior of an I/O buffer across minimum, typical, and maximum temperature and voltage ranges with a specified test load.

The tables and values specified in the IBIS file describe five basic elements of the I/O buffer.

Figure 27. Five Basic Elements of an I/O Buffer in IBIS Models



The following elements correspond to each numbered block.

1. **Pulldown**—A voltage-current table describes the current when the buffer is driven low based on a pull-down voltage range of $-V_{CC}$ to $2 V_{CC}$.
2. **Pullup**—A voltage-current table describes the current when the buffer is driven high based on a pull-up voltage range of $-V_{CC}$ to V_{CC} .
3. **Ground and Power Clamps**—Voltage-current tables describe the current when clamping diodes for electrostatic discharge (ESD) are present. The ground clamp voltage range is $-V_{CC}$ to V_{CC} , and the power clamp voltage range is $-V_{CC}$ to ground.
4. **Ramp and Rising/Falling Waveform**—A voltage-time (dv/dt) ratio describes the rise and fall time of the buffer during a logic transition. Optional rising and falling waveform tables can be added to more accurately describe the characteristics of the rising and falling transitions.
5. **Total Output Capacitance and Package RLC**—The total output capacitance includes the parasitic capacitances of the output pad, clamp diodes (if present), and input transistors. The package RLC is device package-specific and defines the resistance, inductance, and capacitance of the bond wire and pin of the I/O.

Related Links

[AN 283: Simulating Intel Devices with IBIS Models](#)

For more information about IBIS models and Intel-specific features, including links to the official IBIS specification.

6.4.2 Creating Accurate IBIS Models

There are two methods to obtain Intel device IBIS files for your board-level signal integrity simulations. You can download generic IBIS models from the Altera website. You can also use the IBIS writer in the Intel Quartus Prime software to create design-specific models.



The IBIS file generated by the Intel Quartus Prime software contains models of both input and output termination, and is supported for IBIS model versions of 4.2 and later. Arria V , Cyclone V , and Stratix V device families allow the use of bidirectional I/O with dynamic on-chip termination (OCT).

Dynamic OCT is used where a signal uses a series on-chip termination during output operation and a parallel on-chip termination during input operation. Typically this is used in Altera External Memory Interface IP.

The Intel Quartus Prime IBIS dynamic OCT IBIS model names end in g50c_r50c. For example : sst115i_ctnio_g50c_r50c.

In the simulation tool, the IBIS model is attached to a buffer.

- When the buffer is assigned as an output, use the series termination r50c.
- When the buffer is assigned as an input, use the parallel termination g50c.

6.4.2.1 Download IBIS Models

Intel provides IBIS models for almost all FPGA and FPGA configuration devices. You can use the IBIS models from the website to perform early simulations of the I/O buffers you expect to use in your design as part of a pre-layout analysis.

Downloaded IBIS models have the RLC package values set to one particular device in each device family.

The `.ibs` file can be customized for your device package and can be used for any simulation. IBIS models downloaded and used for simulations in this manner are generic. They describe only a certain set of models listed for each device on the Intel IBIS Models page of the Altera website. To create customized models for your design, use the IBIS Writer as described in the next section.

To simulate your design with the model accurately, you must adjust the RLC values in the IBIS model file to match the values for your particular device package by performing the following steps:

1. Download and expand the ZIP file (`.zip`) of the IBIS model for the device family you are using for your design. The `.zip` file contains the `.ibs` file along with an IBIS model user guide and a model data correlation report.
2. Download the Package RLC Values spreadsheet for the same device family.
3. Open the spreadsheet and locate the row that describes the device package used in your design.
4. From the package's **I/O** row, copy the minimum, maximum, and typical values of resistance, inductance, and capacitance for your device package.
5. Open the `.ibs` file in a text editor and locate the [Package] section of the file.
6. Overwrite the listed values copied with the values from the spreadsheet and save the file.

Related Links

[Intel IBIS Models](#)

For information about whether models for your selected device are available.

6.4.2.2 Generate Custom IBIS Models with the IBIS Writer

If you have started your FPGA design and have created custom I/O assignments, you can use the Intel Quartus Prime IBIS Writer to create custom IBIS models to accurately reflect your assignments.

Examples of custom assignments include drive strength settings or the enabling of clamping diodes for ESD protection. IBIS models created with the IBIS Writer take I/O assignment settings into account.

If the **Enable Advanced I/O Timing** option is turned off, the generated `.ibs` files are based on the load value setting for each I/O standard on the **Capacitive Loading** page of the **Device and Pin Options** dialog box in the **Device** dialog box. With the **Enable Advanced I/O Timing** option turned on, IBIS models use an effective capacitive load based on settings found in the board trace model on the **Board Trace Model** page in the **Device and Pin Options** dialog box or the **Board Trace Model** view in the Pin Planner. The effective capacitive load is based on the sum of the **Near capacitance**, **Transmission line distributed capacitance**, and the **Far capacitance** settings in the board trace model. Resistance values and transmission line inductance values are ignored.

Note:

If you made any changes from the default load settings, the delay in the generated IBIS model cannot safely be added to the Intel Quartus Prime t_{CO} measurement to account for the double counting problem. This is because the load values between the two delay measurements do not match. When this happens, the Intel Quartus Prime software displays warning messages when the EDA Netlist Writer runs to remind you about the load value mismatch.

Related Links

- [Intel IBIS models](#)
- [Generating IBIS Output Files with the Intel Quartus Prime Software](#)
In *Intel Quartus Prime Help*
- [AN 283: Simulating Intel Devices with IBIS Models](#)

6.4.3 Design Simulation Using the Mentor Graphics HyperLynx Software

You must integrate IBIS models downloaded from the Altera website or created with the Intel Quartus Prime IBIS Writer into board design simulations to accurately model timing and signal integrity.

The HyperLynx software from Mentor Graphics is one of the most popular tools for design simulation. The HyperLynx software makes it easy to integrate IBIS models into simulations.

The HyperLynx software is a PCB analysis and simulation tool for high-speed designs, consisting of two products, LineSim and BoardSim.

LineSim is an early simulation tool. Before any board routing takes place, you use LineSim to simulate “what if” scenarios that assist in creating routing rules and defining board parameters.



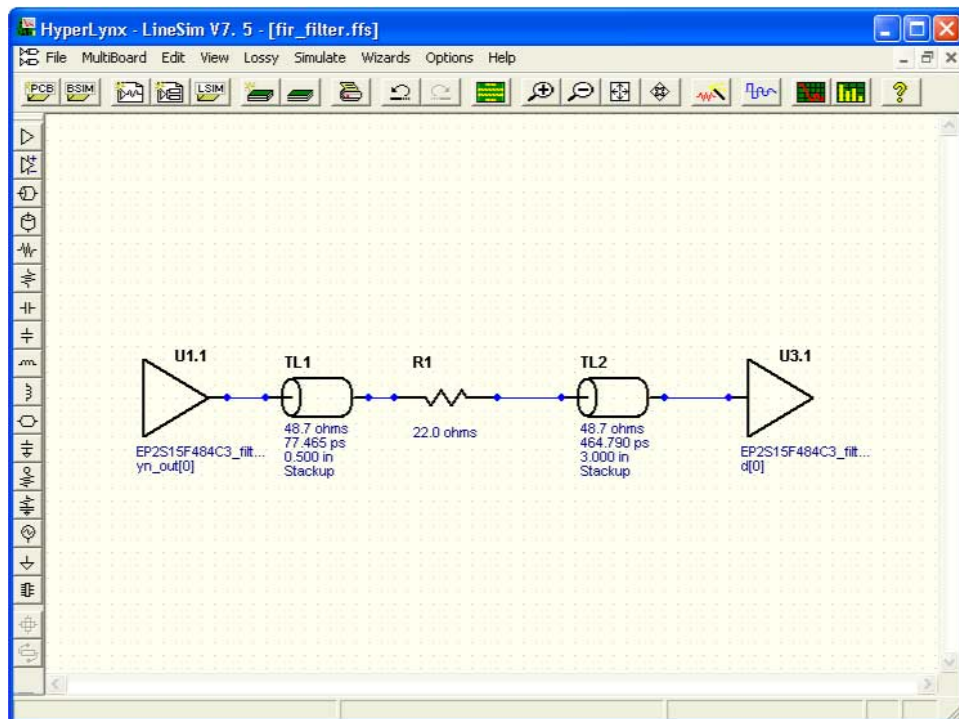
BoardSim is a post-layout tool that you use to analyze existing board routing. You select one or more nets from a board layout file and BoardSim simulates those nets in a manner similar to LineSim. With board and routing parameters, and surrounding signal routing known, highly accurate simulations of the final fabricated PCB are possible.

This section focuses on LineSim. Because the process of creating and running simulations is very similar for both LineSim and BoardSim, the details of IBIS model use in LineSim applies to simulations in BoardSim.

You configure simulations in LineSim using a schematic GUI to create connections and topologies between I/O buffers, route trace segments, and termination components. LineSim provides two methods for creating routing schematics: cell-based and free-form. Cell-based schematics are based on fixed cells consisting of typical placements of buffers, trace impedances, and components. Parts of the grid-based cells are filled with the desired objects to create the topology. A topology in a cell-based schematic is limited by the available connections within and between the cells.

A more robust and expandable way to create a circuit schematic for simulation is to use the free-form schematic format in LineSim. The free-form schematic format makes it easy to place parts into any configuration and edit them as required. This section describes the use of IBIS models with free-form schematics, but the process is nearly identical for cell-based schematics.

Figure 28. HyperLynx LineSim Free-Form Schematic Editor





When you use HyperLynx software to perform simulations, you typically perform the following steps:

1. Create a new LineSim free-form schematic document and set up the board stackup for your PCB using the Stackup Editor. In this editor, specify board layer properties including layer thickness, dielectric constant, and trace width.
2. Create a circuit schematic for the net you want to simulate. The schematic represents all the parts of the routed net including source and destination I/O buffers, termination components, transmission line segments, and representations of impedance discontinuities such as vias or connectors.
3. Assign IBIS models to the source and destination I/O buffers to represent their behavior during operation.
4. Attach probes from the digital oscilloscope that is built in to LineSim to points in the circuit that you want to monitor during simulation. Typically, at least one probe is attached to the pin of a destination I/O buffer. For differential signals, you can attach a differential probe to both the positive and negative pins at the destination.
5. Configure and run the simulation. You can simulate a rising or falling edge and test the circuit under different drive strength conditions.
6. Interpret the results and make adjustments. Based on the waveforms captured in the digital oscilloscope, you can adjust anything in the circuit schematic to correct any signal integrity issues, such as overshoot or ringing. If necessary, you can make I/O assignment changes in the Intel Quartus Prime software, regenerate the IBIS file with the IBIS Writer, and apply the updated IBIS model to the buffers in your HyperLynx software schematic.
7. Repeat the simulations and circuit adjustments until you are satisfied with the results.
8. When the operation of the net meets your design requirements, implement changes to your I/O assignments in the Intel Quartus Prime software and optionally adjust your board routing constraints, component values, and placement to match the simulation.

For more information about HyperLynx software, including schematic creation, simulation setup, model usage, product support, licensing, and training, refer to the Mentor Graphics webpage.

Related Links

www.mentor.com

6.4.4 Configuring LineSim to Use Intel IBIS Models

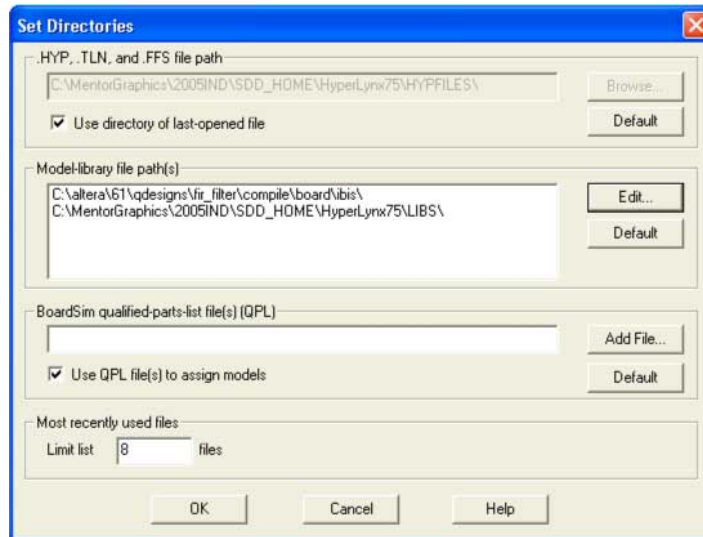
You must configure LineSim to find and use the downloaded or generated IBIS models for your design. To do this, add the location of your `.ibs` file or files to the LineSim Model Library search path. Then you apply a selected model to a buffer in your schematic.

To add the Intel Quartus Prime software's default IBIS model location, `<project directory>/board/ibis`, to the HyperLynx LineSim model library search path, perform the following steps in LineSim:

1. From the Options menu, click **Directories**. The **Set Directories** dialog box appears. The **Model-library file path(s)** list displays the order in which LineSim searches file directories for model files.

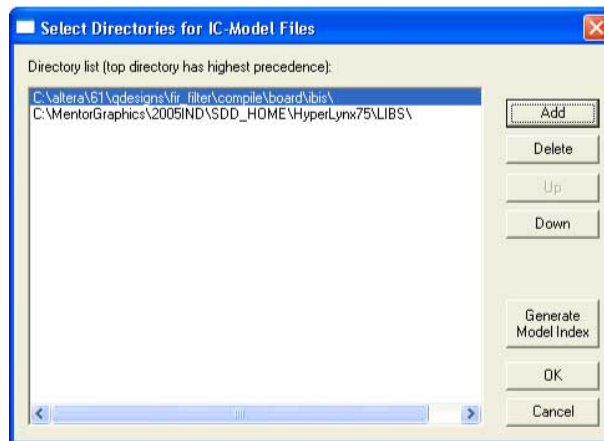


Figure 29. LineSim Set Directories Dialog Box



2. Click **Edit**. A dialog box appears where you can add directories and adjust the order in which LineSim searches them.

Figure 30. LineSim Select Directories Dialog Box



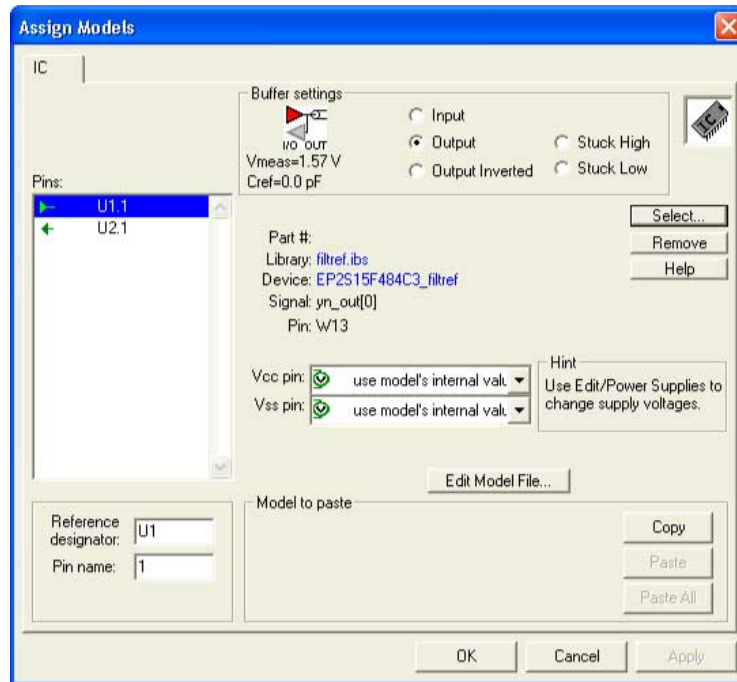
3. Click **Add**
4. Browse to the default IBIS model location, *<project directory>/board/ibis*. Click **OK**.
5. Click **Up** to move the IBIS model directory to the top of the list. Click **Generate Model Index** to update LineSim's model database with the models found in the added directory.
6. Click **OK**. The IBIS model directory for your project is added to the top of the Model-library file path(s) list.
7. To close the **Set Directories** dialog box, click **OK**.

6.4.5 Integrating Intel IBIS Models into LineSim Simulations

When the location for IBIS files has been set, you can assign the downloaded or generated IBIS models to the buffers in your schematic. To do this, perform the following steps:

1. Double-click a buffer symbol in your schematic to open the **Assign Models** dialog box. You can also click **Assign Models** from the buffer symbol's right-click menu.

Figure 31. LineSim Assign Model Dialog Box



2. The pin of the buffer symbol you selected should be highlighted in the **Pins** list. If you want to assign a model to a different symbol or pin, select it from the list.
3. Click **Select**. The **Select IC Model** dialog box appears.

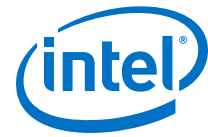
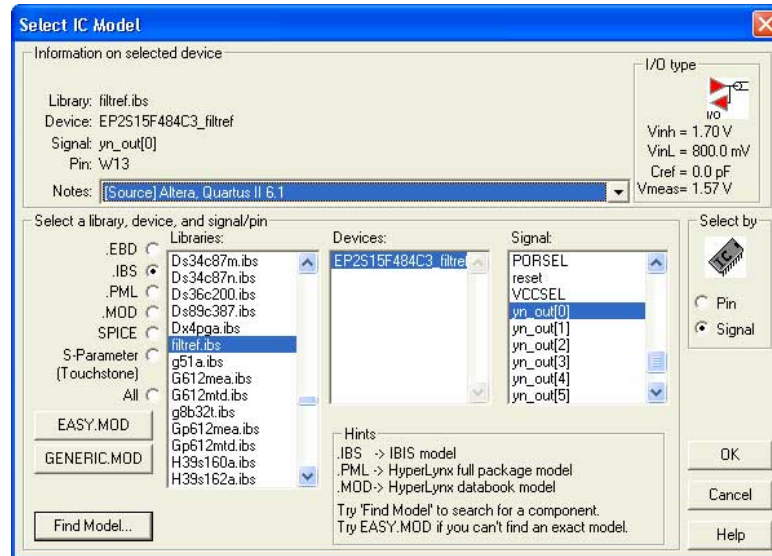


Figure 32. LineSim Select IC Model Dialog Box



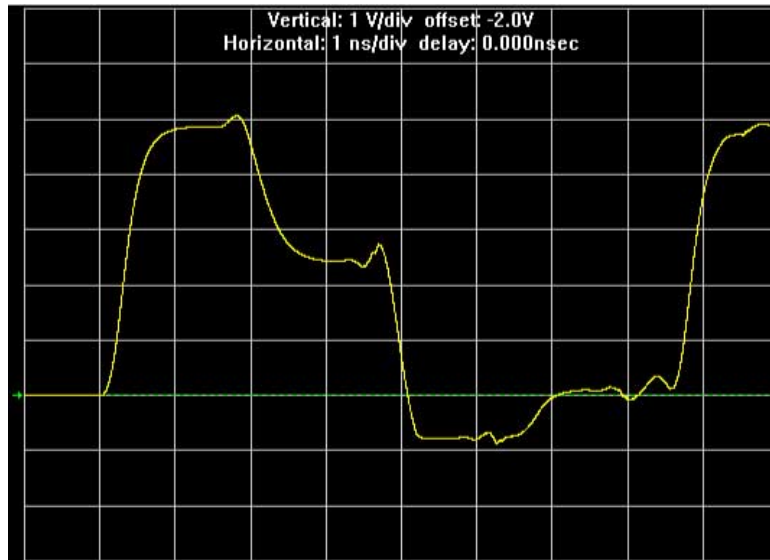
4. To filter the list of available libraries to display only IBIS models, select **.IBS**. Scroll through the **Libraries** list, and click the name of the library for your design. By default, this is *<project name>.ibs*.
5. The device for your design should be selected as the only item in the **Devices** list. If not, select your device from the list.
6. From the **Signal** list, select the name of the signal you want to simulate. You can also choose to select by device pin number.
7. Click **OK**. The **Assign Models** dialog box displays the selected **.ibs** file and signal.
8. If applicable to the signal you chose, adjust the buffer settings as required for the simulation.
9. Select and configure other buffer pins from the **Pins** list in the same manner.
10. Click **OK** when all I/O models are assigned.

6.4.6 Running and Interpreting LineSim Simulations

You can run any simulation and make adjustments to the I/O assignments or simulation parameters as required.

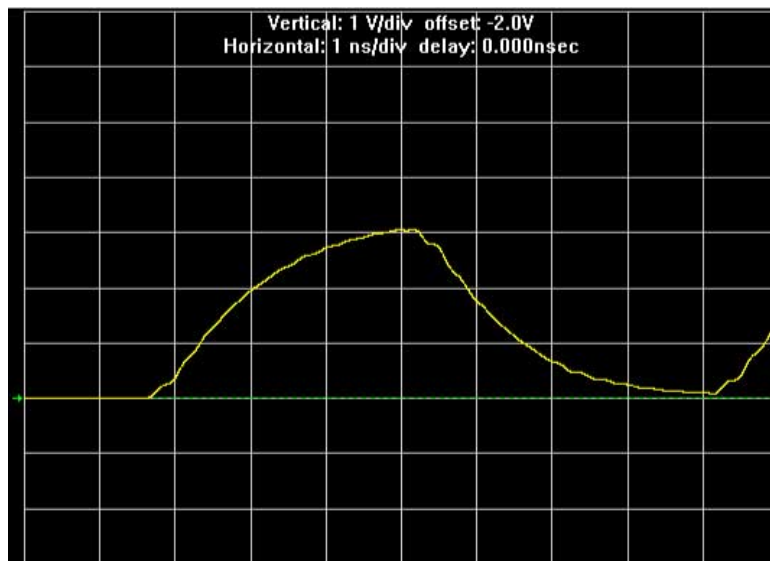
For example, if you see too much overshoot in the simulated signal at the destination buffer after running a simulation, you can adjust the drive strength I/O assignment setting to a lower value. Regenerate the **.ibs** file, and run the simulation again to verify whether the change fixes the problem.

Figure 33. Example of Overshoot in HyperLynx with IBIS Models



If you see a discontinuity or other anomalies at the destination, such as slow rise and fall times, adjust the termination scheme or termination component values. After making these changes, rerun the simulation to check whether your adjustments solved the problem. In this case, it is not necessary to regenerate the .ibs file.

Figure 34. Example of Signal Integrity Anomaly in HyperLynx with IBIS Models



For more information about board-level signal integrity, and to learn about ways to improve it with simple changes to your design, visit the Intel FPGA Signal & Power Integrity Support Center.

Related Links

[Intel Signal & Power Integrity Center](#)



6.5 Simulation with HSPICE Models

HSPICE decks are used to perform highly accurate simulations by describing the physical properties of all aspects of a circuit precisely. HSPICE decks describe I/O buffers, board components, and all the connections between them, as well as defining the parameters of the simulation to be run.

By their nature, HSPICE decks are highly customizable and require a detailed description of the circuit under simulation. For devices that support advanced I/O timing, when **Enable Advanced I/O Timing** is turned on, the HSPICE decks generated by the Intel Quartus Prime HSPICE Writer automatically include board components and topology defined in the Board Trace Model. Configure the board components and topology in the Pin Planner or in the **Board Trace Model** tab of the **Device and Pin Options** dialog box. All HSPICE decks generated by the Intel Quartus Prime software include compensation for the double count problem. You can simulate with the default simulation parameters built in to the generated HSPICE decks or make adjustments to customize your simulation.

Related Links

[The Double Counting Problem in HSPICE Simulations](#) on page 122

6.5.1 Supported Devices and Signaling

The HSPICE Writer in the Intel Quartus Prime software supports Arria , Cyclone, and Stratix devices for the creation of a board trace model in the Intel Quartus Prime software for automatic inclusion in an HSPICE deck.

The HSPICE files include the board trace description you create in the Board Trace Model view in the Pin Planner or the **Board Trace Model** tab in the **Device and Pin Options** dialog box.

Note: Note that for Intel Arria 10 devices, you may need to download the Encrypted HSPICE model from the Altera website.

Related Links

- [I/O Management](#) on page 24
For more information about the **Enable Advanced I/O Timing** option and configuring board trace models for the I/O standards in your design.
- [SPICE Models for Intel Devices](#)
For more information about the Encrypted HSPICE model.

6.5.2 Accessing HSPICE Simulation Kits

You can access the available HSPICE models with the Intel Quartus Prime software's HSPICE Writer tool and also at the Spice Models for Intel Devices web page.

The Intel Quartus Prime software HSPICE Writer tool removes many common sources of user error from the I/O simulation process. The HSPICE Writer tool automatically creates preconfigured I/O simulation spice decks that only require the addition of a user board model. All the difficult tasks required to configure the I/O modes and interpret the timing results are handled automatically by the HSPICE Writer tool.

Related Links

[Spice Models for Intel Devices](#)

For more information about downloadable HSPICE models.

6.5.3 The Double Counting Problem in HSPICE Simulations

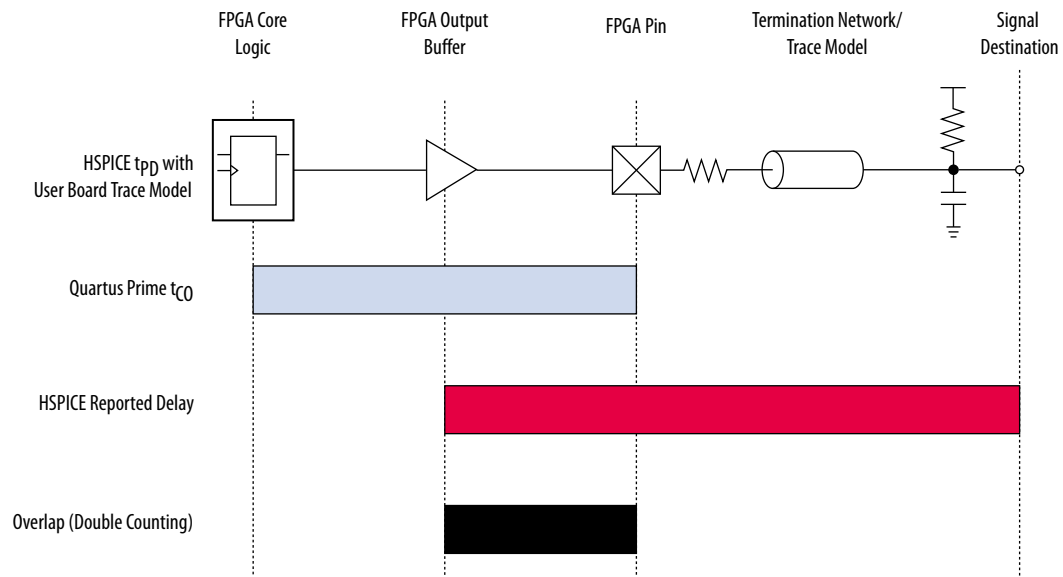
Simulating I/Os using accurate models is extremely helpful for finding and fixing FPGA I/O timing and board signal integrity issues before any boards are built. However, the usefulness of such simulations is directly related to the accuracy of the models used and whether the simulations are set up and performed correctly.

To ensure accuracy in models and simulations created for FPGA output signals you must consider the timing hand-off between t_{CO} timing in the Intel Quartus Prime software and simulation-based board delay. If this hand-off is not handled correctly, the calculated delay could either count some of the delay twice or even miss counting some of the delay entirely.

6.5.3.1 Defining the Double Counting Problem

The double counting problem is inherent to the difference between the method to analyze output timing in the Intel Quartus Prime software versus the method HSPICE models use. The timing analyzer tools in the Intel Quartus Prime software measure delay timing for an output signal from the core logic of the FPGA design through the output buffer, ending at the FPGA pin with a default capacitive load or a specified value for the I/O standard you selected. This measurement is the t_{CO} timing variable.

Figure 35. Double Counting Problem



HSPICE models for board simulation measure t_{PD} (propagation delay) from an arbitrary reference point in the output buffer, through the device pin, out along the board routing, and ending at the signal destination.

If you add these two delays, the delay between the output buffer and the device pin appears twice in the calculation. A model or simulation that does not account for this double count creates overly pessimistic simulation results, because the double-counted delay can limit I/O performance artificially.



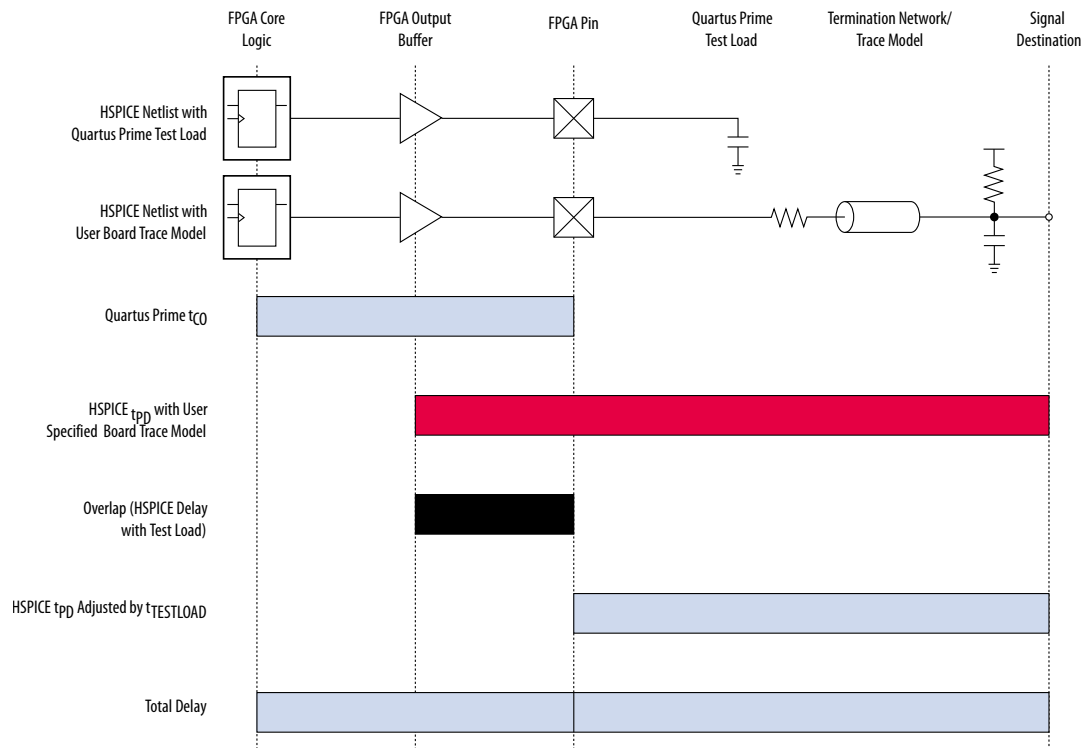
One approach to fix the problem is subtracting the overlap between t_{CO} and t_{PD} to account for the double count. However, this adjustment is not accurate, because each measurement considers a different load.

Note: Input signals do not exhibit this problem, because the HSPICE models for inputs stop at the FPGA pin instead of at the input buffer. In this case, adding the delays together produces an accurate measurement of delay timing.

6.5.3.2 The Solution to Double Counting

To adjust the measurements to account for the double-counting, the delay between the arbitrary point in the output buffer selected by the HSPICE model and the FPGA pin must be subtracted from either t_{CO} or t_{PD} before adding the results together. The subtracted delay must also be based on a common load between the two measurements. This is done by repeating the HSPICE model measurement, but with the same load used by the Intel Quartus Prime software for the t_{CO} measurement.

Figure 36. Common Test Loads Used for Output Timing



With $t_{TESTLOAD}$ known, the total delay is calculated for the output signal from the FPGA logic to the signal destination on the board, accounting for the double count.

$$t_{\text{delay}} = t_{CO} + (t_{PD} - t_{TESTLOAD})$$

The preconfigured simulation files generated by the HSPICE Writer in the Intel Quartus Prime software are designed to account for the double-counting problem based on this calculation automatically.

6.5.4 HSPICE Writer Tool Flow

This section includes information to help you get started using the Intel Quartus Prime software HSPICE Writer tool. The information in this section assumes you have a basic knowledge of the standard Intel Quartus Prime software design flow, such as project and assignment creation, compilation, and timing analysis.

6.5.4.1 Applying I/O Assignments

The first step in the HSPICE Writer tool flow is to configure the I/O standards and modes for each of the pins in your design properly. In the Intel Quartus Prime software, these settings are represented by assignments that map I/O settings, such as pin selection, and I/O standard and drive strength, to corresponding signals in your design.

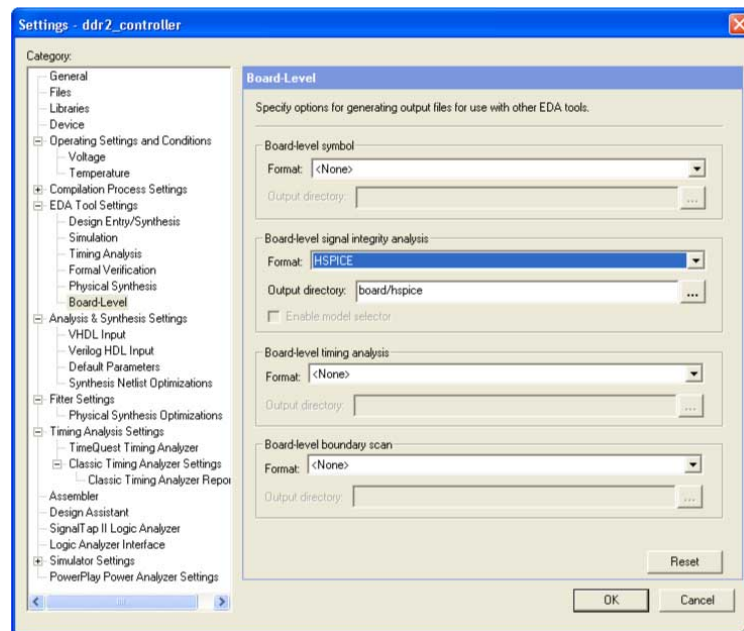
The Intel Quartus Prime software provides multiple methods for creating these assignments:

- Using the Pin Planner
- Using the assignment editor
- Manually editing the `.qsf` file
- By making assignments in a scripted Intel Quartus Prime flow using Tcl

6.5.4.2 Enabling HSPICE Writer

You must enable the HSPICE Writer in the **Settings** dialog box of the Intel Quartus Prime software to generate the HSPICE decks from the Intel Quartus Prime software.

Figure 37. EDA Tool Settings: Board Level Options Dialog Box





6.5.4.3 Enabling HSPICE Writer Using Assignments

You can also use HSPICE Writer in conjunction with a scripted Tcl flow. To enable HSPICE Writer during a full compile, include the following lines in your Tcl script.

Enable HSPICE Writer

```
set_global_assignment -name EDA_BOARD_DESIGN_SIGNAL_INTEGRITY_TOOL \  
"HSPICE (Signal Integrity)"  
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT HSPICE \  
-section_id eda_board_design_signal_integrity  
set_global_assignment -name EDA_NETLIST_WRITER_OUTPUT_DIR <output_directory> \  
-section_id eda_board_design_signal_integrity
```

As with command-line invocation, specifying the output directory is optional. If not specified, the output directory defaults to **board/hspice**.

6.5.4.4 Naming Conventions for HSPICE Files

HSPICE Writer automatically generates simulation files and names them using the following naming convention: <device>_<pin #>_<pin_name>_<in/out> .sp.

For bidirectional pins, two spice decks are produced; one with the I/O buffer configured as an input, and the other with the I/O buffer configured as an output.

The Intel Quartus Prime software supports alphanumeric pin names that contain the underscore (_) and dash (-) characters. Any illegal characters used in file names are converted automatically to underscores.

Related Links

- [Sample Output for I/O HSPICE Simulation Deck](#) on page 135
- [Sample Input for I/O HSPICE Simulation Deck](#) on page 131

6.5.4.5 Invoking HSPICE Writer

After HSPICE Writer is enabled, the HSPICE simulation files are generated automatically each time the project is completely compiled. The Intel Quartus Prime software also provides an option to generate a new set of simulation files without having to recompile manually. In the Processing menu, click **Start EDA Netlist Writer** to generate new simulation files automatically.

Note: You must perform both Analysis & Synthesis and Fitting on a design before invoking the HSPICE Writer tool.

6.5.4.6 Invoking HSPICE Writer from the Command Line

If you use a script-based flow to compile your project, you can create HSPICE model files by including the following commands in your Tcl script (.tcl file).

Create HSPICE Model Files

```
set_global_assignment -name EDA_BOARD_DESIGN_SIGNAL_INTEGRITY_TOOL \  
"HSPICE (Signal Integrity)"  
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT HSPICE \  
-section_id eda_board_design_signal_integrity  
set_global_assignment -name EDA_NETLIST_WRITER_OUTPUT_DIR <output_directory> \  
-section_id eda_board_design_signal_integrity
```



The `<output_directory>` option specifies the location where HSPICE model files are saved. By default, the `<project_directory>/board/hspice` directory is used.

Invoke HSPICE Writer

To invoke the HSPICE Writer tool through the command line, type:

```
quartus_eda.exe <project_name> --board_signal_integrity=on --format=HSPICE \  
--output_directory=<output_directory>
```

`<output_directory>` specifies the location where the tool writes the generated spice decks, relative to the design directory. This is an optional parameter and defaults to `board/hspice`.

6.5.4.7 Customizing Automatically Generated HSPICE Decks

HSPICE models generated by the HSPICE Writer can be used for simulation as generated.

A default board description is included, and a default simulation is set up to measure rise and fall delays for both input and output simulations, which compensates for the double counting problem. However, Intel recommends that you customize the board description to more accurately represent your routing and termination scheme.

The sample board trace loading in the generated HSPICE model files must be replaced by your actual trace model before you can run a correct simulation. To do this, open the generated HSPICE model files for all pins you want to simulate and locate the following section.

Sample Board Trace Section

```
* I/O Board Trace and Termination Description  
* - Replace this with your board trace and termination description
```

You must replace the example load with a load that matches the design of your PCB board. This includes a trace model, termination resistors, and, for output simulations, a receiver model. The spice circuit node that represents the pin of the FPGA package is called **pin**. The node that represents the far pin of the external device is called **load-in** (for output SPICE decks) and **source-in** (for input SPICE decks).

For an input simulation, you must also modify the stimulus portion of the spice file. The section of the file that must be modified is indicated in the following comment block.

Sample Source Stimulus Section

```
* Sample source stimulus placeholder  
* - Replace this with your I/O driver model
```

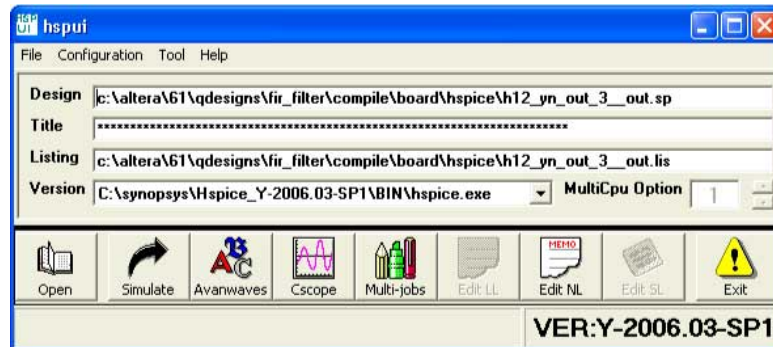
Replace the sample stimulus model with a model for the device that drives the FPGA.

6.5.5 Running an HSPICE Simulation

Because simulation parameters are configured directly in the HSPICE model files, running a simulation requires only that you open an HSPICE file in the HSPICE user interface and start the simulation.



Figure 38. HSPICE User Interface Window



Click **Open** and browse to the location of the HSPICE model files generated by the Intel Quartus Prime HSPICE Writer. The default location for HSPICE model files is *<project directory>/board/hspice*. Select the **.sp** file generated by the HSPICE Writer for the signal you want to simulate. Click **OK**.

To run the simulation, click **Simulate**. The status of the simulation is displayed in the window and saved in an **.lis** file with the same name as the **.sp** file when the simulation is complete. Check the **.lis** file if an error occurs during the simulation requiring a change in the **.sp** file to fix.

6.5.6 Interpreting the Results of an Output Simulation

By default, the automatically generated output simulation spice decks are set up to measure three delays for both rising and falling transitions. Two of the measurements, *tpd_rise* and *tpd_fall*, measure the double-counting corrected delay from the FPGA pin to the load pin. To determine the complete clock-edge to load-pin delay, add these numbers to the Intel Quartus Prime software reported default loading *t_{CO}* delay.

The remaining four measurements, *tpd_uncomp_rise*, *tpd_uncomp_fall*, *t_dblcnt_rise*, and *t_dblcnt_fall*, are required for the double-counting compensation process and are not required for further timing usage.

Related Links

[Simulation Analysis](#) on page 135

6.5.7 Interpreting the Results of an Input Simulation

By default, the automatically generated input simulation SPICE decks are set up to measure delays from the source's driver pin to the FPGA's input pin for both rising and falling transitions.

The propagation delay is reported by HSPICE measure statements as *tpd_rise* and *tpd_fall*. To determine the complete source driver pin-to-FPGA register delay, add these numbers to the Intel Quartus Prime software reported *T_H* and *T_{SU}* input timing numbers.

6.5.8 Viewing and Interpreting Tabular Simulation Results

The `.lis` file stores the collected simulation data in tabular form. The default simulation configured by the HSPICE Writer produces delay measurements for rising and falling transitions on both input and output simulations.

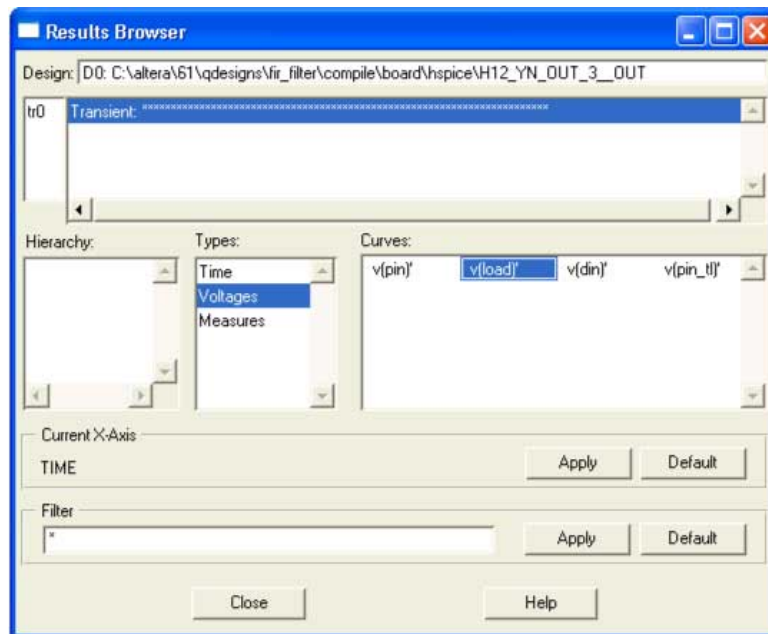
These measurements are found in the `.lis` file and named `tpd_rise` and `tpd_fall`. For output simulations, these values are already adjusted for the double count. To determine the complete delay from the FPGA logic to the load pin, add either of these measurements to the Intel Quartus Prime t_{CO} delay. For input simulations, add either of these measurements to the Intel Quartus Prime t_{SU} and t_H delay values to calculate the complete delay from the far end stimulus to the FPGA logic. Other values found in the `.lis` file, such as `tpd_uncomp_rise`, `tpd_uncomp_fall`, `t_dblcnt_rise`, and `t_dblcnt_fall`, are parts of the double count compensation calculation. These values are not necessary for further analysis.

6.5.9 Viewing Graphical Simulation Results

You can view the results of the simulation quickly as a graphical waveform display using the AvanWaves viewer included with HSPICE. With the default simulation configured by the HSPICE Writer, you can view the simulated waveforms at both the source and destination in input and output simulations.

To see the waveforms for the simulation, in the HSPICE user interface window, click **AvanWaves**. The AvanWaves viewer opens and displays the **Results Browser**.

Figure 39. HSPICE AvanWaves Results Browser



The **Results Browser** lets you select which waveform to view quickly in the main viewing window. If multiple simulations are run on the same signal, the list at the top of the **Results Browser** displays the results of each simulation. Click the simulation



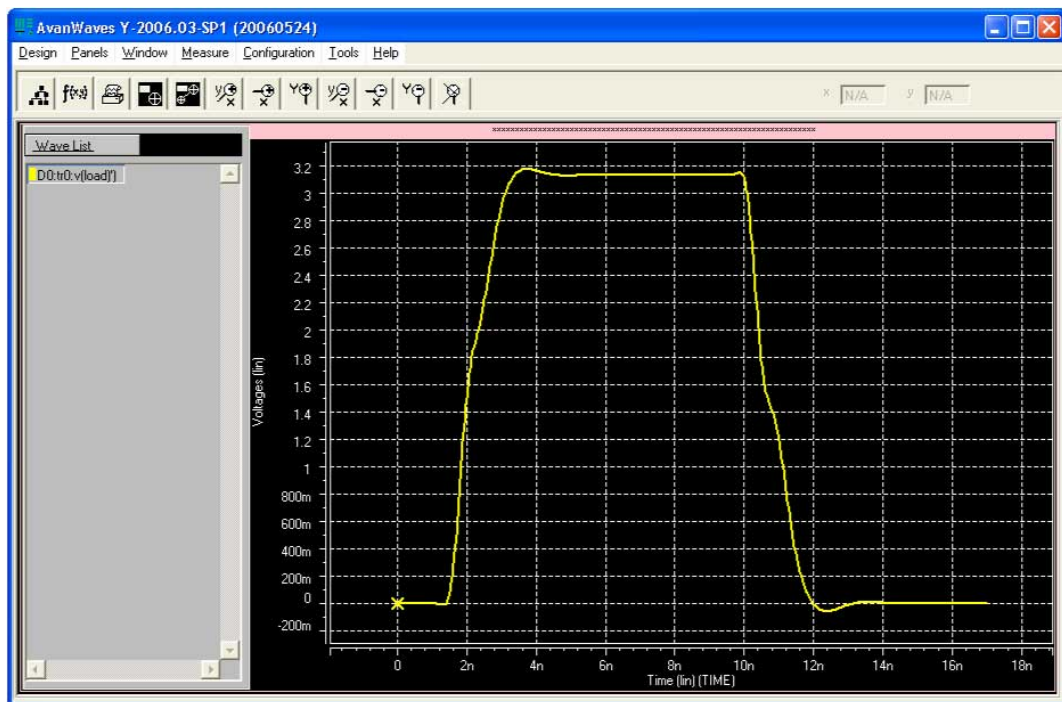
description to select which simulation to view. By default, the descriptions are derived from the first line of the HSPICE file, so the description might appear as a line of asterisks.

Select the type of waveform to view, by performing the following steps:

1. To see the source and destination waveforms with the default simulation, from the **Types** list, select **Voltages**.
2. On the **Curves** list, double-click the waveform you want to view. The waveform appears in the main viewing window.

You can zoom in and out and adjust the view as desired.

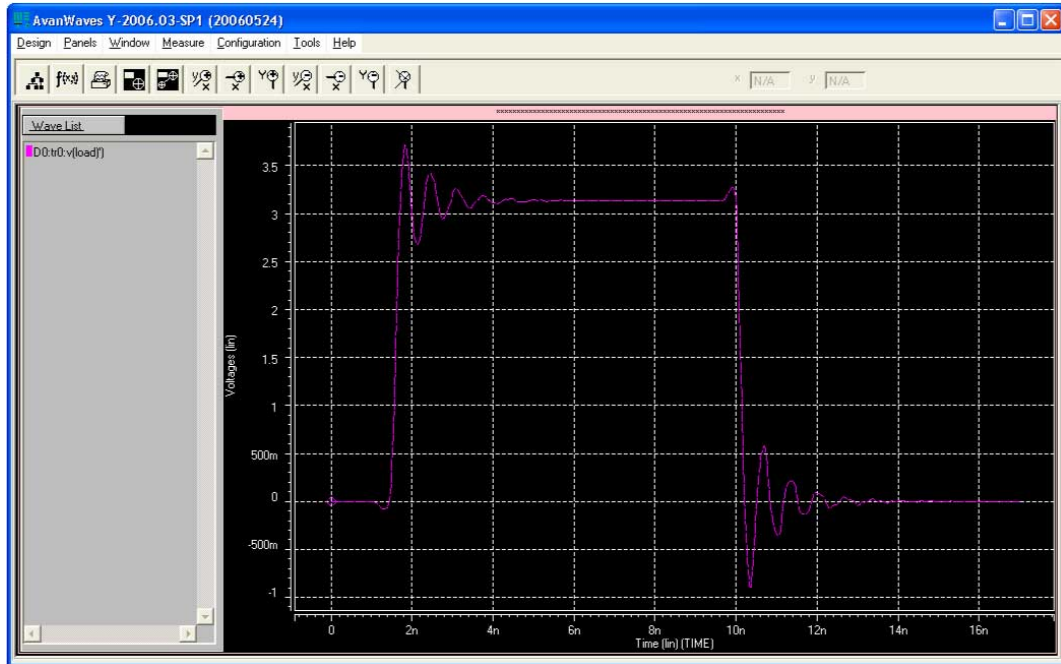
Figure 40. AvanWaves Waveform Viewer



6.5.10 Making Design Adjustments Based on HSPICE Simulations

Based on the results of your simulations, you can make adjustments to the I/O assignments or simulation parameters if required. For example, after you run a simulation and see overshoot or ringing in the simulated signal at the destination buffer, you can adjust the drive strength I/O assignment setting to a lower value. Regenerate the HSPICE deck, and run the simulation again to verify that the change fixed the problem.

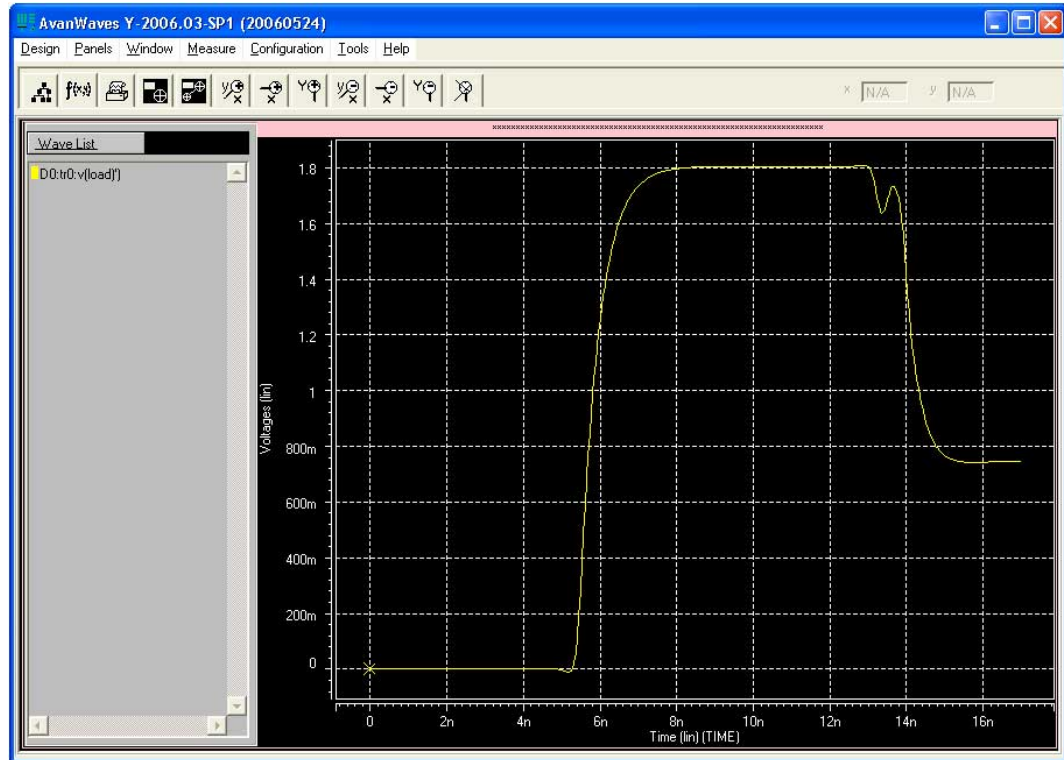
Figure 41. Example of Overshoot in the AvanWaves Waveform Viewer



If there is a discontinuity or any other anomalies at the destination, adjust the board description in the Intel Quartus Prime Board Trace Model, or in the generated HSPICE model files to change the termination scheme or adjust termination component values. After making these changes, regenerate the HSPICE files if necessary, and rerun the simulation to verify whether your adjustments solved the problem.



Figure 42. Example of Signal Integrity Anomaly in the AvanWaves Waveform Viewer



For more information about board-level signal integrity and to learn about ways to improve it with simple changes to your FPGA design, visit the Intel Signal & Power Integrity Center

Related Links

[Intel Signal & Power Integrity Center](#)

6.5.11 Sample Input for I/O HSPICE Simulation Deck

The following sections examine a typical HSPICE simulation spice deck for an I/O of type input. Each section presents the simulation file one block at a time.

6.5.11.1 Header Comment

The first block of an input simulation spice deck is the header comment. The purpose of this block is to provide an easily readable summary of how the simulation file has been automatically configured by the Intel Quartus Prime software.

This block has two main components: The first component summarizes the I/O configuration relevant information such as device, speed grade, and so on. The second component specifies the exact test condition that the Intel Quartus Prime software assumes for the given I/O standard.

Sample Header Comment Block

```

* Intel Quartus Prime HSPICE Writer I/O Simulation Deck*

* This spice simulation deck was automatically generated by
* Quartus for the following IO settings:
*
* Device:          EP2S60F1020C3
* Speed Grade:    C3
* Pin:            AA4 (out96)
* Bank:          IO Bank 6 (Row I/O)
* I/O Standard:  LVTTTL, 12mA
* OCT:           Off
*
* Intel Quartus Prime's default I/O timing delays assume the following slow
* corner simulation conditions.
*
* Specified Test Conditions For Intel Quartus Prime Tco
* Temperature:    85C (Slowest Temperature Corner)
* Transistor Model: TT (Typical Transistor Corner)
* Vccn:          3.135V (Vccn_min = Nominal - 5%)
* Vccpd:        2.97V (Vccpd_min = Nominal - 10%)
* Load:         No Load
* Vtt:          1.5675V (Voltage reference is Vccn/2)
*
* Note: The I/O transistors are specified to operate at least as
* fast as the TT transistor corner, actual production
* devices can be as fast as the FF corner. Any simulations
* for hold times should be conducted using the fast process
* corner with the following simulation conditions.
*   Temperature:    0C (Fastest Commercial Temperature Corner **)
*   Transistor Model: FF (Fastest Transistor Corner)
*   Vccn:          1.98V (Vccn_hold = Nominal + 10%)
*   Vccpd:        3.63V (Vccpd_hold = Nominal + 10%)
*   Vtt:          0.95V (Vtt_hold = Vccn/2 - 40mV)
*   Vcc:          1.25V (Vcc_hold = Maximum Recommended)
*   Package Model: Short-circuit from pad to pin (no parasitics)
*
* Warnings:
  
```

6.5.11.2 Simulation Conditions

The simulation conditions block loads the appropriate process corner models for the transistors. This condition is automatically set up for the slow timing corner and is modified only if other simulation corners are desired.

Simulation Conditions Block

```

* Process Settings

.options brief
.inc 'sii_tt.inc' * TT process corner
  
```

6.5.11.3 Simulation Options

The simulation options block configures the simulation temperature and configures HSPICE with typical simulation options.

Simulation Options Block

```

* Simulation Options

.options brief=0
.options badchr co=132 scale=1e-6 acct ingold=2 nomod dv=1.0
  
```



```
+      dcstep=1 absv=1e-3 absi=1e-8 probe csdf=2 accurate=1
+      converge=1
.temp 85
```

Note: For a detailed description of these options, consult your *HSPICE* manual.

6.5.11.4 Constant Definition

The constant definition block of the simulation file instantiates the voltage sources that controls the configuration modes of the I/O buffer.

Constant Definition Block

```
* Constant Definition
voeb      oeb      0      vc * Set to 0 to enable buffer output
vopdrain  opdrain  0      0  * Set to vc to enable open drain
vrambh    rambh    0      0  * Set to vc to enable bus hold
vrpullup  rpullup  0      0  * Set to vc to enable weak pullup
vpcdp5    rpcdp5  0      rp5 * Set the IO standard
vpcdp4    rpcdp4  0      rp4
vpcdp3    rpcdp3  0      rp3
vpcdp2    rpcdp2  0      rp2
vpcdp1    rpcdp1  0      rp1
vpcdp0    rpcdp0  0      rp0
vpcdn4    rpcdn4  0      rn4
vpcdn3    rpcdn3  0      rn3
vpcdn2    rpcdn2  0      rn2
vpcdn1    rpcdn1  0      rn1
vpcdn0    rpcdn0  0      rn0
vdin din    0      0
```

Where:

- Voltage source `voeb` controls the output enable of the buffer and is set to disabled for inputs.
- `vopdrain` controls the open drain mode for the I/O.
- `vrambh` controls the bus hold circuitry in the I/O.
- `vrpullup` controls the weak pullup.
- The next 11 voltages sources control the I/O standard of the buffer and are configured through a later library call.
- `vdin` is not used on input pins because it is the data pin for the output buffer.

6.5.11.5 Buffer Netlist

The buffer netlist block of the simulation spice deck loads all the load models required for the corresponding input pin.

Buffer Netlist Block

```
* IO Buffer Netlist
.include `vio_buffer.inc`
```

6.5.11.6 Drive Strength

The drive strength block of the simulation SPICE deck loads the configuration bits necessary to configure the I/O into the proper I/O standard and drive strengths.



Although these settings are not relevant to an input buffer, they are provided to allow the SPICE deck to be modifiable to support bidirectional simulations.

Drive Strength Block

```
* Drive Strength Settings
.lib 'drive_select_hio.lib' 3p3ttl_12ma
```

6.5.11.7 I/O Buffer Instantiation

The I/O buffer instantiation block of the simulation SPICE deck instantiates the necessary power supplies and I/O model components that are necessary to simulate the given I/O.

I/O Buffer Instantiation

```
I/O Buffer Instantiation

* Supply Voltages Settings
.param vcn=3.135
.param vpd=2.97
.param vc=1.15

* Instantiate Power Supplies|
vvcc      vcc      0      vc      * FPGA core voltage
vvss      vss      0      0      * FPGA core ground
vvccn     vccn     0      vcn     * IO supply voltage
vvssn     vssn     0      0      * IO ground
vvccpd    vccpd    0      vpd     * Pre-drive supply voltage

* Instantiate I/O Buffer
xvio_buf  din oeb opdrain die rambh
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp5 rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup vccn vccpd vcpad0 vio_buf

* Internal Loading on Pad
* - No loading on this pad due to differential buffer/support
*   circuitry

* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg
```

6.5.11.8 Board Trace and Termination

The board trace and termination block of the simulation SPICE deck is provided only as an example. Replace this block with your own board trace and termination models.

Board Trace and Termination Block

```
* I/O Board Trace and Termination Description
* - Replace this with your board trace and termination description

wtline pin vssn load vssn N=1 L=1 RLGCMODEL=tlinemodel
.MODEL tlinemodel W MODELTYPE=RLGC N=1 Lo=7.13n Co=2.85p
Rterm2 load vssn 1x
```



6.5.11.9 Stimulus Model

The stimulus model block of the simulation spice deck is provided only as a place holder example. Replace this block with your own stimulus model. Options for this include an IBIS or HSPICE model, among others.

Stimulus Model Block

```
* Sample source stimulus placeholder
* - Replace this with your I/O driver model

Vsource source 0 pulse(0 vcn 0s 0.4ns 0.4ns 8.5ns 17.4ns)
```

6.5.11.10 Simulation Analysis

The simulation analysis block of the simulation file is configured to measure the propagation delay from the source to the FPGA pin. Both the source and end point of the delay are referenced against the 50% V_{CCN} crossing point of the waveform.

Simulation Analysis Block

```
* Simulation Analysis Setup

* Print out the voltage waveform at both the source and the pin
.print tran v(source) v(pin)
.tran 0.020ns 17ns

* Measure the propagation delay from the source pin to the pin
* referenced against the 50% voltage threshold crossing point

.measure TRAN tpd_rise TRIG v(source) val='vcn*0.5' rise=1
+ TARG v(pin) val = 'vcn*0.5' rise=1
.measure TRAN tpd_fall TRIG v(source) val='vcn*0.5' fall=1
+ TARG v(pin) val = 'vcn*0.5' fall=1
```

6.5.12 Sample Output for I/O HSPICE Simulation Deck

A typical HSPICE simulation SPICE deck for an I/O-type output has several sections. Each section presents the simulation file one block at a time.

6.5.12.1 Header Comment

The first block of an output simulation SPICE deck is the header comment. The purpose of this block is to provide a readable summary of how the simulation file has been automatically configured by the Intel Quartus Prime software.

This block has two main components:

- The first component summarizes the I/O configuration relevant information such as device, speed grade, and so on.
- The second component specifies the exact test condition that the Intel Quartus Prime software assumes when generating t_{CO} delay numbers. This information is used as part of the double-counting correction circuitry contained in the simulation file.

The SPICE decks are preconfigured to calculate the slow process corner delay but can also be used to simulate the fast process corner as well. The fast corner conditions are listed in the header under the notes section.



The final section of the header comment lists any warning messages that you must consider when you use the SPICE decks.

Header Comment Block

```
* Intel Quartus Prime HSPICE Writer I/O Simulation Deck
*
* This spice simulation deck was automatically generated by
* Intel Quartus Prime for the following IO settings:
*
* Device:          EP2S60F1020C3
* Speed Grade:    C3
* Pin:            AA4 (out96)
* Bank:          IO Bank 6 (Row I/O)
* I/O Standard:  LVTTL, 12mA
* OCT:           Off
*
* Quartus' default I/O timing delays assume the following slow
* corner simulation conditions.
* Specified Test Conditions For Intel Quartus Prime Tco
* Temperature:    85C (Slowest Temperature Corner)
* Transistor Model: TT (Typical Transistor Corner)
* Vccn:          3.135V (Vccn_min = Nominal - 5%)
* Vccpd:         2.97V (Vccpd_min = Nominal - 10%)
* Load:         No Load
* Vtt:           1.5675V (Voltage reference is Vccn/2)
* For C3 devices, the TT transistor corner provides an
* approximation for worst case timing. However, for functionality
* simulations, it is recommended that the SS corner be simulated
* as well.
*
* Note: The I/O transistors are specified to operate at least as
* fast as the TT transistor corner, actual production
* devices can be as fast as the FF corner. Any simulations
* for hold times should be conducted using the fast process
* corner with the following simulation conditions.
* Temperature:    0C (Fastest Commercial Temperature Corner **)
* Transistor Model: FF (Fastest Transistor Corner)
* Vccn:          1.98V (Vccn_hold = Nominal + 10%)
* Vccpd:         3.63V (Vccpd_hold = Nominal + 10%)
* Vtt:           0.95V (Vtt_hold = Vccn/2 - 40mV)
* Vcc:           1.25V (Vcc_hold = Maximum Recommended)
* Package Model: Short-circuit from pad to pin
* Warnings:
```

6.5.12.2 Simulation Conditions

The simulation conditions block loads the appropriate process corner models for the transistors. This condition is automatically set up for the slow timing corner and must be modified only if other simulation corners are desired.

Simulation Conditions Block

```
* Process Settings

.options brief
.inc 'sii_tt.inc' * typical-typical process corner
```

Note: Two separate corners cannot be simulated at the same time. Instead, simulate the base case using the Quartus corner as one simulation and then perform a second simulation using the desired customer corner. The results of the two simulations can be manually added together.



6.5.12.3 Simulation Options

The simulation options block configures the simulation temperature and configures HSPICE with typical simulation options.

Simulation Options Block

```
* Simulation Options
.options brief=0
.options badchr co=132 scale=1e-6 acct ingold=2 nomod dv=1.0
+      dcstep=1 absv=1e-3 absi=1e-8 probe csdf=2 accurate=1
+      converge=1
.temp 85
```

Note: For a detailed description of these options, consult your *HSPICE* manual.

6.5.12.4 Constant Definition

The constant definition block of the output simulation SPICE deck instantiates the voltage sources that controls the configuration modes of the I/O buffer.

Constant Definition Block

```
* Constant Definition

voeb      oeb      0      0 * Set to 0 to enable buffer output
vopdrain  opdrain  0      0 * Set to vc to enable open drain
vrambh    rambh    0      0 * Set to vc to enable bus hold
vrpullup  rpullup  0      0 * Set to vc to enable weak pullup
vpci      rpci      0      0 * Set to vc to enable pci mode
vpcdp4    rpcdp4    0      rp4 * These control bits set the IO standard
vpcdp3    rpcdp3    0      rp3
vpcdp2    rpcdp2    0      rp2
vpcdp1    rpcdp1    0      rp1
vpcdp0    rpcdp0    0      rp0
vpcdn4    rpcdn4    0      rn4
vpcdn3    rpcdn3    0      rn3
vpcdn2    rpcdn2    0      rn2
vpcdn1    rpcdn1    0      rn1
vpcdn0    rpcdn0    0      rn0
vdin      din      0      pulse(0 vc 0s 0.2ns 0.2ns 8.5ns 17.4ns)
```

Where:

- Voltage source `voeb` controls the output enable of the buffer.
- `vopdrain` controls the open drain mode for the I/O.
- `vrambh` controls the bus hold circuitry in the I/O.
- `vrpullup` controls the weak pullup.
- `vpci` controls the PCI clamp.
- The next ten voltage sources control the I/O standard of the buffer and are configured through a later library call.
- `vdin` is connected to the data input of the I/O buffer.
- The edge rate of the input stimulus is automatically set to the correct value by the Intel Quartus Prime software.

6.5.12.5 I/O Buffer Netlist

The I/O buffer netlist block loads all of the models required for the corresponding pin. These include a model for the I/O output buffer, as well as any loads that might be present on the pin.

I/O Buffer Netlist Block

```
*IO Buffer Netlist

.include 'hio_buffer.inc'
.include 'lvds_input_load.inc'
.include 'lvds_oct_load.inc'
```

6.5.12.6 Drive Strength

The drive strength block of the simulation spice deck loads the configuration bits for configuring the I/O to the proper I/O standard and drive strength. These options are set by the HSPICE Writer tool and are not changed for expected use.

Drive Strength Block

```
* Drive Strength Settings

.lib 'drive_select_hio.lib' 3p3ttl_12ma
```

6.5.12.7 Slew Rate and Delay Chain

Stratix and Cyclone devices have sections for configuring the slew rate and delay chain settings.

Slew Rate and Delay Chain Settings

```
* Programmable Output Delay Control Settings

.lib 'lib/output_delay_control.lib' no_delay

* Programmable Slew Rate Control Settings

.lib 'lib/slew_rate_control.lib' slow_slow
```

6.5.12.8 I/O Buffer Instantiation

The I/O buffer instantiation block of the output simulation spice deck instantiates the necessary power supplies and I/O model components that are necessary to simulate the given I/O.

I/O Buffer Instantiation Block

```
* I/O Buffer Instantiation

* Supply Voltages Settings
.param vcn=3.135
.param vpd=2.97
.param vc=1.15

* Instantiate Power Supplies
vvcc      vcc      0      vc      * FPGA core voltage
vvss      vss      0      0      * FPGA core ground
vvccn     vccn     0      vcn     * IO supply voltage
vvssn     vssn     0      0      * IO ground
vvccpd    vccpd    0      vpd     * Pre-drive supply voltage
```



```

* Instantiate I/O Buffer
xhio_buf din oeb opdrain die rambh
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup vccn vccpd vcpad0 hio_buf

* Internal Loading on Pad
* - This pad has an LVDS input buffer connected to it, along
*   with differential OCT circuitry. Both are disabled but
*   introduce loading on the pad that is modeled below.
xlvds_input_load die vss vccn lvds_input_load
xlvds_oct_load die vss vccpd vccn vcpad0 vccn lvds_oct_load

* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg

```

6.5.12.9 Board and Trace Termination

The board trace and termination block of the simulation SPICE deck is provided only as an example. Replace this block with your specific board loading models.

Board Trace and Termination Block

```

* I/O Board Trace And Termination Description
* - Replace this with your board trace and termination description
wtline pin vssn load vssn N=1 L=1 RLGCMODEL=tlinemodel
.MODEL tlinemodel W MODELTYPE=RLGC N=1 Lo=7.13n Co=2.85p
Rterm2 load vssn 1x

```

6.5.12.10 Double-Counting Compensation Circuitry

The double-counting compensation circuitry block of the simulation SPICE deck instantiates a second I/O buffer that is used to measure double-counting. The buffer is configured identically to the user I/O buffer but is connected to the Intel Quartus Prime software test load. The simulated delay of this second buffer can be interpreted as the amount of double-counting between the Intel Quartus Prime software and HSPICE Writer simulated results.

As the amount of double-counting is constant for a given I/O standard on a given pin, consider separating the double-counting circuitry from the simulation file. In doing so, you can perform any number of I/O simulations while referencing the delay only once.

(Part of)Double-Counting Compensation Circuitry Block

```

* Double Counting Compensation Circuitry
*
* The following circuit is designed to calculate the amount of
* double counting between Intel Quartus Prime and the HSPICE models. If
* you have not changed the default simulation temperature or
* transistor corner this spice deck automatically compensates the double
* counting.
* In the event you wish to
* simulate an IO at a different temperature or transistor corner
* you need to remove this section of code and manually
* account for double counting. A description of Intel's
* recommended procedure for this process can be found in the
* Intel Quartus Prime HSPICE Writer AppNote.

* Supply Voltages Settings
.param vcn_tl=3.135
.param vpd_tl=2.97

```



```
* Test Load Constant Definition
vopdrain_tl  opdrain_tl  0    0
vrambh_tl   rambh_tl    0    0
vrpullup_tl rpullup_tl  0    0

* Instantiate Power Supplies
vvccn_tl    vccn_tl     0    vcn_tl
vvssn_tl    vssn_tl    0    0
vvccpd_tl   vccpd_tl   0    vpd_tl

* Instantiate I/O Buffer
xhio_testload din oeb opdrain_tl die_tl rambh_tl
+ rpcdn4 rpcdn3 rpcdn2 rpcdn1 rpcdn0
+ rpcdp4 rpcdp3 rpcdp2 rpcdp1 rpcdp0
+ rpullup_tl vccn_tl vccpd_tl vcpad0_tl hio_buf

* Internal Loading on Pad
xlvs_input_testload die_tl vss vccn_tl lvds_input_load
xlvs_oct_testload die_tl vss vccpd_tl vccn_tl vcpad0_tl vccn_tl
lvds_oct_load
* I/O Buffer Package Model
* - Single-ended I/O standard on a Row I/O
.lib 'lib/package.lib' hio
xpkg die pin hio_pkg
* Default Intel Test Load
* - 3.3V LVTTTL default test condition is an open load
```

Related Links

[The Double Counting Problem in HSPICE Simulations](#) on page 122

6.5.12.11 Simulation Analysis

The simulation analysis block is set up to measure double-counting corrected delays. This is accomplished by measuring the uncompensated delay of the I/O buffer when connected to the user load, and when subtracting the simulated amount of double-counting from the test load I/O buffer.

Simulation Analysis Block

```
* Simulation Analysis Setup

*Print out the voltage waveform at both the pin and far end load
.print tran v(pin) v(load)
.tran 0.020ns 17ns

* Measure the propagation delay to the load pin. This value
* includes some double counting with Intel Quartus Prime's Tco
.measure TRAN tpd_uncomp_rise TRIG v(din) val='vc*0.5' rise=1+ TARG v(load)
val='vcn*0.5' rise=1
.measure TRAN tpd_uncomp_fall TRIG v(din) val='vc*0.5' fall=1
+ TARG v(load) val='vcn*0.5' fall=1

* The test load buffer can calculate the amount of double counting
.measure TRAN t_dblcnt_rise TRIG v(din) val='vc*0.5' rise=1
+ TARG v(pin_tl) val='vcn_tl*0.5' rise=1
.measure TRAN t_dblcnt_fall TRIG v(din) val='vc*0.5' fall=1
+ TARG v(pin_tl) val='vcn_tl*0.5' fall=1

* Calculate the true propagation delay by subtraction
.measure TRAN tpd_rise PARAM='tpd_uncomp_rise-t_dblcnt_rise'
.measure TRAN tpd_fall PARAM='tpd_uncomp_fall-t_dblcnt_fall'
```



6.5.13 Advanced Topics

The information in this section describes some of the more advanced topics and methods employed when setting up and running HSPICE simulation files.

6.5.13.1 PVT Simulations

The automatically generated HSPICE simulation files are set up to simulate the slow process corner using low voltage, high temperature, and slow transistors. To ensure a fully robust link, Intel recommends that you run simulations over all process corners.

To perform process, voltage, and temperature (PVT) simulations, manually modify the spice decks in a two step process:

1. Remove the double-counting compensation circuitry from the simulation file. This is required as the amount of double-counting is dependant upon how the Intel Quartus Prime software calculates delays and is not based on which PVT corner is being simulated. By default, the Intel Quartus Prime software provides timing numbers using the slow process corner.
2. Select the proper corner for the PVT simulation by setting the correct HSPICE temperature, changing the supply voltage sources, and loading the correct transistor models.

A more detailed description of HSPICE process corners can be found in the family-specific HSPICE model documentation.

Related Links

[Accessing HSPICE Simulation Kits](#) on page 121

6.5.13.2 Hold Time Analysis

Intel recommends performing worst-case hold time analysis using the fast corner models, which use fast transistors, high voltage, and low temperature. This involves modifying the SPICE decks to select the correct temperature option, change the supply voltage sources, and load the correct fast transistor models. The values of these parameters are located in the header comment section of the corresponding simulation deck files.

For a truly worst-case analysis, combine the HSPICE Writer hold time analysis results with the Intel Quartus Prime software fast timing model. This requires that you change the double-counting compensation circuitry in the simulations files to also simulate the fast process corners, as this is what the Intel Quartus Prime software uses for the fast timing model.

Note: This method of hold time analysis is recommended only for globally synchronous buses. Do not apply this method of hold-time analysis to source synchronous buses. This is because the source synchronous clocking scheme is designed to cancel out some of the PVT timing effects. If this is not taken into account, the timing results are not accurate. Proper source synchronous timing analysis is beyond the scope of this document.

6.5.13.3 I/O Voltage Variations

Use each of the FPGA family datasheets to verify the recommended operating conditions for supply voltages. For current FPGA families, the maximum recommended voltage corresponds to the fast corner, while the minimum recommended voltage



corresponds to the slow corner. These voltage recommendations are specified at the power pins of the FPGA and are not necessarily the same voltage that are seen by the I/O buffers due to package IR drops.

The automatically generated HSPICE simulation files model this IR effect pessimistically by including a 50-mV IR drop on the V_{CCPD} supply when a high drive strength standard is being used.

6.5.13.4 Correlation Report

Correlation reports for the HSPICE I/O models are located in the family-specific HSPICE I/O buffer simulation kits.

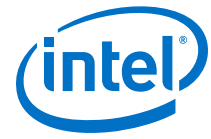
Related Links

[Accessing HSPICE Simulation Kits](#) on page 121

6.6 Document Revision History

Table 23. Document Revision History

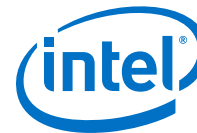
Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> Reorganized chapter introduction.
2016.10.31	16.1.0	<ul style="list-style-type: none"> Corrected statement about timing simulation and double counting.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
June 2014	14.0.0	Updated format.
December 2010	10.0.1	Template update.
July 2010	10.0.0	Updated device support.
November 2009	9.1.0	No change to content.
March 2009	9.0.0	<ul style="list-style-type: none"> Was volume 3, chapter 12 in the 8.1.0 release. No change to content.
November 2008	8.1.0	<ul style="list-style-type: none"> Changed to 8-1/2 x 11 page size. Added information for Stratix III devices. Input signals for Cyclone III devices are supported.
May 2008	8.0.0	<ul style="list-style-type: none"> Updated "Introduction" on page 12-1. Updated Figure 12-1. Updated Figure 12-3. Updated Figure 12-13. Updated "Output File Generation" on page 12-6. Updated "Simulation with HSPICE Models" on page 12-17. Updated "Invoking HSPICE Writer from the Command Line" on page 12-22. Added "Sample Input for I/O HSPICE Simulation Deck" on page 12-29. Added "Sample Output for I/O HSPICE Simulation Deck" on page 12-33. Updated "Correlation Report" on page 12-41. Added hyperlinks to referenced documents and websites throughout the chapter. Made minor editorial updates.



Related Links

[Documentation Archive](#)

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



7 Mentor Graphics PCB Design Tools Support

You can integrate the Mentor Graphics[®] I/O Designer or DxDesigner PCB design tools into the Intel Quartus Prime design flow. This combination provides a complete FPGA-to-board design workflow.

With today's large, high-pin-count and high-speed FPGA devices, good and correct PCB design practices are essential to ensure correct system operation. The PCB design takes place concurrently with the design and programming of the FPGA. The FPGA or ASIC designer initially creates signal and pin assignments, and the board designer must correctly transfer these assignments to the symbols in their system circuit schematics and board layout. As the board design progresses, Intel recommends reassigning pins to optimize the PCB layout. Ensure that you inform the FPGA designer of the pin reassignments so that the new assignments are included in an updated placement and routing of the design.

The Mentor Graphics I/O Designer software allows you to take advantage of the full FPGA symbol design, creation, editing, and back-annotation flow supported by the Mentor Graphics tools.

This chapter covers the following topics:

- Mentor Graphics and Intel software integration flow
- Generating supporting files
- Adding Intel Quartus Prime I/O assignments to I/O Designer
- Updating assignment changes between the I/O Designer the Intel Quartus Prime software
- Generating I/O Designer symbols
- Creating DxDesigner symbols from the Intel Quartus Prime output files

This chapter is intended for board design and layout engineers who want to start the FPGA board integration while the FPGA is still in the design phase. Alternatively, the board designer can plan the FPGA pin-out and routing requirements in the Mentor Graphics tools and pass the information back to the Intel Quartus Prime software for placement and routing. Part librarians can also benefit from this chapter by learning how to use output from the Intel Quartus Prime software to create new library parts and symbols.

The procedures in this chapter require the following software:

- The Intel Quartus Prime software version 5.1 or later
- DxDesigner software version 2004 or later
- Mentor Graphics I/O Designer software (optional)

Note: To obtain and license the Mentor Graphics tools and for product information, support, and training, refer to the Mentor Graphics website.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

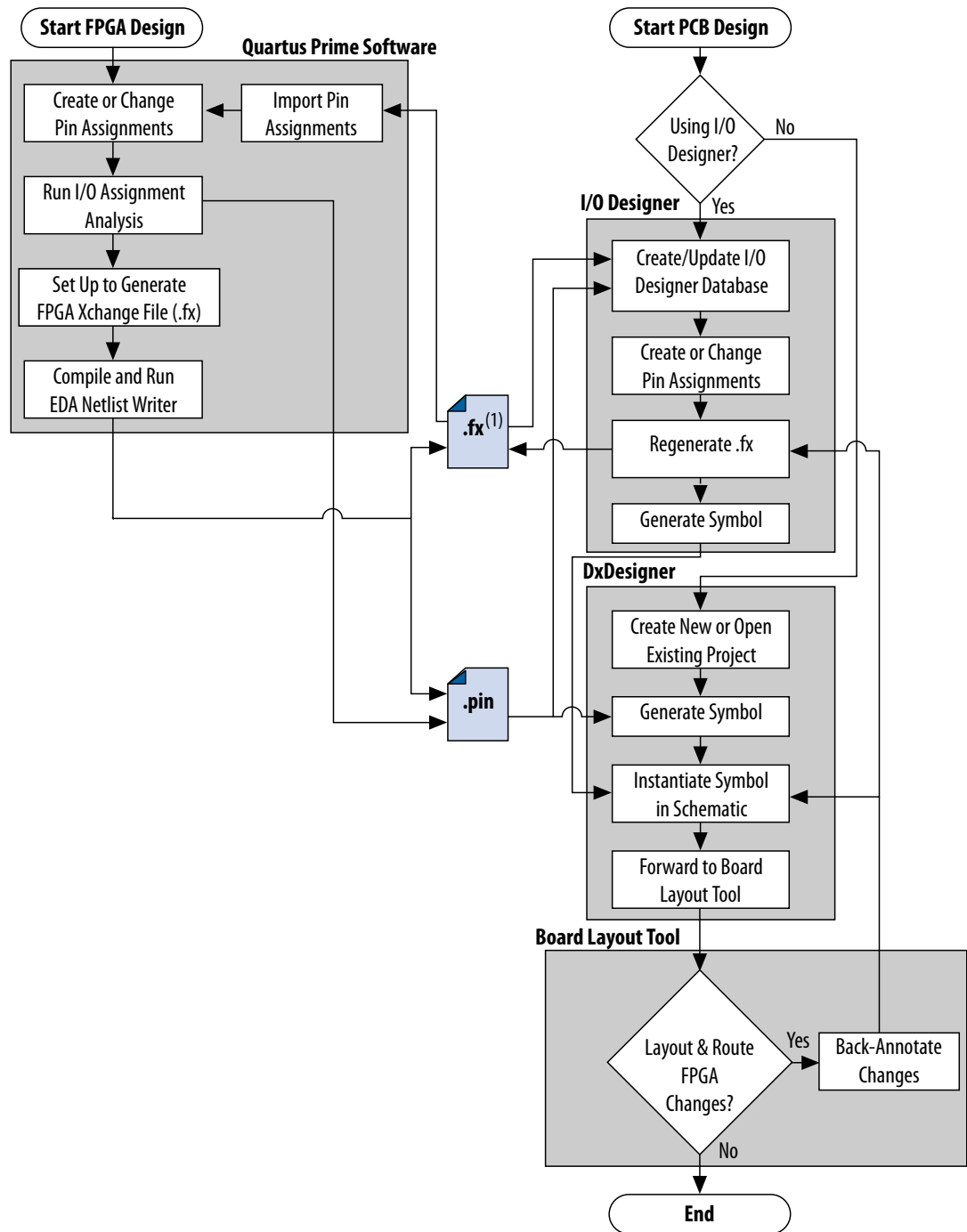
ISO
9001:2008
Registered



7.1 FPGA-to-PCB Design Flow

You can create a design flow integrating an Intel FPGA design from the Intel Quartus Prime software, and a circuit schematic in the DxDesigner software.

Figure 43. Design Flow with and Without the I/O Designer Software



Note: The Intel Quartus Prime software generates the **.fx** in the output directory you specify in the **Board-Level** page of the **Settings** dialog box. However, the Intel Quartus Prime software and the I/O Designer software can import pin assignments from an **.fx** located in any directory. Use a backup **.fx** to prevent overwriting existing assignments or importing invalid assignments.



To integrate the I/O Designer into your design flow, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments** > **Settings** > **EDA Tool Settings** > **Board-Level** to specify settings for **.fx** symbol file generation.
2. Compile your design to generate the **.fx** and Pin-Out File (**.pin**) in the Intel Quartus Prime project directory.
3. Create a board design with the DxDesigner software and the I/O Designer software by performing the following steps:
 - a. Create a new I/O Designer database based on the **.fx** and the **.pin** files.
 - b. In the I/O Designer software, make adjustments to signal and pin assignments.
 - c. Regenerate the **.fx** in the I/O Designer software to export the I/O Designer software changes to the Intel Quartus Prime software.
 - d. Generate a single or fractured symbol for use in the DxDesigner software.
 - e. Add the symbol to the **sym** directory of a DxDesigner project, or specify a new DxDesigner project with the new symbol.
 - f. Instantiate the symbol in your DxDesigner schematic and export the design to the board layout tool.
 - g. Back-annotate pin changes created in the board layout tool to the DxDesigner software and back to the I/O Designer software and the Intel Quartus Prime software.
4. Create a board design with the DxDesigner software without the I/O Designer software by performing the following steps:
 - a. Create a new DxBoardLink symbol with the **Symbol** wizard and reference the **.pin** from the Intel Quartus Prime software in an existing DxDesigner project.
 - b. Instantiate the symbol in your DxDesigner schematic and export the design to a board layout tool.

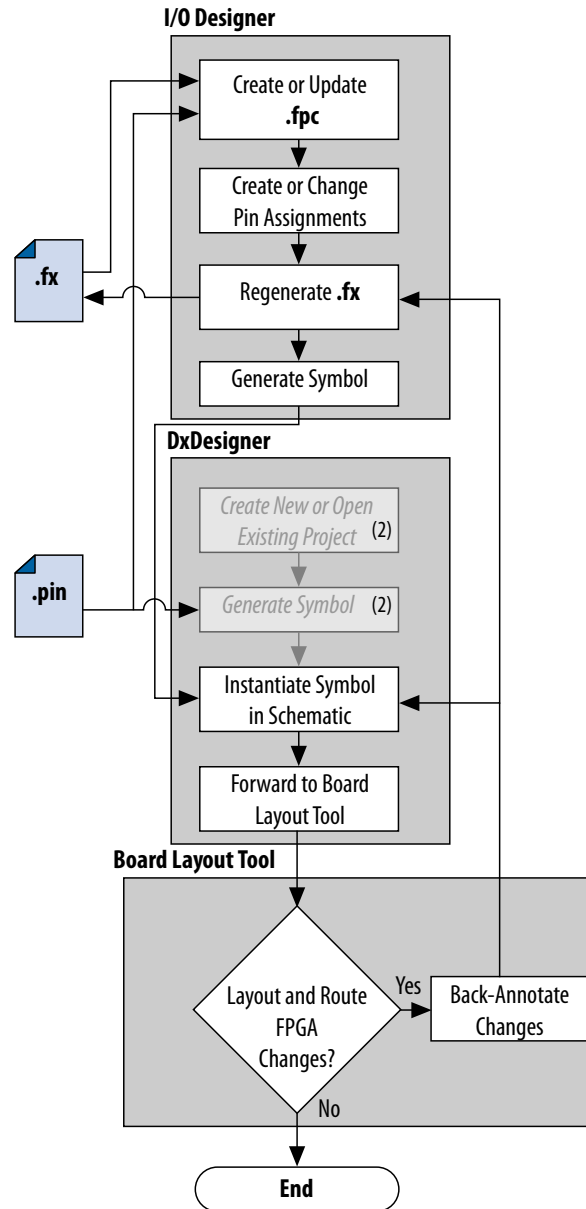
Note: You can update these symbols with design changes with or without the I/O Designer software. If you use the Mentor Graphics I/O Designer software and you change symbols with the DxDesigner software, you must reimport the symbols into I/O Designer to avoid overwriting your symbol changes.

7.2 Integrating with I/O Designer

You can integrate the Mentor Graphics I/O Designer software into the Intel Quartus Prime design flow. Pin and signal assignment changes can be made anywhere in the design flow with either the Intel Quartus Prime Pin Planner or the I/O Designer software. The I/O Designer software facilitates moving these changes, as well as synthesis, placement, and routing changes, between the Intel Quartus Prime software, an external synthesis tool (if used), and a schematic capture tool such as the DxDesigner software.

This section describes how to use the I/O Designer software to transfer pin and signal assignment information to and from the Intel Quartus Prime software with an **.fx**, and how to create symbols for the DxDesigner software.

Figure 44. I/O Designer Design Flow



Note: (2) DxDesigner software-specific steps in the design flow are not part of the I/O Designer flow.

7.2.1 Generating Pin Assignment Files

You transfer I/O pin assignments from the Intel Quartus Prime software to the Mentor Graphics PCB tools by generating optional **.pin** and **.fx** files during Intel Quartus Prime compilation. These files contain pin assignment information for use in other tools.



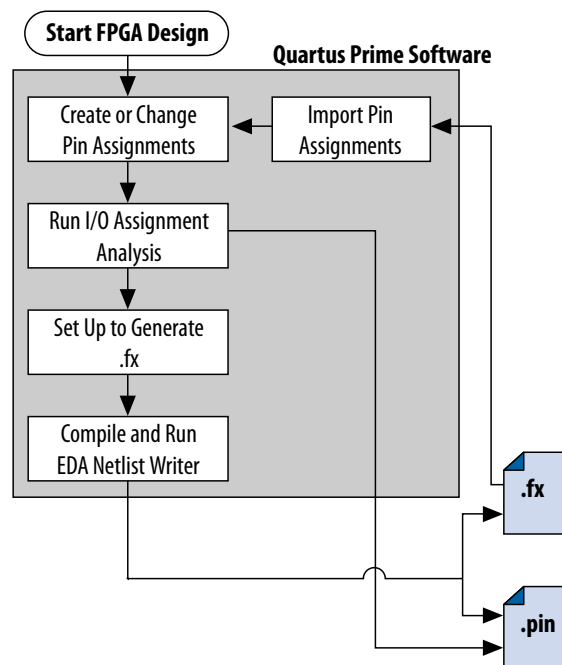
Click **Assignments** > **Settings** > **Board-Level** to specify settings for optional PCB tool file generation. Click **Processing** > **Start Compilation** to compile the design to generate the file(s) in the project directory.

The Intel Quartus Prime-generated **.pin** contains the I/O pin name, number, location, direction, and I/O standard for all used and unused pins in the design. Click **Assignments** > **Pin Planner** to modify I/O pin assignments. You cannot import pin assignment changes from a Mentor Graphics **.pin** into the Intel Quartus Prime software.

The **.fx** is an input or output of either the Intel Quartus Prime or I/O Designer software. You can generate an **.fx** in the Intel Quartus Prime software for symbol generation in the Mentor Graphics I/O Designer software. An Intel Quartus Prime **.fx** contains the pin name, number, location, direction, I/O standard, drive strength, termination, slew rate, IOB delay, and differential pins. An I/O Designer **.fx** additionally includes information about unused pins and pin set groups.

The I/O Designer software can also read from or update an Intel Quartus Prime Settings File (**.qsf**). You can use the **.qsf** in the same way as use of the **.fx**, but pin swap group information does not transfer between I/O Designer and the Intel Quartus Prime software. Use the **.fx** rather than the **.qsf** for transferring I/O assignment information.

Figure 45. Generating **.pin** and **.fx** files



7.2.2 I/O Designer Settings

You can directly export I/O Designer symbols to the DxDesigner software. To set options for integrating I/O Designer with Dx Designer, follow these steps:



1. Start the I/O Designer software.
2. Click **Tools > Preferences**.
3. Click **Paths**, and then double-click the **DxDesigner executable file** path field to select the location of the DxDesigner application.
4. Click **Apply**.
5. Click **Symbol Editor**, and then click **Export**. In the Export type menu, under **General**, select **DxDesigner/PADS-Designer**.
6. Click **Apply**, and then click **OK**.
7. Click **File > Properties**.
8. Click the **PCB Flow** tab, and then click **Path to a DxDesigner project directory**.
9. Click **OK**.
If you do not have a new DxDesigner project in the Database wizard and a DxDesigner project, you must create a new database with the DxDesigner software, and then specify the project location in I/O Designer.

7.2.3 Transferring I/O Assignments

You can transfer Intel Quartus Prime signal and pin assignments contained in **.pin** and **.fx** files into an I/O Designer database. Use the I/O Designer Database Wizard to create a new database incorporating the **.fx** and **.pin** files. You can also create a new, empty database and manually add the assignment information. If there is no available signal or pin assignment information, you can create an empty database containing only a selection of the target device. This technique is useful if you know the signals in your design and the pins you want to assign. You can subsequently transfer this information to the Intel Quartus Prime software for placement and routing.

You may create a very simple I/O Designer database that includes only the **.pin** or **.fx** file information. However, when using only a **.pin**, you cannot import I/O assignment changes from I/O Designer back into the Intel Quartus Prime software without also generating an **.fx**. If your I/O Designer database includes only **.fx** file information, the database may not contain all the available I/O assignment information. The Intel Quartus Prime **.fx** file only lists assigned pins. The **.pin** lists all assigned and unassigned device pins. Use both the **.pin** and the **.fx** to produce the most complete set of I/O Designer database information.

To create a new I/O Designer database using the Database wizard, follow these steps;

1. Start the I/O Designer software. The **Welcome to I/O Designer** dialog box appears. Select **Wizard to create new database** and click **OK**.
If the **Welcome to I/O Designer** dialog box does not appear, you can access the wizard through the menu. To access the wizard, click **File > Database Wizard**.
2. Click **Next**. The **Define HDL source file** page appears



If no HDL files are available, or if the **.fx** contains your signal and pin assignments, you can skip Step 3 and proceed to Step 4.

3. If your design includes a Verilog HDL or VHDL file, you can add a top-level Verilog HDL or VHDL file in the I/O Designer software. Adding a file allows you to create functional blocks or get signal names from your design. You must create all physical pin assignments in I/O Designer if you are not using an **.fx** or a **.pin**. Click **Next**. The **Database Name** page appears.
4. In the **Database Name** page, type your database file name. Click **Next**. The Database Location window appears.
5. Add a path to the new or an existing database in the **Location** field, or browse to a database location. Click **Next**. The **FPGA flow** page appears.
6. In the Vendor menu, click **Altera**.
7. In the Tool/Library menu, click **Intel Quartus Prime <version>** to select your version of the Intel Quartus Prime software.

Note: The Intel Quartus Prime software version listed may not match your actual software version. If your version is not listed, select the latest version. If your target device is not available, the device may not yet be supported by the I/O Designer software.

8. Select the appropriate device family, device, package, and speed (if applicable), from the corresponding menus. Click **Next**. The **Place and route** page appears.
9. In the **FPGAX file name** field, type or browse to the backup copy of the **.fx** generated by the Intel Quartus Prime software.
10. In the **Pin report file name** field, type or browse to the **.pin** generated by the Intel Quartus Prime software. Click **Next**.
You can also select a **.qsf** for update. The I/O Designer software can update the pin assignment information in the **.qsf** without affecting any other information in the file.

Note: You can import a **.pin** without importing an **.fx**. The I/O Designer software does not generate a **.pin**. To transfer assignment information to the Intel Quartus Prime software, select an additional file and file type. Intel recommends selecting an **.fx** in addition to a **.pin** for transferring all the assignment information in the **.fx** and **.pin** files. In some versions of the I/O Designer software, the standard file picker may incorrectly look for a **.pin** instead of an **.fx**. In this case, select **All Files (*.*)** from the **Save as type** list and select the file from the list.

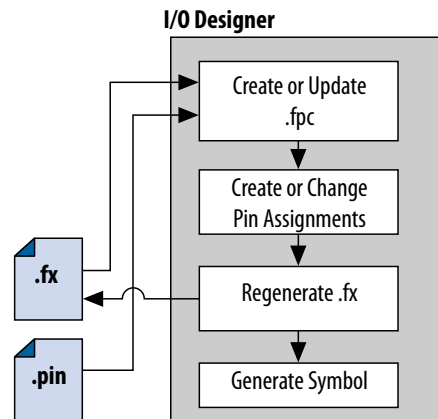
11. On the **Synthesis** page, specify an external synthesis tool and a synthesis constraints file for use with the tool. If you do not use an external synthesis tool, click **Next**.
12. On the **PCB Flow** page, you can select an existing schematic project or create a new project as a symbol information destination.

- To select an existing project, select Choose existing project and click Browse after the Project Path field. The Select project dialog box appears. Select the project.
 - To create a new project, in the Select project dialog box, select Create new empty project. Type the project file name in the Name field and browse to the location where you want to save the file. Click OK.
13. If you have not specified a design tool to which you can send symbol information in the I/O Designer software, click **Advanced** in the **PCB Flow** page and select your design tool. If you select the DxDesigner software, you have the option to specify a Hierarchical Occurrence Attributes (.oat) file to import into the I/O Designer software. Click **Next** and then click **Finish** to create the database.Updating

7.2.4 Updating I/O Designer with Intel Quartus Prime Pin Assignments

As you fine tune your design in the Intel Quartus Prime software, changes to design logic and pin assignments are likely. You must transfer any pin assignment changes made during design iterations for correct analysis in your circuit schematic and board layout tools. You transfer Intel Quartus Prime pin assignment changes to I/O Designer by updating the .fx and the .pin files in the Intel Quartus Prime software. When you update the .fx or the .pin, the I/O Designer database imports the changes automatically when configured according to the following instructions.

Figure 46. Updating Intel Quartus Prime Pin Assignments in I/O Designer



To update the .fx in your selected output directory and the .pin in your project directory after making changes to the design, perform the following tasks:

1. In the I/O Designer software, click **File > Properties**.
2. Under **FPGA Xchange**, specify the .fx file name and location.
3. Under **Place and Route**, specify the .pin file name and location. After you have set up these file locations, the I/O Designer software monitors these files for changes. If the specified .fx or .pin is modified during design processing, three indicators flash red in the lower right corner of the I/O Designer GUI. You can click the indicators to open the **I/O Designer Update Wizard** dialog box. The **I/O Designer Update Wizard** dialog box lists the updated files in the database.
- 4.



Make logic or pin assignment changes in your design.

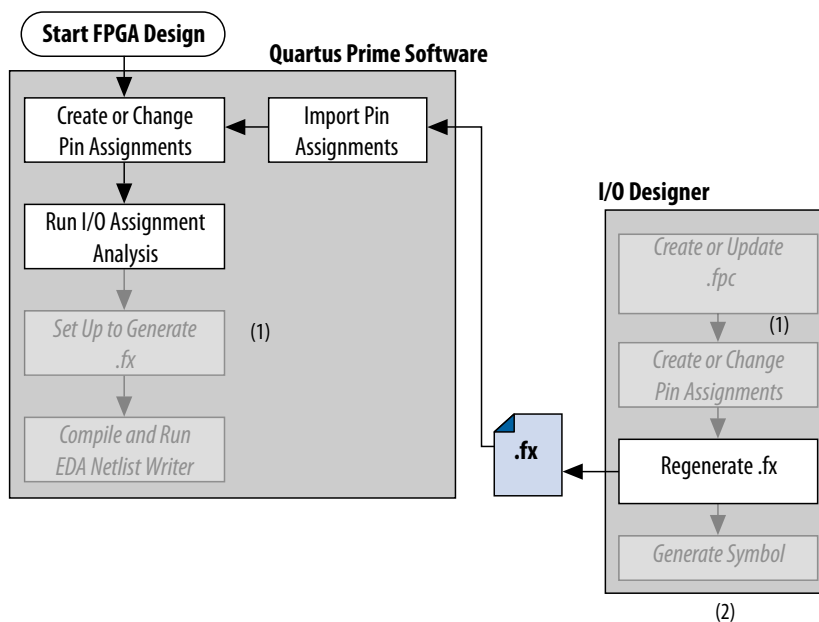
5. Click **Processing > Start > Start I/O Assignment Analysis** to validate your latest assignment changes.
6. To preserve your changes and update the corresponding the **.fx** and **.pin** files, click **Processing > Start > Start EDA Netlist Writer** or **Processing > Start Compilation**.

Note: Your I/O Designer database should use a backup copy of the **.fx** generated by the Intel Quartus Prime software. Otherwise, updating the file in the Intel Quartus Prime software overwrites any changes made to the file by the I/O Designer software. If there are I/O Designer assignments in the **.fx** that you want to preserve, create a backup copy of the file before updating it in the Intel Quartus Prime software, and verify that your I/O Designer database points to the backup copy.

7.2.5 Updating Intel Quartus Prime with I/O Designer Pin Assignments

As you fine tune your board design in I/O Designer, changes to signal routing and layout are likely. You must import any routing and layout changes into the Intel Quartus Prime software for accurate place and route to match the new pin-out. The I/O Designer tool supports this flow.

Figure 47. Importing I/O Designer Pin Assignments



To import I/O Designer pin assignments, follow these steps:

1. Make pin assignment changes directly in the I/O Designer software, or the software can automatically update changes made in a board layout tool that are back-annotated to a schematic entry program such as the DxDesigner software.
2. To update the **.fx** with the changes, click **Generate > FPGA Xchange File**.
3. Open your Intel Quartus Prime project.
4. Click **Assignments > Import Assignments**.



5. (Optional) To preserve original assignments before import, turn on **Copy existing assignments into <project name>.qsf.bak** before importing before importing the **.fx**.
6. Select the **.fx** and click **Open**.
7. Click **OK**.

7.2.6 Generating Schematic Symbols in I/O Designer

Circuit board schematic creation is one of the first tasks required in the design of a new PCB. You can use the I/O Designer software to generate schematic symbols for your Intel Quartus Prime FPGA design for use in the DXDesigner schematic entry tools. The I/O Designer software can generate symbols for use in various Mentor Graphics schematic entry tools, and can import changes back-annotated by board layout tools to update the database and update the Intel Quartus Prime software with the **.fx**

Most FPGA devices contain hundreds of pins, requiring large schematic symbols that may not fit on a single schematic page. Symbol designs in the I/O Designer software can be split or fractured into various functional blocks, allowing multiple part fractures on the same schematic page or across multiple pages. In the DxDesigner software, these part fractures join together with the use of the **HETERO** attribute.

You can use the I/O Designer **Symbol** wizard to quickly create symbols that you can subsequently refine. Alternatively, you can import symbols from another DXDesigner project, and then assign an FPGA to the symbol. To import symbols in the I/O Designer software, **File > Import Symbol**.

I/O Designer symbols are either functional, physical (PCB), or both. Signals imported into the database, usually from Verilog HDL or VHDL files, are the basis of a functional symbol. No physical device pins must be associated with the signals to generate a functional symbol. This section focuses on board-level PCB symbols with signals directly mapped to physical device pins through assignments in either the Intel Quartus Prime Pin Planner or in the I/O Designer database.

7.2.6.1 Generating Schematic Symbols

To create a symbol based on a selected Intel FPGA device, follow these steps:

1. Start the I/O Designer software.
2. Click **Symbol > Symbol Wizard**.
3. In the **Symbol name** field, type the symbol name. The **DEVICE** and **PKG_TYPE** fields display the device and package information.
Note: If **DEVICE** and **PKG_TYPE** are blank or incorrect, close the Symbol wizard and specify the correct device information (**File > Properties > FPGA Flow**).
4. Under **Symbol type**, click **PCB**. Under **Use signals**, click **All**, then click **Next**.
5. Select fracturing options for your symbol. If you are using the Symbol wizard to edit a previously created fractured symbol, you must turn on **Reuse existing fractures** to preserve your current fractures. Select other options on this page as appropriate for your symbol. Click **Next**.
6. Select additional fracturing options for your symbol. Click **Next**.



7. Select the options for the appearance of the symbols. Click **Next**.
8. Define the information you want to label for the entire symbol and for individual pins. Click **Next**.
9. Add any additional signals and pins to the symbol. Click **Finish**.
You can view your symbol and any fractures you created with the Symbol Editor. You can edit parts of the symbol, delete fractures, or rerun the Symbol wizard. When you modify pin assignments in I/O Designer database, I/O Designer symbols automatically reflect these changes. Modify assignments in the I/O Designer software by supplying and updated **.fx** from the Intel Quartus Prime software, or by back-annotating changes in your board layout tool.

7.2.7 Exporting Schematic Symbols to DxDesigner

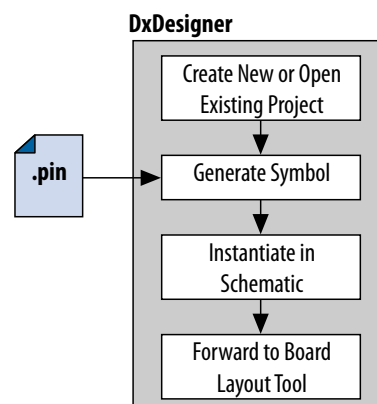
You can export your I/O Designer schematic symbols for to DxDesigner for further design entry work. To generate all fractures of a symbol, click **Generate > All Symbols**. To generate only the currently displayed symbol, click **Generate > Current Symbol Only**. The DxDesigner project **/sym** directory preserves each symbol in the database as a separate file. You can instantiate the symbols in your DxDesigner schematics.

7.3 Integrating with DxDesigner

You can integrate the Mentor Graphics DxDesigner schematic capture tool into the Intel Quartus Prime design flow. Use DxDesigner to create flat circuit schematics or to create hierarchical schematics that facilitate design reuse and a team-based design for all PCB types. Use DxDesigner in conjunction with I/O Designer software for a complete FPGA I/O and PCB design flow.

If you use DxDesigner without the I/O Designer software, the design flow is one-way, using only the **.pin** generated by the Intel Quartus Prime software. You can only make signal and pin assignment changes in the Intel Quartus Prime software. You cannot back-annotate changes made in a board layout tool or in a DxDesigner symbol to the Intel Quartus Prime software.

Figure 48. DxDesigner-only Flow (without I/O Designer)



7.3.1 DxDesigner Project Settings

DxDesigner new projects automatically create FPGA symbols by default. However, if you are using the I/O Designer with DxDesigner, you must enable DxBoardLink Flow options for integration with the I/O Designer software. To enable the DxBoardLink flow design configuration when creating a new DxDesigner project, follow these steps:

1. Start the DxDesigner software.
2. Click **File** ► **New**, and then click the **Project** tab.
3. Click **More**. Turn on **DxBoardLink**. To enable the DxBoardLink Flow design configuration for an existing project, click **Design Configurations** in the Design Configuration toolbar and turn on **DxBoardLink**.

7.3.2 Creating Schematic Symbols in DxDesigner

You can create schematic symbols in the DxDesigner software manually or with the Symbol wizard. The DxDesigner Symbol wizard is similar to the I/O Designer Symbol wizard, but with fewer fracturing options. The DxDesigner Symbol wizard creates, fractures, and edits FPGA symbols based on the specified Intel device. To create a symbol with the Symbol wizard, follow these steps;

1. Start the DxDesigner software.
2. Click **Symbol Wizard** in the toolbar.
3. Type the new symbol name in the name field and click **OK**.
4. Specify creation of a new symbol or modification of an existing symbol. To modify an existing symbol, specify the library path or alias, and select the existing symbol. To create a new symbol, select DxBoardLink for the symbol source. The DxDesigner block type defaults to Module because the FPGA design does not have an underlying DxDesigner schematic. Choose whether or not to fracture the symbol. Click **Next**.
5. Type a name for the symbol, an overall part name for all the symbol fractures, and a library name for the new library created for this symbol. By default, the part and library names are the same as the symbol name. Click **Next**.
6. Specify the appearance of the generated symbol and how itthe grid you have set in your DxDesigner project schematic. After making your selections. Click **Next**.
7. In the **FPGA vendor** list, select **Intel Quartus**. In the **Pin-Out file to import** field, select the **.pin** from your Intel Quartus Prime project directory. You can also specify Fracturing Scheme, Bus pin, and Power pin options. Click **Next**.
8. Select to create or modify symbol attributes for use in the DxDesigner software. Click **Next**.
9. On the **Pin Settings** page, make any final adjustments to pin and label location and information. Each tabbed spreadsheet represents a fracture of your symbol. Click **Save Symbol**.

After creating the symbol, you can examine and place any fracture of the symbol in your schematic. You can locate separate files of all the fractures you created in the library you specified or created in the **/sym** directory in your DxDesigner project. You can add the symbols to your schematics or you can manually edit the symbols or with the Symbol wizard.



7.4 Analyzing FPGA Simultaneous Switching Noise (SSN)

With the Intel Quartus Prime software, you can extract pin assignment data and perform SSN analysis of your design. Perform SSN analysis early in the board layout stage as part of your overall pin planning process. Use the Intel Quartus Prime SSN Analyzer to optimize the pin assignments for better SSN performance.

7.5 Scripting API

The I/O Designer software includes a command line Tcl interpreter. All commands input through the I/O Designer GUI translate into Tcl commands run by the tool. You can run individual Tcl commands or scripts in the I/O Designer Console window, rather than using the GUI.

You can use the following Tcl commands to control I/O Designer.

- `set_fpga_xchange_file <file name>`—specifies the **.fx** from which the I/O Designer software updates assignments.
- `update_from_fpga_xchange_file`—updates the I/O Designer database with assignment updates from the currently specified **.fx**.
- `generate_fpga_xchange_file`—updates the **.fx** with I/O Designer software changes for transfer back into the Intel Quartus Prime software.
- `set_pin_report_file -quartus_pin <file name>`—imports assignment data from an Intel Quartus Prime software **.pin** file.
- `symbolwizard`—runs the I/O Designer Symbol wizard.
- `set_dx_designer_project -path <path>`

7.6 Document Revision History

Table 24. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	<ul style="list-style-type: none"> • Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
2014.06.30	14.0.0	<ul style="list-style-type: none"> • Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor.
June 2012	12.0.0	<ul style="list-style-type: none"> • Removed survey link.
December 2010	10.1.0	<ul style="list-style-type: none"> • Changed to new document template.



Related Links

[Documentation Archive](#)

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



8 Cadence PCB Design Tools Support

8.1 Cadence PCB Design Tools Support

The Intel Quartus Prime software interacts with the following software to provide a complete FPGA-to-board integration design workflow: the Cadence Allegro Design Entry HDL software and the Cadence Allegro Design Entry CIS (Component Information System) software (also known as OrCAD Capture CIS). The information is useful for board design and layout engineers who want to begin the FPGA board integration process while the FPGA is still in the design phase. Part librarians can also benefit by learning the method to use output from the Intel Quartus Prime software to create new library parts and symbols.

With today's large, high-pin-count and high-speed FPGA devices, good PCB design practices are important to ensure the correct operation of your system. The PCB design takes place concurrently with the design and programming of the FPGA. An FPGA or ASIC designer initially creates the signal and pin assignments and the board designer must transfer these assignments to the symbols used in their system circuit schematics and board layout correctly. As the board design progresses, you must perform pin reassignments to optimize the layout. You must communicate pin reassignments to the FPGA designer to ensure the new assignments are processed through the FPGA with updated placement and routing.

You require the following software:

- The Intel Quartus Prime software version 5.1 or later
- The Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software version 15.2 or later
- The OrCAD Capture software with the optional CIS option version 10.3 or later (optional)

Note: These programs are very similar because the Cadence Allegro Design Entry CIS software is based on the OrCAD Capture software. Any procedural information can also apply to the OrCAD Capture software unless otherwise noted.

Related Links

- www.cadence.com
For more information about obtaining and licensing the Cadence tools and for product information, support, and training
- www.orcad.com
For more information about the OrCAD Capture software and the CIS option
- www.ema-eda.com
For more information about Cadence and OrCAD support and training.



8.2 Product Comparison

Table 25. Cadence and OrCAD Product Comparison

Description	Cadence Allegro Design Entry HDL	Cadence Allegro Design Entry CIS	OrCAD Capture CIS
Former Name	Concept HDL Expert	Capture CIS Studio	—
History	More commonly known by its former name, Cadence renamed all board design tools in 2004 under the Allegro name.	Based directly on OrCAD Capture CIS, the Cadence Allegro Design Entry CIS software is still developed by OrCAD but sold and marketed by Cadence. EMA provides support and training.	The basis for Design Entry CIS is still developed by OrCAD for continued use by existing OrCAD customers. EMA provides support and training for all OrCAD products.
Vendor Design Flow	Cadence Allegro 600 series, formerly known as the Expert Series, for high-end, high-speed design.	Cadence Allegro 200 series, formerly known as the Studio Series, for small- to medium-level design.	—

Related Links

- www.cadence.com
- www.ema-eda.com

8.3 FPGA-to-PCB Design Flow

You can create a design flow integrating an Intel FPGA design from the Intel Quartus Prime software through a circuit schematic in the Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software.



Figure 49. Design Flow with the Cadence Allegro Design Entry HDL Software

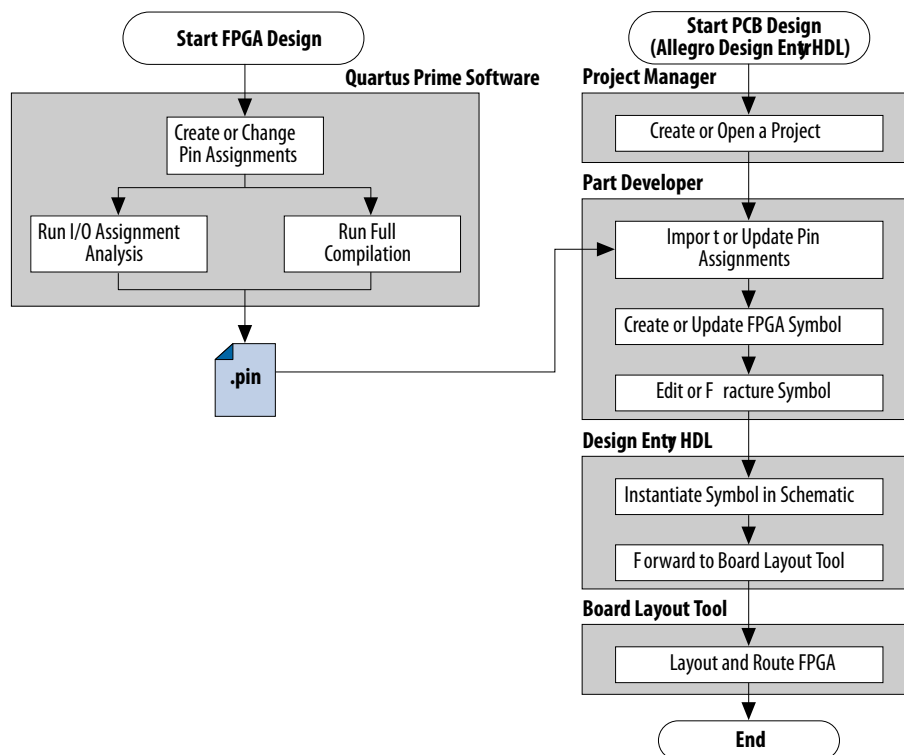
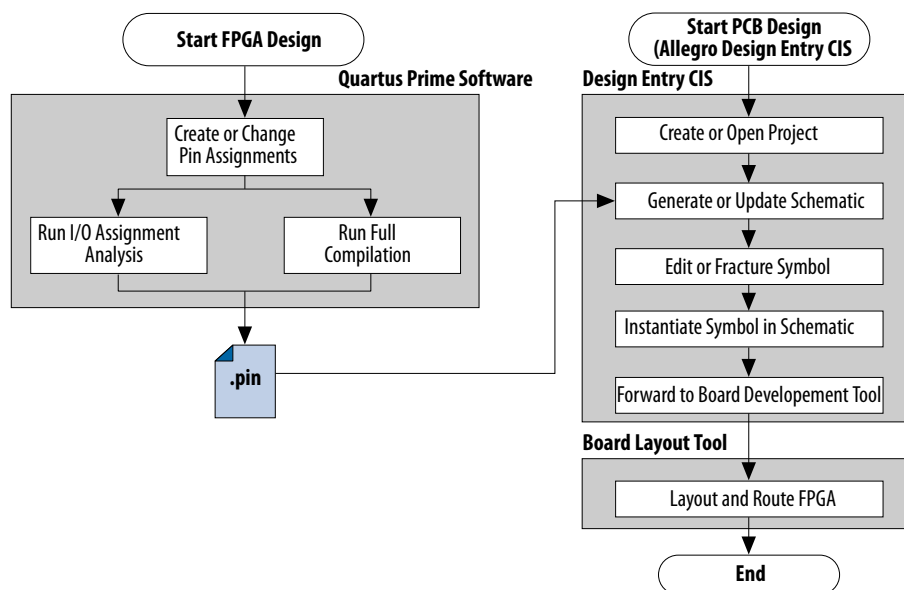


Figure 50. Design Flow with the Cadence Allegro Design Entry CIS Software



To create FPGA symbols using the Cadence Allegro PCB Librarian Part Developer tool, you must obtain the Cadence PCB Librarian Expert license. You can update symbols with changes made to the FPGA design using any of these tools.



8.3.1 Integrating Intel FPGA Design

To integrate an Intel FPGA design starting in the Intel Quartus Prime software through to a circuit schematic in the Cadence Allegro Design Entry HDL software or the Cadence Allegro Design Entry CIS software, follow these steps:

1. In the Intel Quartus Prime software, compile your design to generate a Pin-Out File (**.pin**) to transfer the assignments to the Cadence software.
2. If you are using the Cadence Allegro Design Entry HDL software for your schematic design, follow these steps:
 - a. Open an existing project or create a new project in the Cadence Allegro Project Manager tool.
 - b. Construct a new symbol or update an existing symbol using the Cadence Allegro PCB Librarian Part Developer tool.
 - c. With the Cadence Allegro PCB Librarian Part Developer tool, edit your symbol or fracture it into smaller parts (optional).
 - d. Instantiate the symbol in your Cadence Allegro Design Entry HDL software schematic and transfer the design to your board layout tool.or
If you are using the Cadence Allegro Design Entry CIS software for your schematic design, follow these steps:
 - e. Generate a new part in a new or existing Cadence Allegro Design Entry CIS project, referencing the **.pin** output file from the Intel Quartus Prime software. You can also update an existing symbol with a new **.pin**.
 - f. Split the symbol into smaller parts as necessary.
 - g. Instantiate the symbol in your Cadence Allegro Design Entry CIS schematic and transfer the design to your board layout tool.

8.3.2 Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA

With the Intel Quartus Prime software, you can extract pin assignment data and perform SSN analysis of your FPGA design for designs targeting the Stratix III device family.

You can analyze SSN in your device early in the board layout stage as part of your overall pin planning process; however, you do not have to perform SSN analysis to generate pin assignment data from the Intel Quartus Prime software. You can use the SSN Analyzer tool to optimize the pin assignments for better SSN performance of your device.

Related Links

[Simultaneous Switching Noise \(SSN\) Analysis and Optimizations](#) on page 48

8.4 Setting Up the Intel Quartus Prime Software

You can transfer pin and signal assignments from the Intel Quartus Prime software to the Cadence design tools by generating the Intel Quartus Prime project **.pin**. The **.pin** is an output file generated by the Intel Quartus Prime Fitter containing pin



assignment information. You can use the Intel Quartus Prime Pin Planner to set and change the assignments in the `.pin` and then transfer the assignments to the Cadence design tools. You cannot, however, import pin assignment changes from the Cadence design tools into the Intel Quartus Prime software with the `.pin`.

The `.pin` lists all used and unused pins on your selected Intel device. The `.pin` also provides the following basic information fields for each assigned pin on the device:

- Pin signal name and usage
- Pin number
- Signal direction
- I/O standard
- Voltage
- I/O bank
- User or Fitter-assigned

Related Links

[I/O Management](#) on page 24

For more information about using the Intel Quartus Prime Pin Planner to create or change pin assignment details.

8.4.1 Generating a `.pin` File

To generate a `.pin`, follow these steps:

1. Compile your design.
2. Locate the `.pin` in your Intel Quartus Prime project directory with the name `<project name>.pin`.

Related Links

[I/O Management](#) on page 24

For more information about pin and signal assignment transfer and the files that the Intel Quartus Prime software can import and export.

8.5 FPGA-to-Board Integration with the Cadence Allegro Design Entry HDL Software

The Cadence Allegro Design Entry HDL software is a schematic capture tool and is part of the Cadence 600 series design flow. Use the Cadence Allegro Design Entry HDL software to create flat circuit schematics for all types of PCB design. The Cadence Allegro Design Entry HDL software can also create hierarchical schematics to facilitate design reuse and team-based design. With the Cadence Allegro Design Entry HDL software, the design flow from FPGA-to-board is one-way, using only the `.pin` generated by the Intel Quartus Prime software. You can only make signal and pin assignment changes in the Intel Quartus Prime software and these changes reflect as updated symbols in a Cadence Allegro Design Entry HDL project.

For more information about the design flow with the Cadence Allegro Design Entry HDL software, refer to [Figure 49](#) on page 161.



Note: Routing or pin assignment changes made in a board layout tool or a Cadence Allegro Design Entry HDL software symbol cannot be back-annotated to the Intel Quartus Prime software.

Related Links

www.cadence.com

Provides information about the Cadence Allegro Design Entry HDL software and the Cadence Allegro PCB Librarian Part Developer tool, including licensing, support, usage, training, and product updates.

8.5.1 Creating Symbols

In addition to circuit simulation, circuit board schematic creation is one of the first tasks required when designing a new PCB. Schematics must understand how the PCB works, and to generate a netlist for a board layout tool for board design and routing. The Cadence Allegro PCB Librarian Part Developer tool allows you to create schematic symbols based on FPGA designs exported from the Intel Quartus Prime software.

You can create symbols for the Cadence Allegro Design Entry HDL project with the Cadence Allegro PCB Librarian Part Developer tool, which is available in the Cadence Allegro Project Manager tool. Intel recommends using the Cadence Allegro PCB Librarian Part Developer tool to import FPGA designs into the Cadence Allegro Design Entry HDL software.

You must obtain a PCB Librarian Expert license from Cadence to run the Cadence Allegro PCB Librarian Part Developer tool. The Cadence Allegro PCB Librarian Part Developer tool provides a GUI with many options for creating, editing, fracturing, and updating symbols. If you do not use the Cadence Allegro PCB Librarian Part Developer tool, you must create and edit symbols manually in the Symbol Schematic View in the Cadence Allegro Design Entry HDL software.

Note: If you do not have a PCB Librarian Expert license, you can automatically create FPGA symbols using the programmable IC (PIC) design flow found in the Cadence Allegro Project Manager tool.

Before creating a symbol from an FPGA design, you must open a Cadence Allegro Design Entry HDL project with the Cadence Allegro Project Manager tool. If you do not have an existing Cadence Allegro Design Entry HDL project, you can create one with the Cadence Allegro Design Entry HDL software. The Cadence Allegro Design Entry HDL project directory with the name <project name> .cpm contains your Cadence Allegro Design Entry HDL projects.

While the Cadence Allegro PCB Librarian Part Developer tool refers to symbol fractures as slots, the other tools use different names to refer to symbol fractures.

Table 26. Symbol Fracture Naming Conventions

	Cadence Allegro PCB Librarian Part Developer Tool	Cadence Allegro Design Entry HDL Software	Cadence Allegro Design Entry CIS Software
During symbol generation	Slots	—	Sections
During symbol schematic instantiation	—	Versions	Parts



Related Links

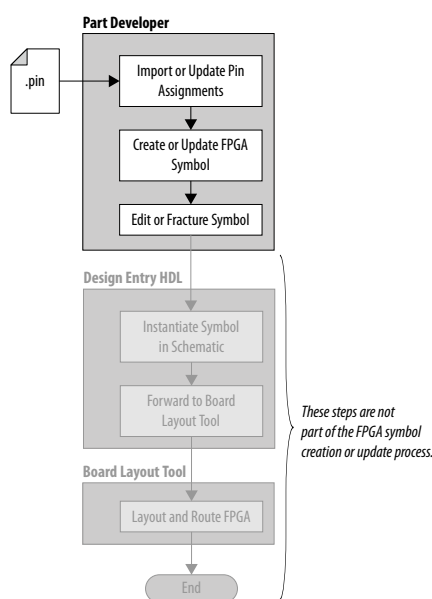
www.cadence.com

Provides information about using the PIC design flow.

8.5.1.1 Cadence Allegro PCB Librarian Part Developer Tool

You can create, fracture, and edit schematic symbols for your designs using the Cadence Allegro PCB Librarian Part Developer tool. Symbols designed in the Cadence Allegro PCB Librarian Part Developer tool can be split or fractured into several functional blocks called slots, allowing multiple smaller part fractures to exist on the same schematic page or across multiple pages.

8.5.1.1.1 Cadence Allegro PCB Librarian Part Developer Tool in the Design Flow



To run the Cadence Allegro PCB Librarian Part Developer tool, you must open a Cadence Allegro Design Entry HDL project in the Cadence Allegro Project Manager tool. To open the Cadence Allegro PCB Librarian Part Developer tool, on the Flows menu, click **Library Management**, and then click **Part Developer**.

Related Links

[FPGA-to-PCB Design Flow](#) on page 160

8.5.1.1.2 Import and Export Wizard

After starting the Cadence Allegro PCB Librarian Part Developer tool, use the **Import and Export** wizard to import your pin assignments from the Intel Quartus Prime software.

Note: Intel recommends using your PCB Librarian Expert license file. To point to your PCB Librarian Expert license file, on the File menu, click **Change Product** and then select the correct product license.



To access the Import and Export wizard, follow these steps:

1. On the File menu, click **Import and Export**.
2. Select **Import ECO-FPGA**, and then click **Next**.
3. In the **Select Source** page of the **Import and Export** wizard, specify the following settings:
 - a. In the **Vendor** list, select **Altera**.
 - b. In the **PnR Tool** list, select **quartusII**.
 - c. In the **PR File** box, browse to select the **.pin** in your Intel Quartus Prime project directory.
 - d. Click **Simulation Options** to select simulation input files.
 - e. Click **Next**.
4. In the **Select Destination** dialog box, specify the following settings:
 - a. Under **Select Component**, click **Generate Custom Component** to create a new component in a library,
or
Click **Use standard component** to base your symbol on an existing component.
Note: Intel recommends creating a new component if you previously created a generic component for an FPGA device. Generic components can cause some problems with your design. When you create a new component, you can place your pin and signal assignments from the Intel Quartus Prime software on this component and reuse the component as a base when you have a new FPGA design.
 - b. In the **Library** list, select an existing library. You can select from the cells in the selected library. Each cell represents all the symbol versions and part fractures for a particular part. In the **Cell** list, select the existing cell to use as a base for your part.
 - c. In the **Destination Library** list, select a destination library for the component. Click **Next**.
 - d. Review and edit the assignments you import into the Cadence Allegro PCB Librarian Part Developer tool based on the data in the **.pin** and then click **Finish**. The location of each pin is not included in the **Preview of Import Data** page of the **Import and Export** wizard, but input pins are on the left side of the created symbol, output pins on the right, power pins on the top, and ground pins on the bottom.

8.5.1.1.3 Editing and Fracturing Symbol

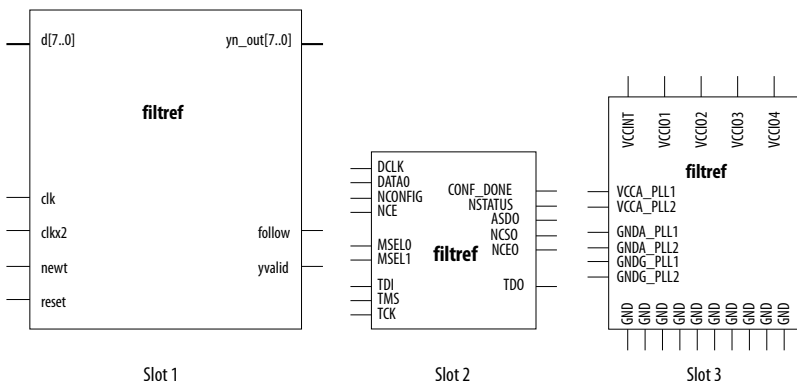
After creating your new symbol in the Cadence Allegro PCB Librarian Part Developer tool, you can edit the symbol graphics, fracture the symbol into multiple slots, and add or change package or symbol properties.

The Part Developer Symbol Editor contains many graphical tools to edit the graphics of a particular symbol. To edit the symbol graphics, select the symbol in the cell hierarchy. The **Symbol Pins** tab appears. You can edit the preview graphic of the symbol in the **Symbol Pins** tab.



Fracturing a Cadence Allegro PCB Librarian Part Developer package into separate symbol slots is useful for FPGA designs. A single symbol for most FPGA packages might be too large for a single schematic page. Splitting the part into separate slots allows you to organize parts of the symbol by function, creating cleaner circuit schematics. For example, you can create one slot for an I/O symbol, a second slot for a JTAG symbol, and a third slot for a power/ground symbol.

Figure 51. Splitting a Symbol into Multiple Slots



- This diagram represents a Cyclone device with JTAG or passive serial (PS) mode configuration option settings. Symbols created for other devices or other configuration modes may have different sets of configuration pins, but can be fractured in a similar manner.
 - The power/ground slot shows only a representation of power and ground pins because the device contains a large number of power and ground pins.

To fracture a part into separate slots, or to modify the slot locations of pins on parts fractured in the Cadence Allegro PCB Librarian Part Developer tool, follow these steps:

1. Start the Cadence Allegro Design Project Manager.
2. On the Flows menu, click **Library Management**.
3. Click **Part Developer**.
4. Click the name of the package you want to change in the cell hierarchy.
5. Click **Functions/Slots**. If you are not creating new slots but want to change the slot location of some pins, proceed to Step 6. If you are creating new slots, click **Add**. A dialog box appears, allowing you to add extra symbol slots. Set the number of extra slots you want to add to the existing symbol, not the total number of desired slots for the part. Click **OK**.
6. Click **Distribute Pins**. Specify the slot location for each pin. Use the checkboxes in each column to move pins from one slot to another. Click **OK**.
7. After distributing the pins, click the **Package Pin** tab and click **Generate Symbol(s)**.
8. Select whether to create a new symbol or modify an existing symbol in each slot. Click **OK**.

The newly generated or modified slot symbols appear as separate symbols in the cell hierarchy. Each of these symbols can be edited individually.

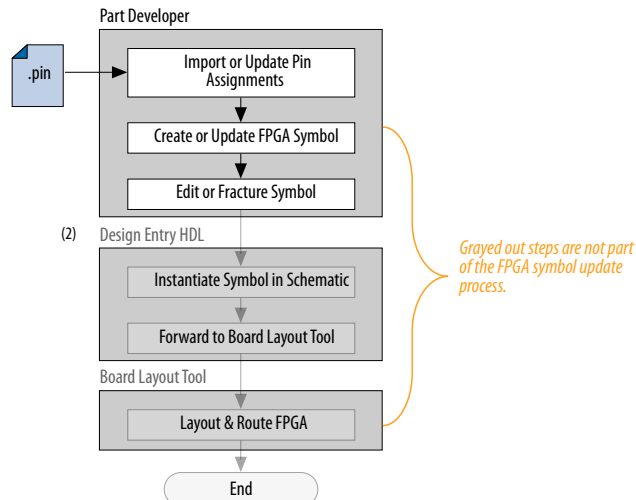
Caution: The Cadence Allegro PCB Librarian Part Developer tool allows you to remap pin assignments in the **Package Pin** tab of the main Cadence Allegro PCB Librarian Part Developer window. If signals remap to different pins in the Cadence Allegro PCB Librarian Part Developer tool, the changes reflect only in regenerated symbols for use in your schematics. You cannot transfer pin assignment changes to the Intel Quartus Prime software from the Cadence Allegro PCB Librarian Part Developer tool, which creates a potential mismatch of the schematic symbols and assignments in the FPGA design. If pin assignment changes are necessary, make the changes in the Intel Quartus Prime Pin Planner instead of the Cadence Allegro PCB Librarian Part Developer tool, and update the symbol as described in the following sections.

For more information about creating, editing, and organizing component symbols with the Cadence Allegro PCB Librarian Part Developer tool, refer to the Part Developer Help.

8.5.1.1.4 Updating FPGA Symbols

As the design process continues, you must make logic changes in the Intel Quartus Prime software, placing signals on different pins after recompiling the design, or use the Intel Quartus Prime Pin Planner to make changes manually. The board designer can request such changes to improve the board routing and layout. To ensure signals connect to the correct pins on the FPGA, you must carry forward these types of changes to the circuit schematic and board layout tools. Updating the `.pin` in the Intel Quartus Prime software facilitates this flow.

Figure 52. Updating the FPGA Symbol in the Design Flow





To update the symbol using the Cadence Allegro PCB Librarian Part Developer tool after updating the `.pin`, follow these steps:

1. On the File menu, click **Import and Export**. The Import and Export wizard appears.
2. In the list of actions to perform, select **Import ECO - FPGA**. Click **Next**. The **Select Source** dialog box appears.
3. Select the updated source of the FPGA assignment information. In the **Vendor** list, select **Altera**. In the **PnR Tool** list, select **quartusII**. In the **PR File** field, click **browse** to specify the updated `.pin` in your Intel Quartus Prime project directory. Click **Next**. The Select Destination window appears.
4. Select the source component and a destination cell for the updated symbol. To create a new component based on the updated pin assignment data, select **Generate Custom Component**. Selecting **Generate Custom Component** replaces the cell listed under the **Specify Library and Cell** name header with a new, nonfractured cell. You can preserve these edits by selecting **Use standard component and select the existing library and cell**. Select the destination library for the component and click **Next**. The **Preview of Import Data** dialog box appears.
5. Make any additional changes to your symbol. Click **Next**. A list of ECO messages appears summarizing the changes made to the cell. To accept the changes and update the cell, click **Finish**.
6. The main Cadence Allegro PCB Librarian Part Developer window appears. You can edit, fracture, and generate the updated symbols as usual from the main Cadence Allegro PCB Librarian Part Developer window.

Note: If the Cadence Allegro PCB Librarian Part Developer tool is not set up to point to your PCB Librarian Expert license file, an error message appears in red at the bottom of the message text window of the Part Developer when you select the **Import and Export** command. To point to your PCB Librarian Expert license, on the File menu, click **Change Product**, and select the correct product license.

Related Links

[FPGA-to-PCB Design Flow](#) on page 160

8.5.2 Instantiating the Symbol in the Cadence Allegro Design Entry HDL Software

To instantiate the symbol in your Cadence Allegro Design Entry HDL schematic after saving the new symbol in the Cadence Allegro PCB Librarian Part Developer tool, follow these steps:

1. In the Cadence Allegro Project Manager tool, switch to the board design flow.
2. On the Flows menu, click **Board Design**.
3. To start the Cadence Allegro Design Entry HDL software, click **Design Entry**.
4. To add the newly created symbol to your schematic, on the Component menu, click **Add**. The **Add Component** dialog box appears.
5. Select the new symbol library location, and select the name of the cell you created from the list of cells.

The symbol attaches to your cursor for placement in the schematic. To fracture the symbol into slots, right-click the symbol and choose **Version** to select one of the slots for placement in the schematic.

Related Links

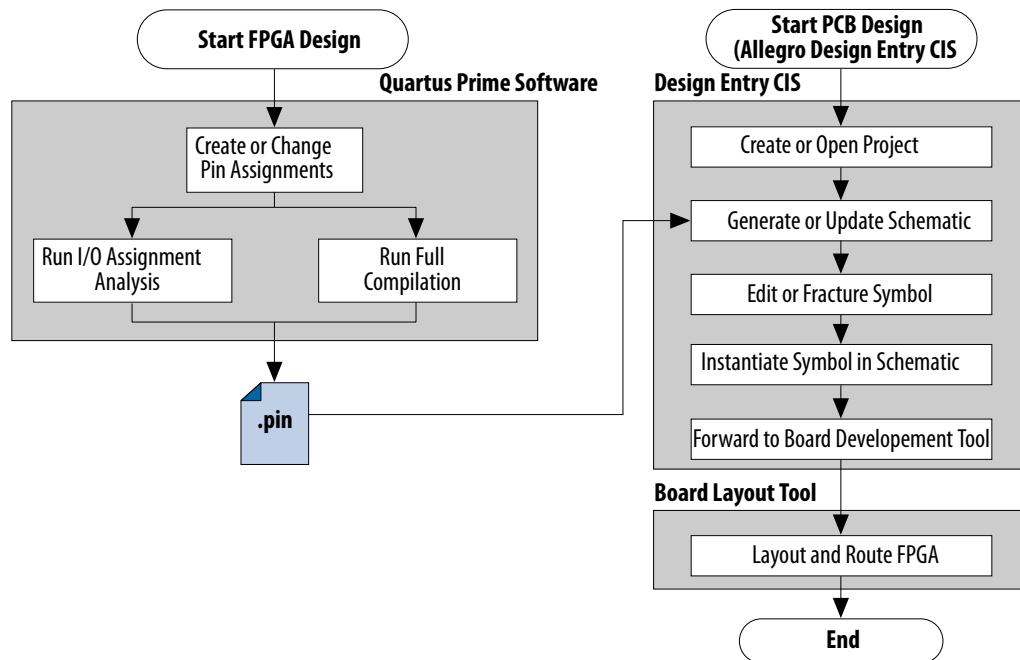
www.cadence.com

Provides more information about the Cadence Allegro Design Entry HDL software, including licensing, support, usage, training, and product updates.

8.6 FPGA-to-Board Integration with Cadence Allegro Design Entry CIS Software

The Cadence Allegro Design Entry CIS software is a schematic capture tool (part of the Cadence 200 series design flow based on OrCAD Capture CIS). Use the Cadence Allegro Design Entry CIS software to create flat circuit schematics for all types of PCB design. You can also create hierarchical schematics to facilitate design reuse and team-based design using the Cadence Allegro Design Entry CIS software. With the Cadence Allegro Design Entry CIS software, the design flow from FPGA-to-board is unidirectional using only the `.pin` generated by the Intel Quartus Prime software. You can only make signal and pin assignment changes in the Intel Quartus Prime software. These changes reflect as updated symbols in a Cadence Allegro Design Entry CIS schematic project.

Figure 53. Design Flow with the Cadence Allegro Design Entry CIS Software



Note: Routing or pin assignment changes made in a board layout tool or a Cadence Allegro Design Entry CIS symbol cannot be back-annotated to the Intel Quartus Prime software.



Related Links

- www.cadence.com
For more information about the Cadence Allegro Design Entry CIS software, including licensing, support, usage, training, and product updates.
- www.ema-eda.com
For more information about the Cadence Allegro Design Entry CIS software, including licensing, support, usage, training, and product updates.

8.6.1 Creating a Cadence Allegro Design Entry CIS Project

The Cadence Allegro Design Entry CIS software has built-in support for creating schematic symbols using pin assignment information imported from the Intel Quartus Prime software.

To create a new project in the Cadence Allegro Design Entry CIS software, follow these steps:

1. On the File menu, point to **New** and click **Project**. The New Project wizard starts.
When you create a new project, you can select the PC Board wizard, the Programmable Logic wizard, or a blank schematic.
2. Select the PC Board wizard to create a project where you can select which part libraries to use, or select a blank schematic.

The Programmable Logic wizard only builds an FPGA logic design in the Cadence Allegro Design Entry CIS software.

Your new project is in the specified location and consists of the following files:

- OrCAD Capture Project File (.opj)
- Schematic Design File (.dsn)

8.6.2 Generating a Part

After you create a new project or open an existing project in the Cadence Allegro Design Entry CIS software, you can generate a new schematic symbol based on your Intel Quartus Prime FPGA design. You can also update an existing symbol. The Cadence Allegro Design Entry CIS software stores component symbols in OrCAD Library File (.olb). When you place a symbol in a library attached to a project, it is immediately available for instantiation in the project schematic.

You can add symbols to an existing library or you can create a new library specifically for the symbols generated from your FPGA designs. To create a new library, follow these steps:

1. On the File menu, point to **New** and click **Library** in the Cadence Allegro Design Entry CIS software to create a default library named **library1.olb**. This library appears in the **Library** folder in the Project Manager window of the Cadence Allegro Design Entry CIS software.
2. To specify a desired name and location for the library, right-click the new library and select **Save As**. Saving the new library creates the library file.

8.6.3 Generating Schematic Symbol

You can now create a new symbol to represent your FPGA design in your schematic.

To generate a schematic symbol, follow these steps:

1. Start the Cadence Allegro Design Entry CIS software.
2. On the Tools menu, click **Generate Part**. The **Generate Part** dialog box appears.
3. To specify the **.pin** from your Intel Quartus Prime design, in the **Netlist/source file type** field, click **Browse**.
4. In the **Netlist/source file type** list, select **Altera Pin File**
5. Type the new part name.
6. Specify the **Destination part library** for the symbol. Failing to select an existing library for the part creates a new library with a default name that matches the name of your Cadence Allegro Design Entry CIS project.
7. To create a new symbol for this design, select **Create new part**. If you updated your **.pin** in the Intel Quartus Prime software and want to transfer any assignment changes to an existing symbol, select **Update pins on existing part in library**.
8. Select any other desired options and set **Implementation type** to **<none>**. The symbol is for a primitive library part based only on the **.pin** and does not require special implementation. Click **OK**.
9. Review the Undo warning and click **Yes** to complete the symbol generation.

You can locate the generated symbol in the selected library or in a new library found in the **Outputs** folder of the design in the Project Manager window. Double-click the name of the new symbol to see its graphical representation and edit it manually using the tools available in the Cadence Allegro Design Entry CIS software.

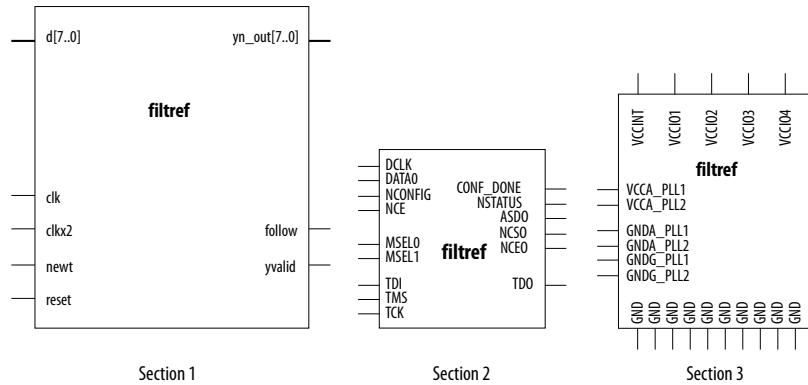
Note: For more information about creating and editing symbols in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

8.6.4 Splitting a Part

After saving a new symbol in a project library, you can fracture the symbol into multiple parts called sections. Fracturing a part into separate sections is useful for FPGA designs. A single symbol for most FPGA packages might be too large for a single schematic page. Splitting the part into separate sections allows you to organize parts of the symbol by function, creating cleaner circuit schematics. For example, you can create one slot for an I/O symbol, a second slot for a JTAG symbol, and a third slot for a power/ground symbol.



Figure 54. Splitting a Symbol into Multiple Sections



- This diagram represents a Cyclone device with JTAG or passive serial (PS) mode configuration option settings. Symbols created for other devices or other configuration modes might have different sets of configuration pins, but can be fractured in a similar manner.
 - The power/ground section shows only a representation of power and ground pins because the device contains a high number of power and ground pins.

Note: Although symbol generation in the Design Entry CIS software refers to symbol fractures as sections, other tools use different names to refer to symbol fractures.

To split a part into sections, select the part in its library in the Project Manager window of the Cadence Allegro Design Entry CIS software. On the Tools menu, click **Split Part** or right-click the part and choose **Split Part**. The **Split Part Section Input Spreadsheet** appears.

Figure 55. Split Part Section Input Spreadsheet

Section Column

Number	Name	Type	Order	Length	User Assig	I/O Bank	Voltage	I/O Standard	Location	Section
1	H1	clk	Input	0	Line	1			Left	1
2	G1	clkx2	Input	1	Line	1			Left	1
3	K13	CONF_DONE	Passive	2	Line	3			Left	2
4	C7	d[0]	Input	3	Line	2			Left	1
5	A6	d[1]	Input	4	Line	2			Left	1
6	D7	d[2]	Input	5	Line	2			Left	1
7	B7	d[3]	Input	6	Line	2			Left	1
8	B8	d[4]	Input	7	Line	2			Left	1
9	M7	d[5]	Input	8	Line	4			Left	1
10	A8	d[6]	Input	9	Line	2			Left	1
11	B6	d[7]	Input	10	Line	2			Left	1
12	H2	DATA0	Input	11	Line	1			Left	2
13	K4	DCLK	Bidirectional	12	Line	1			Left	2
14	C6	follow	Output	13	Line	2			Right	1
15	J3	MSEL0	Passive	14	Line	1			Left	2
16	J2	MSEL1	Passive	15	Line	1			Left	2
17	J4	nCE	Passive	16	Line	1			Left	2
18	H4	nCEO	Passive	17	Line	1			Left	2
19	H3	nCONFIG	Passive	18	Line	1			Left	2
20	H5	newt	Input	19	Line	1			Left	1
21	J13	nSTATUS	Passive	20	Line	3			Left	2
22	G16	reset	Input	21	Line	3			Left	1

Split... Cancel Help

Each row in the spreadsheet represents a pin in the symbol. The **Section** column indicates the section of the symbol to which each pin is assigned. You can locate all pins in a new symbol in section 1. You can change the values in the **Section** column to assign pins to various sections of the symbol. You can also specify the side of a section on the location of the pin by changing the values in the **Location** column. When you are ready, click **Split**. A new symbol appears in the same library as the original with the name <original part name>_Split1.

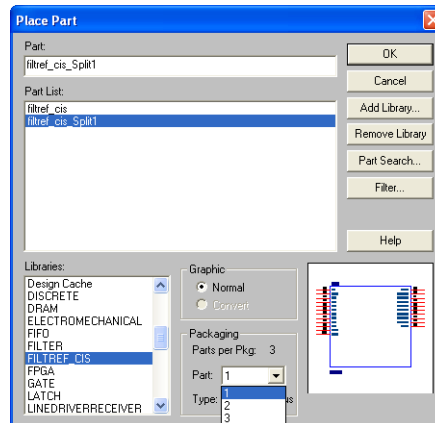
View and edit each section individually. To view the new sections of the part, double-click the part. The Part Symbol Editor window appears and the first section of the part displays for editing. On the View menu, click **Package** to view thumbnails of all the part sections. To edit the section of the symbol, double-click the thumbnail.

For more information about splitting parts into sections and editing symbol sections in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

8.6.5 Instantiating a Symbol in a Design Entry CIS Schematic

After saving a new symbol in a library in your Cadence Allegro Design Entry CIS project, you can instantiate the new symbol on a page in your schematic. Open a schematic page in the Project Manager window of the Cadence Allegro Design Entry CIS software. To add the new symbol to your schematic on the schematic page, on the Place menu, click **Part**. The **Place Part** dialog box appears.

Figure 56. Place Part Dialog Box



Select the new symbol library location and the newly created part name. If you select a part that is split into sections, you can select the section to place from the **Part** menu. Click **OK**. The symbol attaches to your cursor for placement in the schematic. To place the symbol, click the schematic page.

For more information about using the Cadence Allegro Design Entry CIS software, refer to the Help in the software.

8.6.6 Intel Libraries for the Cadence Allegro Design Entry CIS Software

Intel provides downloadable **.olb** for many of its device packages. You can add these libraries to your Cadence Allegro Design Entry CIS project and update the symbols with the pin assignments contained in the **.pin** generated by the Intel Quartus Prime software. You can use the downloaded library symbols as a base for creating custom



schematic symbols with your pin assignments that you can edit or fracture. This method increases productivity by reducing the amount of time it takes to create and edit a new symbol.

8.6.6.1 Using the Intel-provided Libraries with your Cadence Allegro Design Entry CIS Project

To use the Intel-provided libraries with your Cadence Allegro Design Entry CIS project, follow these steps:

1. Download the library of your target device from the Download Center page found through the Support page on the Altera website.
2. Create a copy of the appropriate **.olb** to maintain the original symbols. Place the copy in a convenient location, such as your Cadence Allegro Design Entry CIS project directory.
3. In the Project Manager window of the Cadence Allegro Design Entry CIS software, click once on the **Library** folder to select it. On the Edit menu, click **Project** or right-click the **Library** folder and choose **Add File** to select the copy of the downloaded **.olb** and add it to your project. You can locate the new library in the list of part libraries for your project.
4. On the Tools menu, click **Generate Part**. The **Generate Part** dialog box appears.
5. In the **Netlist/source file** field, click **Browse** to specify the **.pin** in your Intel Quartus Prime design.
6. From the **Netlist/source file type** list, select **Altera Pin File**.
7. For **Part name**, type the name of the target device the same as it appears in the downloaded library file. For example, if you are using a device from the **CYCLONE06.OLB** library, type the part name to match one of the devices in this library such as **ep1c6f256**. You can rename the symbol in the Project Manager window after updating the part.
8. Set the **Destination part library** to the copy of the downloaded library you added to the project.
9. Select **Update pins on existing part in library**. Click **OK**.
10. Click **Yes**.

The symbol is updated with your pin assignments. Double-click the symbol in the Project Manager window to view and edit the symbol. On the View menu, click **Package** if you want to view and edit other sections of the symbol. If the symbol in the downloaded library is fractured into sections, you can edit each section but you cannot further fracture the part. You can generate a new part without using the downloaded part library if you require additional sections.

For more information about creating, editing, and fracturing symbols in the Cadence Allegro Design Entry CIS software, refer to the Help in the software.



8.7 Document Revision History

Table 27. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	<ul style="list-style-type: none">Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
June 2014	14.0.0	Converted to DITA format.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	<ul style="list-style-type: none">General style editing.Removed Referenced Document Section.Added a link to Help in "Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA" on page 9-5.
November 2009	9.1.0	<ul style="list-style-type: none">Added "Performing Simultaneous Switching Noise (SSN) Analysis of Your FPGA" on page 9-5.General style editing.Edited Figure 9-4 on page 9-10 and Figure 9-8 on page 9-16.
March 2009	9.0.0	<ul style="list-style-type: none">Chapter 9 was previously Chapter 7 in the 8.1 software release.No change to content.
November 2008	8.1.0	Changed to 8-1/2 x 11 page size.
May 2008	8.0.0	Updated references.

Related Links

[Documentation Archive](#)

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



9 Reviewing Printed Circuit Board Schematics with the Intel Quartus Prime Software

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

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

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

Related Links

- [Schematic Review Worksheets](#)
- [Pin Connection Guidelines](#)

9.1 Reviewing Intel Quartus Prime Software Settings

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

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

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

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

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



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

Related Links

[Migration Devices Dialog Box](#)
In Intel Quartus Prime Help

9.1.1 Device and Pins Options Dialog Box Settings

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

9.1.1.1 Configuration Settings

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

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

Related Links

[Dual-Purpose Pins Settings](#) on page 178

9.1.1.2 Unused Pin Settings

The **Unused Pin** page specifies the behavior of all unused pins in your design. Use the **Unused Pin** page to ensure that unused pin settings are compatible with your PCB.

For example, if you reserve all unused pins as outputs driving ground, you must ensure that you do not connect unused I/O pins to VCC pins on your PCB. Connecting unused I/O pins to VCC pins may result in contention that could lead to higher than expected current draw and possible device overstress.

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

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

9.1.1.3 Dual-Purpose Pins Settings

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



9.1.1.4 Voltage Settings

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

Ensure that the settings in the **Voltage** page match the settings in your PCB schematic, especially if the target device includes transceivers.

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

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

Related Links

- [Reviewing Device Pin-Out Information in the Fitter Report](#) on page 179
- [Pin Connection Guidelines](#)

9.1.1.5 Error Detection CRC Settings

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

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

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

Related Links

[Device and Pin Options Dialog Box](#)
In Intel Quartus Prime Help

9.2 Reviewing Device Pin-Out Information in the Fitter Report

After you compile your design, you can use the reports in the Resource section of the Fitter report to check your device pin-out in detail.

The Input Pins, Output Pins, and Bidirectional Pins reports identify all the user I/O pins in your design and the features enabled for each I/O pin. For example, you can find use of weak internal pull-ups, PCI clamp diodes, and on-chip termination (OCT) pin assignments in these sections of the Fitter report. You can check the pin assignments reported in the Input Pins, Output Pins, and Bidirectional Pins reports against your PCB schematic to determine whether your PCB requires external components.

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

Figure 57. Resource Section Report

Open the **Compilation Report** tab with **Ctrl+R**, then click **Fitter > Plan Stage Input Pins** (or **Output Pins** or **Bidir Pins**). The following figure shows the pins the Fitter chose for the OCT external calibration resistor connections (RUP/RDN) and the name of the associated termination block in the Input Pins report. You should make these types of assignments user assignments.

Input Pins					
	Name	Pin #	I/O Bank	X coordin...	Y coordin
1	clock_source	AB39	2C	0	59
2	global_reset_n	AB41	2C	0	60
3	termination_blk0~_rdn_pad	C40	1A	0	113
4	termination_blk0~_rup_pad	D40	1A	0	113

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

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

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



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

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

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

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

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

9.3 Reviewing Compilation Error and Warning Messages

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

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

9.4 Using Additional Intel Quartus Prime Software Features

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

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

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

Related Links

- [Signal Integrity Analysis with Third-Party Tools](#) on page 105
- [Managing Device I/O Pins](#) on page 24



9.5 Using Additional Intel Quartus Prime Software Tools

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

You can also use the SSN Analyzer to assist you with reviewing your PCB schematics.

9.5.1 Pin Planner

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

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

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

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

Related Links

[Managing Device I/O Pins](#) on page 24

9.5.2 SSN Analyzer

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

9.6 Document Revision History

Table 28. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
June 2014	14.0.0	Template update.
November 2012	12.1.0	Minor update of Pin Planner description for task and report windows.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Changed to new document template. No change to content.
July 2010	10.0.0	Initial release.



Related Links

[Documentation Archive](#)

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



10 Design Optimization Overview

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

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

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

10.1 Initial Compilation: Required Settings

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

10.1.1 Device Settings

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

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

10.1.2 Device Migration Settings

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

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

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



10.1.3 I/O Assignments

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

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

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

Related Links

- [Timing Closure and Optimization](#) on page 201
- [Managing Device I/O Pins](#) on page 24

10.1.4 Timing Requirement Settings

For best results, use your real time requirements. If you apply more demanding timing requirements than you need, it may result in increased resource usage, higher power utilization, increased compilation time, or all these.

Comprehensive timing requirement settings achieve the best results for the following reasons:

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

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

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

If you already have a .sdc file in your project, the `write_sdc` command from the command line and the **Write SDC File** option from the Timing Analyzer GUI allow you to either create a new .sdc file that combines the constraints from your current .sdc file and any new constraints added through the GUI or command window, or overwrite the existing .sdc file with your newly applied constraints.



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

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

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

Related Links

- [Timing Closure and Optimization](#) on page 201
- [Advanced Settings \(Fitter\)](#)
In *Intel Quartus Prime Help*
- [The Intel Quartus Prime Timing Analyzer](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*
- [Intel Quartus Prime Timing Analyzer Cookbook](#)

10.1.5 Partitions and Floorplan Assignments for Incremental Compilation

The Intel Quartus Prime incremental compilation feature enables hierarchical and team-based design flows in which you can compile parts of your design while other parts of your design remain unchanged. You can also Import parts of your design from separate Intel Quartus Prime projects.

Using incremental compilation for your design with good design partitioning methodology helps to achieve timing closure. Creating design partitions on some of the major blocks in your design and assigning them to Logic Lock (Standard)[™] regions, reduces Fitter time and improves the quality and repeatability of the results. Logic Lock (Standard) regions are flexible, reusable floorplan location constraints that help you place logic on the target device. When you assign entity instances or nodes to a Logic Lock (Standard) region, you direct the Fitter to place those entity instances or nodes inside the region during fitting.

Using incremental compilation helps you achieve timing closure block by block and preserve the timing performance between iterations, which aid in achieving timing closure for the entire design. Incremental compilation may also help reduce compilation times.

Note: If you plan to use incremental compilation, you must create a floorplan for your design. If you are not using incremental compilation, creating a floorplan is optional.



Related Links

- [Reducing Compilation Time](#) on page 192
- [Best Practices for Incremental Compilation Partitions and Floorplan Assignments](#)

10.2 Physical Implementation

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

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

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

Related Links

[Design Planning with the Intel Quartus Prime Software](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

10.2.1 Trade-Offs and Limitations

Many optimization goals can conflict with one another, so you might need to resolve conflicting goals.

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

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

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

Related Links

[Timing Closure and Optimization](#) on page 201

10.2.2 Preserving Results and Enabling Teamwork

For some Intel Quartus Prime Fitter algorithms, small changes to the design can have a large impact on the final result. For example, a critical path delay can change by 10% or more because of seemingly insignificant changes. If you are close to meeting your timing objectives, you can use the Fitter algorithm to your advantage by changing the fitter seed, which changes the pseudo-random result of the Fitter.

Conversely, if you cannot meet timing on a portion of your design, you can partition that portion and prevent it from recompiling if an unrelated part of the design is changed. This feature, known as incremental compilation, can reduce the Fitter runtimes by up to 70% if the design is partitioned, such that only small portions require recompilation at any one time.

When you use incremental compilation, you can apply design optimization options to individual design partitions and preserve performance in other partitions by leaving them untouched. Many optimization techniques often result in longer compilation times, but by applying them only on specific partitions, you can reduce this impact and complete iterations more quickly.

In addition, by physically floorplanning your partitions with Logic Lock (Standard) regions, you can enable team-based flows and allow multiple people to work on different portions of the design.

Related Links

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Designs](#)

10.2.3 Reducing Area

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

Related Links

- [Netlist Optimizations and Physical Synthesis](#) on page 306
- [Reducing Area](#) on page 188
- [Recommended HDL Coding Styles](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

10.2.4 Reducing Critical Path Delay

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

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

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



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

If you use incremental compilation, you can help resolve inter-partition timing requirements by locking down results, one partition at a time, or by guiding the placement of the partitions with Logic Lock (Standard) regions. You might be able to improve the timing on such paths by placing the partitions optimally to reduce the length of critical paths. Once your inter-partition timing requirements are met, use incremental compilation to preserve the results and work on partitions that have not met timing requirements.

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

Related Links

[Critical Paths](#) on page 202

10.2.5 Reducing Power Consumption

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

Related Links

[Power Optimization](#) on page 247

10.2.6 Reducing Runtime

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

The Intel Quartus Prime software supports parallel compilation in computers with multiple processors. This can reduce compilation times by up to 15%.

You can also reduce compilation time with your iterations by using incremental compilation. Use incremental compilation when you want to change parts of your design, while keeping most of the remaining logic unchanged.

10.3 Using Intel Quartus Prime Tools

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

10.3.1 Design Analysis

The Intel Quartus Prime software provides tools that help with a visual representation of your design.

- You can use the RTL Viewer to see a schematic representation of your design before synthesis and place-and-route.
- The Design Partition Planner displays designs at partition and entity levels, and can display connectivity between entities.
- The Technology Map Viewer provides a schematic representation of the design implementation in the selected device architecture after synthesis and place-and-route. Optionally, the can also include timing information.
- With the Chip Planner, you can make floorplan assignments, implement engineering change orders (ECOs), perform power analysis, and visualize critical paths and routing congestion.
- With incremental compilation, the Design Partition Planner and the Chip Planner allow you to partition and layout your design at a higher level.

Related Links

- [Design Floorplan Analysis in the Chip Planner](#) on page 280
- [Optimizing the Design Netlist](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

10.3.2 Advisors

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

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

- Resource Optimization Advisor
- Timing Optimization Advisor
- Power Optimization Advisor
- Compilation Time Advisor
- Pin Optimization Advisor
- Incremental Compilation Advisor

Related Links

- [Compilation Time Advisor](#) on page 192
- [Timing Optimization Advisor](#) on page 222
- [Power Optimization Advisor](#) on page 264

10.3.3 Design Space Explorer II

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



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

Related Links

[Launch Design Space Explorer Command \(Tools Menu\)](#)
In *Intel Quartus Prime Help*

10.4 Document Revision History

Table 29. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Minor changes for HardCopy.
May 2013	13.0.0	Added the information about initial compilation requirements. This section was moved from the Area Optimization chapter of the Intel Quartus Prime Handbook. Minor updates to delineate division of Timing and Area optimization chapters.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.3	Template update.
December 2010	10.0.2	Changed to new document template. No change to content.
August 2010	10.0.1	Corrected link
July 2010	10.0.0	Initial release. Chapter based on topics and text in Section III of volume 2.

Related Links

[Documentation Archive](#)

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

11 Reducing Compilation Time

You can employ various techniques to reduce the time required for synthesis and fitting in the Intel Quartus Prime Compiler.

11.1 Compilation Time Advisor

A Compilation Time Advisor is available in the Intel Quartus Prime GUI by clicking **Tools > Advisors > Compilation Time Advisor**. This chapter describes all the compilation time optimizing techniques available in the Compilation Time Advisor.

11.2 Strategies to Reduce the Overall Compilation Time

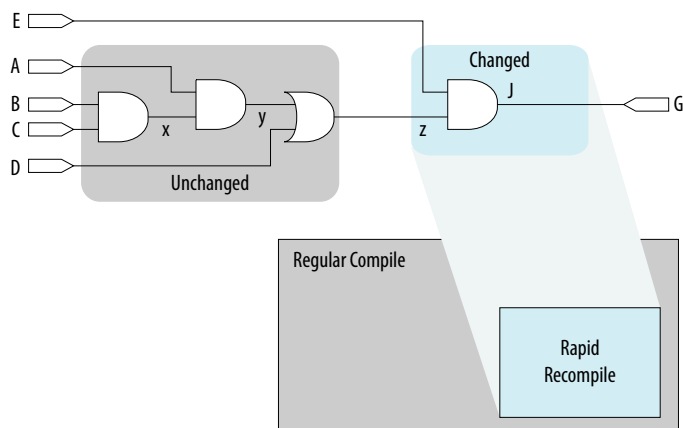
You can use the following strategies to reduce the overall time required to compile your design:

- Parallel compilation (for systems with multiple processor cores)
- Incremental compilation reduces compilation time by only recompiling design partitions that have not met design requirements.
- Rapid Recompile and Smart Compilation reuse results from a previous compilation to reduce overall compilation time

11.2.1 Running Rapid Recompile

During Rapid Recompile the Compiler reuses previous synthesis and fitting results whenever possible, and does not reprocess unchanged design blocks. Use Rapid Recompile to reduce timing variations and the total recompilation time after making small design changes.

Figure 58. Rapid Recompile



Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.



To run Rapid Recompile, follow these steps:

1. To start Rapid Recompile following an initial compilation (or after running the Route stage of the Fitter), click **Processing > Start > Start Rapid Recompile**. Rapid Recompile implements the following types of design changes without full recompilation:
 - Changes to nodes tapped by the Signal Tap Logic Analyzer
 - Changes to combinational logic functions
 - Changes to state machine logic (for example, new states, state transition changes)
 - Changes to signal or bus latency or addition of pipeline registers
 - Changes to coefficients of an adder or multiplier
 - Changes register packing behavior of DSP, RAM, or I/O
 - Removal of unnecessary logic
 - Changes to synthesis directives

The Incremental Compilation Preservation Summary report provides details about placement and routing implementation.

2. Click the Rapid Recompile Preservation Summary report to view detailed information about the percentage of preserved compilation results.

Figure 59. Rapid Recompile Preservation Summary

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

11.2.2 Enabling Multi-Processor Compilation

The Compiler can detect and use multiple processors to reduce total compilation time. You specify the number of processors the Compiler uses. The Intel Quartus Prime software can use up to 16 processors to run algorithms in parallel. The Compiler uses parallel compilation by default. To reserve some processors for other tasks, specify a maximum number of processors that the software uses.

This technique reduces the compilation time by up to 10% on systems with two processing cores, and by up to 20% on systems with four cores. When running timing analysis independently, two processors reduce the timing analysis time by an average of 10%. This reduction reaches an average of 15% when using four processors.

The Intel Quartus Prime software does not necessarily use all the processors that you specify during a given compilation. Additionally, the software never uses more than the specified number of processors. This fact enables you to work on other tasks without slowing down your computer. The use of multiple processors does not affect the quality of the fit. For a given Fitter seed, and given **Maximum processors allowed** setting on a specific design, the fit is exactly the same and deterministic. This remains true, regardless of the target machine, and the number of available processors. Different **Maximum processors allowed** specifications produce different results of the same quality. The impact is similar to changing the Fitter seed setting.

To enable multiprocessor compilation, follow these steps:

1. Open or create an Intel Quartus Prime project.
2. To enable multiprocessor compilation, click **Assignments** > **Settings** > **Compilation Process Settings**.
3. Under **Parallel compilation**, specify options for the number of processors the Compiler uses.
4. View detailed information about processor in the Parallel Compilation report following compilation.

To specify the number of processors for compilation at the command line, use the following Tcl command in your script:

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

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

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

```
set_global_assignment -name NUM_PARALLEL_PROCESSORS ALL
```

The actual reduction in compilation time when using incremental compilation partitions depends on your design and on the specific compilation settings. For example, compilations with multi-corner optimization enabled benefit more from using multiple processors than compilations without multi-corner optimization. The Fitter (`quartus_fit`) and the Intel Quartus Prime Timing Analyzer (`quartus_sta`) stages in the compilation can, in certain cases, benefit from the use of multiple processors. The Flow Elapsed Time report shows the average number of processors for these stages. The Parallel Compilation report shows a more detailed breakdown of processor usage. This report displays only if you enable parallel compilation.

For designs with partitions, once you partition your design and enable partial compilation, the Intel Quartus Prime software can use different processors to compile those partitions simultaneously during Analysis & Synthesis. This can cause higher peak memory usage during Analysis & Synthesis.

Note: The Compiler detects Intel Hyper-Threading as a single processor. If your system includes a single processor with Intel Hyper-Threading, set the number of processors to one. Do not use the Intel Hyper-Threading feature for Intel Quartus Prime compilations.

11.2.3 Using Incremental Compilation

The incremental compilation feature can accelerate design iteration time by up to 70% for small design changes, and helps you reach design timing closure more efficiently.

You can speed up design iterations by recompiling only a particular design partition and merging results with previous compilation results from other partitions. You can also use physical synthesis optimization techniques for specific design partitions while leaving other parts of your design untouched to preserve performance.

If you are using a third-party synthesis tool, you can create separate atom netlist files for the parts of your design that you already have synthesized and optimized so that you update only the parts of your design that change.



In the standard incremental compilation design flow, you can divide the top-level design into partitions, which the software can compile and optimize in the top-level Intel Quartus Prime project. You can preserve fitting results and performance for completed partitions while other parts of your design are changing. Incremental compilation reduces the compilation time for each design iteration because the software does not recompile the unchanged partitions in your design.

The incremental compilation feature facilitates team-based design flows by enabling designers to create and optimize design blocks independently, when necessary, and supports third-party IP integration.

Related Links

[Incremental Compilation for Hierarchical and Team-Based Design](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

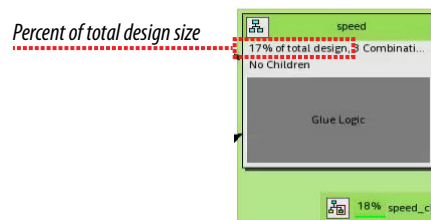
11.2.4 Using Block-Based Compilation

During the design process, you can isolate functional blocks that meet placement and timing requirements from others still undergoing change and optimization. By isolating functional blocks into partitions, you can apply optimization techniques to selected areas and recompile only those areas.

To create partitions dividing functional blocks:

1. In the Design Partition Planner, identify blocks of a size suitable for partitioning.
In general, a partition represents roughly 15 to 20 percent of the total design size. Use the information area below the bar at the top of each entity.

Figure 60. Entity representation in the Design Partition Planner



2. Extract and collapse entities as necessary to achieve stand-alone blocks
3. For each entity of the desired size containing related blocks of logic, right-click the entity and click **Create Design Partition** to place that entity in its own partition.
The goal is to achieve partitions containing related blocks of logic.

11.3 Reducing Synthesis Time and Synthesis Netlist Optimization Time

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

11.3.1 Settings to Reduce Synthesis Time and Synthesis Netlist Optimization Time

Synthesis netlist and physical synthesis optimization settings can significantly increase the overall compilation time for large designs. Refer to Analysis and Synthesis messages to determine the length of optimization time.

If your design already meets performance requirements without synthesis netlist or physical synthesis optimizations, turn off these options to reduce compilation time. If you require synthesis netlist optimizations to meet performance, optimize partitions of your design hierarchy separately to reduce the overall time spent in Analysis and Synthesis.

11.3.2 Use Appropriate Coding Style to Reduce Synthesis Time

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

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

Related Links

[Recommended HDL Coding Styles](#)

In Intel Quartus Prime Standard Edition Handbook Volume 1

11.4 Reducing Placement Time

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

You can reduce the placement time by changing the settings for the placement algorithm, or by using incremental compilation to preserve the placement for the unchanged parts of your design.

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

11.4.1 Fitter Effort Setting

For designs with very tight timing requirements, both **Auto Fit** and **Standard Fit** use the maximum effort during optimization. Intel recommends using **Auto Fit** for reducing compilation time.

The highest Fitter effort setting, **Standard Fit**, requires the most runtime, but does not always yield a better result than using the default **Auto Fit**. If you are certain that your design has only easy-to-meet timing constraints, you can select **Fast Fit** for an even greater runtime savings.



11.4.2 Placement Effort Multiplier Settings

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

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

11.4.3 Physical Synthesis Effort Settings

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

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

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

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

11.4.4 Preserving Placement with Incremental Compilation

Preserving information about previous placements can make future placements faster. The incremental compilation feature provides an easy-to-use method for preserving placement results.

Related Links

[Using Incremental Compilation](#) on page 194

11.5 Reducing Routing Time

The routing time is usually not a significant amount of the compilation time. The time required to route a design depends on three factors: the device architecture, the placement of your design in the device, and the connectivity between different parts of your design.



If your design requires a long time to route, perform one or more of the following actions:

- Check for routing congestion.
- Turn off **Fitter Aggressive Routability Optimization**.
- Use incremental compilation to preserve routing information for parts of your design.

11.5.1 Identifying Routing Congestion with the Chip Planner

To identify areas of routing congestion in your design:

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

The Intel Quartus Prime compilation messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60% indicate possible difficulties fitting your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, indicate a high chance of not getting a valid fit.

Related Links

[Using Incremental Compilation](#) on page 194

11.5.1.1 Areas with Routing Congestion

Even if average congestion is not high, the design may have areas where congestion is high in a specific type of routing. You can use the Chip Planner to identify areas of high congestion for specific interconnect types.

- You can change the connections in your design to reduce routing congestion
- If the area with routing congestion is in a Logic Lock (Standard) region or between Logic Lock (Standard) regions, change or remove the Logic Lock (Standard) regions and recompile your design.
 - If the routing time remains the same, the time is a characteristic of your design and the placement
 - If the routing time decreases, consider changing the size, location, or contents of Logic Lock (Standard) regions to reduce congestion and decrease routing time.

Related Links

- [Analyzing and Optimizing the Design Floorplan](#) on page 280
- [Using Incremental Compilation](#) on page 194



11.5.1.2 Congestion due to HDL Coding style

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

Related Links

[Recommended HDL Coding Styles](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 1*

11.5.1.3 Preserving Routing with Incremental Compilation

Preserving the previous routing results for part of your design can reduce future routing time. Incremental compilation provides an easy-to-use methodology that preserves placement and routing results.

11.6 Reducing Static Timing Analysis Time

If you are performing timing-driven synthesis, the Intel Quartus Prime software runs the Timing Analyzer during Analysis and Synthesis.

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

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

11.7 Setting Process Priority

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

Related Links

[Processing Page \(Options Dialog Box\)](#)

In Intel Quartus Prime Help.



11.8 Document Revision History

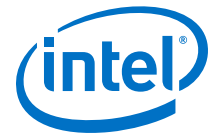
Table 30. Document Revision History

Date	Version	Changes
2016.05.02	16.0.0	<ul style="list-style-type: none">Corrected typo in Using Parallel Compilation with Multiple Processors.Stated limitations about deprecated physical synthesis options.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
2014.12.15	14.1.0	<ul style="list-style-type: none">Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.Added information about Rapid Recompile feature.
2014.08.18	14.0a10.0	Added restriction about smart compilation in Arria 10 devices.
June 2014	14.0.0	Updated format.
May 2013	13.0.0	Removed the "Limit to One Fitting Attempt", "Using Early Timing Estimation", "Final Placement Optimizations", and "Using Rapid Recompile" sections. Updated "Placement Effort Multiplier Settings" section. Updated "Identifying Routing Congestion in the Chip Planner" section. General editorial changes throughout the chapter.
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	<ul style="list-style-type: none">Updated "Using Parallel Compilation with Multiple Processors".Updated "Identifying Routing Congestion in the Chip Planner".General editorial changes throughout the chapter.
December 2010	10.1.0	<ul style="list-style-type: none">Template update.Added details about peak and average interconnect usage.Added new section "Reducing Static Timing Analysis Time".Minor changes throughout chapter.
July 2010	10.0.0	Initial release.

Related Links

Documentation Archive

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



12 Timing Closure and Optimization

12.1 About Timing Closure and Optimization

This chapter describes techniques to improve timing performance when designing for Intel devices.

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

12.2 Optimize Multi-Corner Timing

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

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

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

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

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

12.3 Critical Paths

Critical paths are timing paths in your design that have a negative slack. These timing paths can span from device I/Os to internal registers, registers to registers, or from registers to device I/Os.

The slack of a path determines its criticality; slack appears in the timing analysis report, which you can generate using the Timing Analyzer.

Design analysis for timing closure is a fundamental requirement for optimal performance in highly complex designs. The analytical capability of the Chip Planner helps you close timing on complex designs.

Related Links

- [Reducing Critical Path Delay](#) on page 188
- [Displaying Path Reports with the Timing Analyzer](#) on page 217

12.3.1 Viewing Critical Paths

Viewing critical paths in the Chip Planner shows why a specific path is failing. You can see if any modification in the placement can reduce the negative slack. To display paths in the floorplan, perform a timing analysis and display results on the Timing Analyzer.

12.4 Design Evaluation for Timing Closure

Follow the guidelines in this section when you encounter timing failures in a design. The guidelines show you how to evaluate compilation results of a design and how to address problems. While the guideline does not cover specific examples of restructuring RTL to improve design speed, the analysis techniques help you to evaluate changes to RTL that can help you to close timing.

12.4.1 Review Compilation Results

12.4.1.1 Review Messages

After compiling your design, review the messages in each section of the compilation report.

Most designs that fail timing start out with other problems that the Fitter reports as warning messages during compilation. Determine what causes a warning message, and whether to fix or ignore the warning.

After reviewing the warning messages, review the informational messages. Take note of anything unexpected, for example, unconnected ports, ignored constraints, missing files, and assumptions or optimizations that the software made.

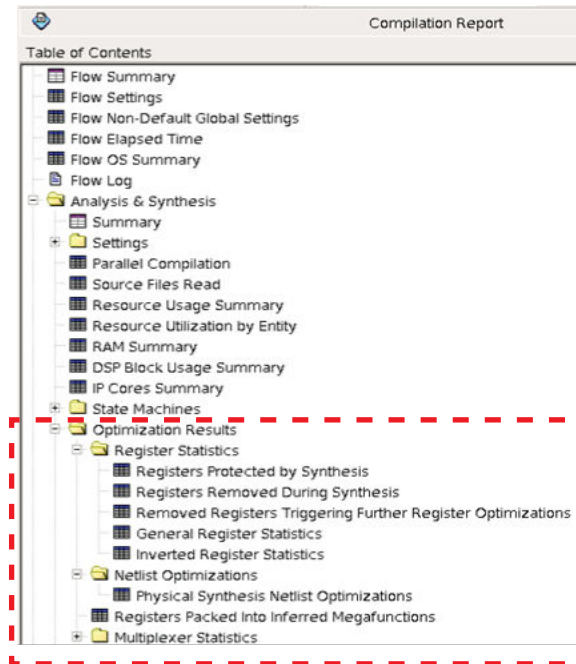
12.4.1.2 Evaluate Physical Synthesis Results

If you enable physical synthesis options, the Compiler can duplicate and retime registers, and modify combinatorial logic during synthesis. After compilation, review the Optimization Results reports in the Analysis & Synthesis section. The reports list



the optimizations performed by the physical synthesis optimizations, such as register duplication, retiming, and removal. These reports can be found in the Compilation Report panel.

Figure 61. Optimization Results Reports



When you enable physical synthesis, the compilation messages include a information about the physical synthesis algorithm performance improvement. The reported improvement is the sum of the largest timing-critical clock domain. Although typically similar, the values for the slack improvements vary per compilation due to the random starting point of compilation algorithms.

12.4.1.3 Evaluate Fitter Netlist Optimizations

The Fitter can also perform optimizations to the design netlist. Major changes include register packing, duplicating or deleting logic cells, inverting signals, or modifying nodes in a general way such as moving an input from one logic cell to another. Find and review these reports in the Netlist Optimizations results of the Fitter section.

12.4.1.4 Evaluate Optimization Results

After checking what optimizations were done and how they improved performance, evaluate the runtime it took to get the extra performance. To reduce compilation time, review the physical synthesis and netlist optimizations over a couple of compilations, and edit the RTL to reflect the changes that physical synthesis performed. If a particular set of registers consistently get retimed, edit the RTL to retime the registers the same way. If the changes are made to match what the physical synthesis algorithms did, the physical synthesis options can be turned off to save compile time while getting the same type of performance improvement.

12.4.1.5 Evaluate Resource Usage

Evaluate a variety of resources used in the design, including global and non-global signal usage, routing utilization, and clustering difficulty.

12.4.1.5.1 Global and Non-global Usage

If your design contains a lot of clocks, evaluate global and non-global signals. Determine whether global resources are used effectively, and if not, consider making changes. You can find these reports in the Resource section under Fitter in the **Compilation Report** panel.

The figure shows an example of inefficient use of a global clock. The highlighted line has a single fan-out from a global clock.

Figure 62. Inefficient Use of a Global Clock

Global & Other Fast Signals			
Location	Fan-Out	Global Resource Used	Global Line Name
FRACTIONALPLL_X98_Y2_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y2_N1	29044	Global Clock	GCLK7
PLLOUTPUTCOUNTER_X98_Y13_N1	253103	Global Clock	GCLK6
FF_X185_Y66_N13	280349	Global Clock	GCLK8
PIN_AE17	4887	Global Clock	GCLK4
FRACTIONALPLL_X98_Y11_N0	1	Global Clock	--
PLLOUTPUTCOUNTER_X98_Y3_N1	1	Global Clock	GCLK5
PLLOUTPUTCOUNTER_X98_Y1_N1	1691	Regional Clock	RCLK29
PLLOUTPUTCOUNTER_X98_Y8_N1	302	Regional Clock	RCLK23
PLLOUTPUTCOUNTER_X98_Y11_N1	141	Regional Clock	RCLK25
PLLOUTPUTCOUNTER_X98_Y10_N1	17	Regional Clock	RCLK22

If you assign it to a Regional Clock, the Global Clock becomes available for another signal. You can ignore signals with an empty value in the **Global Line Name** column as the signal uses dedicated routing, and not a clock buffer.

The Non-Global High Fan-Out Signals report lists the highest fan-out nodes not routed on global signals.

Reset and enable signals appear at the top of the list.

If there is routing congestion in the design, and there are high fan-out non-global nodes in the congested area, consider using global or regional signals to fan-out the nodes, or duplicate the high fan-out registers so that each of the duplicates can have fewer fan-outs.

Use the Chip Planner to locate high fan-out nodes, to report routing congestion, and to determine whether the alternatives are viable.

12.4.1.5.2 Routing Usage

Review routing usage reported in the **Fitter Resource Usage Summary** report. The figure shows an example of the report.

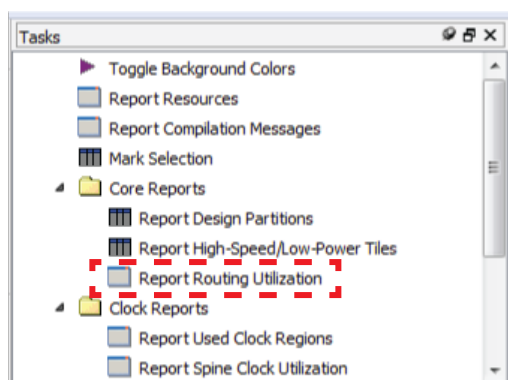


Figure 63. Fitter Resource Usage Summary Report

Resource	Usage
10G TX PCSs	12 / 36 (33 %)
HSSI PMA TX Serializers	12 / 36 (33 %)
CHANNEL PLLs	12 / 36 (33 %)
Impedance control blocks	1 / 4 (25 %)
Average interconnect usage (total/H/V)	55% / 55% / 55%
Peak interconnect usage (total/H/V)	88% / 88% / 90%

The average interconnect usage reports the average amount of interconnect that is used, out of what is available on the device. The peak interconnect usage reports the largest amount of interconnect used in the most congested areas. Designs with an average value below 50% typically do not have any problems with routing. Designs with an average between 50-65% may have difficulty routing. Designs with an average over 65% typically have difficulty meeting timing unless the RTL is well designed to tolerate a highly utilized chip. Peak values at or above 90% are likely to have problems with timing closure; a 100% peak value indicates that all routing in an area of the device has been used, so there is a high possibility of degradation in timing performance. The figure shows the **Report Routing Utilization** report.

Figure 64. Report Routing Utilization Report

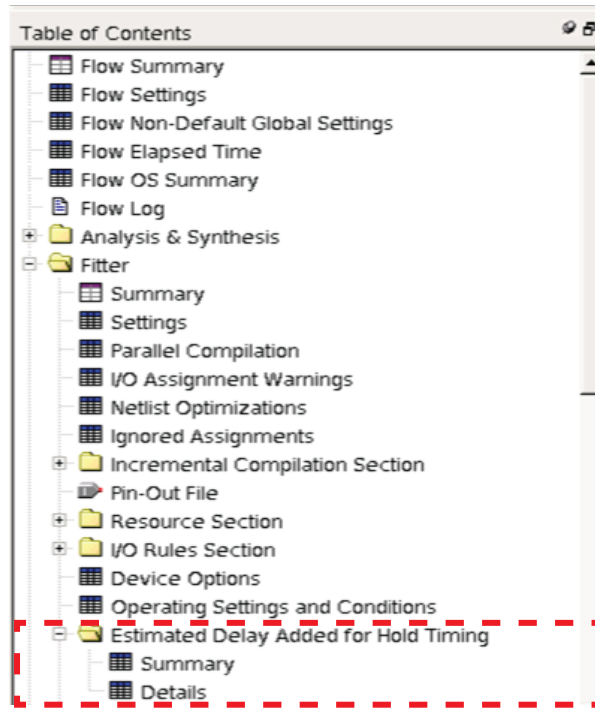


12.4.1.5.3 Wires Added for Hold

As part of the fitting process, the router can add wire between register paths to increase delay to meet hold time requirements. During the routing process, the router reports how much extra wire was used to meet hold time requirements. Excessive added wire can indicate problems with the constraint. Typically this situation is caused by incorrect multicycle transfers, particularly between different rate clocks, and between different clock networks.

The Fitter reports how much routing delay was added in the **Estimated Delay Added for Hold Timing** report. You can review specific register paths to check whether a delay was added to meet hold requirements.

Figure 65. Estimated Delay Added for Hold Timing Report

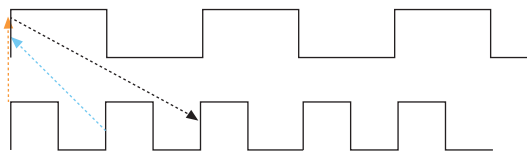


An example of an incorrect constraint which can cause the router to add wire for hold requirements is when there is data transfer from 1x to 2x clocks. Assume the design intent is to allow two cycles per transfer. Data can arrive any time in the two destination clock cycles by adding a multicycle setup constraint as shown in the example:

```
set_multicycle_path -from 1x -to 2x -setup -end 2
```

The timing requirement is relaxed by one 2x clock cycle, as shown in the black line in the waveform in the figure.

Figure 66. Timing Requirement Relaxed Waveform



The default hold requirement, shown with the dashed blue line, can force the router to add wire to guarantee that data is delayed by one cycle. To correct the hold requirement, add a multicycle constraint with a hold option.

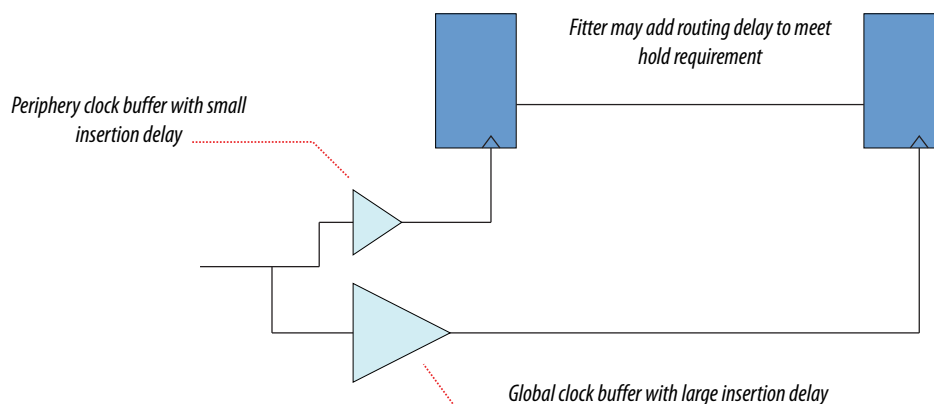
```
set_multicycle_path -from 1x -to 2x -setup -end 2
set_multicycle_path -from 1x -to 2x -hold -end 1
```

The orange dashed line in the figure above represents the hold relationship, and no extra wire is required to delay the data.



The router can also add wire for hold timing requirements when data is transferred in the same clock domain, but between clock branches that use different buffering. Transferring between clock network types happens more often between the periphery and the core. The figure below shows a case where data is coming into a device, and uses a periphery clock to drive the source register, and a global clock to drive the destination register. A global clock buffer has larger insertion delay than a periphery clock buffer. The clock delay to the destination register is much larger than to the source register, hence extra delay is necessary on the data path to ensure that it meets its hold requirement.

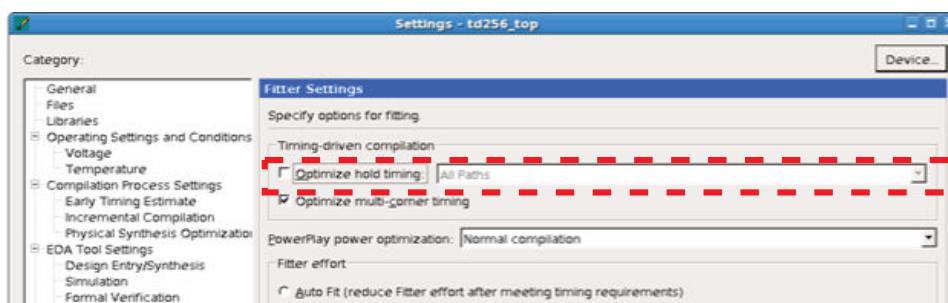
Figure 67. Clock Delay



To identify cases where a path has different clock network types, review the path in the Timing Analyzer, and check nodes along the source and destination clock paths. Also, check the source and destination clock frequencies to see whether they are the same, or multiples, and whether there are multicycle exceptions on the paths. Finally, ensure that all cross-domain paths that are false by intent have an associated false path exception.

If you suspect that routing is added to fix real hold problems, then disable the **Optimize hold timing** option. Recompile the design and rerun timing analysis to uncover paths that fail hold time.

Figure 68. Optimize Hold Timing Option



Note: Disable the **Optimize hold timing** option only when debugging your design. Ensure to enable the option (default state) during normal compiles. Wire added for hold is a normal part of timing optimization during routing and is not always a problem.



12.4.1.6 Evaluate Other Reports and Adjust Settings Accordingly

12.4.1.6.1 Difficulty Packing Design

In the Fitter Resource Section, under the **Resource Usage Summary**, review the **Difficulty Packing Design** report. The **Difficulty Packing Design** report details the effort level (low, medium, or high) of the Fitter to fit the design into the device, partition, and Logic Lock (Standard) region.

As the effort level of **Difficulty Packing Design** increases, timing closure gets harder. Going from medium to high can result in significant drop in performance or increase in compile time. Consider reducing logic to reduce packing difficulty.

12.4.1.6.2 Review Ignored Assignments

The **Compilation Report** includes details of any assignments ignored by the Fitter. Assignments typically get ignored if design names change, but assignments are not updated. Make sure any intended assignments are not being ignored.

12.4.1.6.3 Review Non-Default Settings

The reports from Synthesis and Fitter show non-default settings used in a compilation. Review the non-default settings to ensure the design benefits from the change.

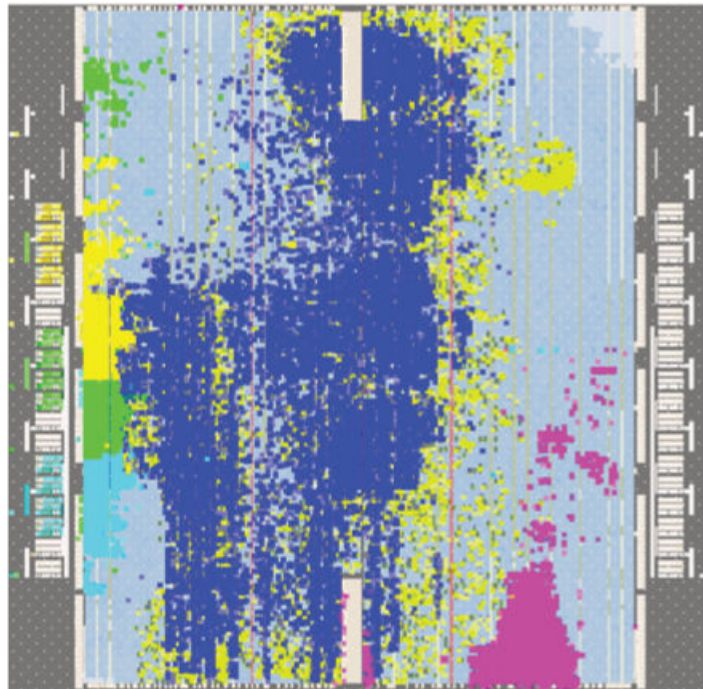
12.4.1.6.4 Review Floorplan

Use the Chip Planner for reviewing placement.

You can use the Chip Planner to locate hierarchical entities, using colors for each located entity in the floorplan. Look for logic that seems out of place, based on where you expect it to be.

For example, logic that interfaces with I/Os should be close to the I/Os, and logic that interfaces with an IP or memory should be close to the IP or memory.

Figure 69. Floorplan with Color-Coded Entities



- The figure shows a floorplan with color-coded entities. In the floorplan, the green block is spread apart. Check to see if those paths are failing timing, and if so, what connects to that module that could affect placement.
- The blue and aqua blocks are spread out and mixed together. Check if connections between the two modules contribute to this.
- The pink logic at the bottom must interface with I/Os at the bottom edge. Check fan-in and fan-out of a highlighted module by using the buttons on the task bar.

Figure 70. Fan-in and Fan-Out Buttons



Look for signals that go a long way across the chip and see if they are contributing to timing failures.

- Check global signal usage for signals that affect logic placement, and verify if the Fitter placed logic feeding a global buffer close to the buffer and away from related logic. Use settings like high fan-out on non-global resource to pull logic together.
- Check for routing congestion. The Fitter spreads out logic in highly congested areas, making the design harder to route.

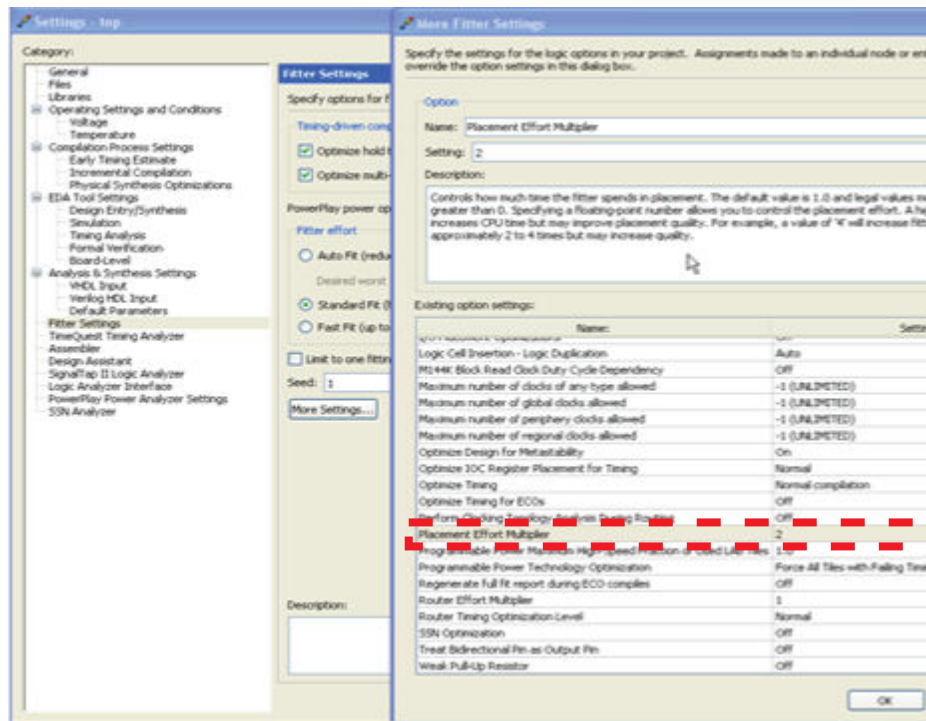
12.4.1.6.5 Evaluate Placement and Routing

Review duration of parts of compile time in Fitter messages. If routing takes much more time than placement, then meeting timing may be more difficult than the placer predicted.

12.4.1.6.6 Adjust Placement Effort

The benefit of increasing the **Placement Effort Multiplier** to improve placement quality at the cost of higher compile time is design dependent. Adjust the multiplier after reviewing and optimizing other settings and RTL. Try an increased value, up to 4, and reset to default if performance or compile time does not improve.

Figure 71. Placement Effort Multiplier



12.4.1.6.7 Adjust Fitter Effort

Fitter settings allow you to adjust high-level Compiler optimization settings.

To increase effort, enable the **Standard Fit (highest effort)** option. The default **Auto Fit** option reduces Fitter effort when it estimates timing requirements are met.

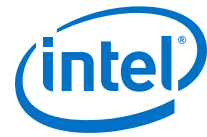
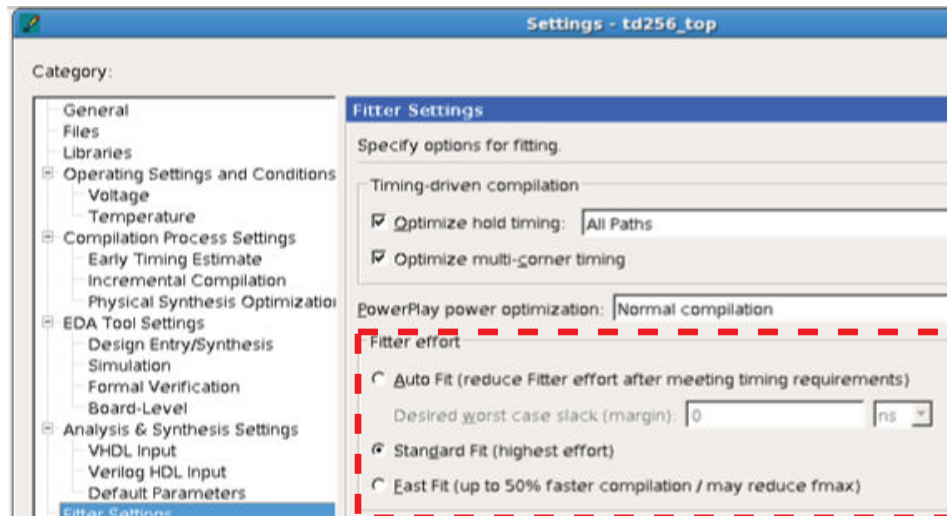


Figure 72. Fitter Effort



12.4.1.6.8 Review Timing Constraints

Ensure that clocks are constrained with the correct frequency requirements. Using the `derive_pll_clocks` assignment keeps generated clock settings updated. Timing Analyzer can be useful in reviewing SDC constraints. For example, under **Diagnostic** in the Task panel, the **Report Ignored Constraints** report shows any incorrect names in the design, most commonly caused by changes in the design hierarchy. Use the **Report Unconstrained Paths** report to locate unconstrained paths. Add constraints as necessary so that the design can be optimized.

12.4.1.7 Evaluate Clustering Difficulty

You can evaluate clustering difficulty to help reach timing closure.

You can monitor clustering difficulty whenever you add logic and recompile. Use the clustering information to gauge how much timing closure difficulty is inherent in your design:

- If your design is full but clustering difficulty is low or medium, your design itself, rather than clustering, is likely the main cause of congestion.
- Conversely, congestion occurring after adding a small amount of logic to the design, can be due to clustering. If clustering difficulty is high, this contributes to congestion regardless of design size.

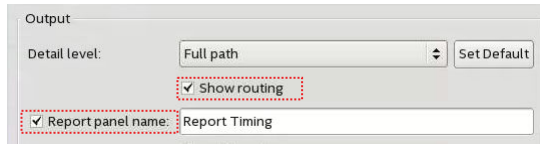
12.4.2 Review Details of Timing Paths

12.4.2.1 Show Timing Path Routing

Showing routing for a path can help uncover unusual routing delays.

In the Timing Analyzer **Report Timing** dialog box, enable the **Report panel name** and **Show routing** options, and click **Report Timing**.

Figure 73. Report Pane and Show Routing Options



The **Extra Fitter Information** tab shows a miniature floorplan with the path highlighted.

You can also locate the path in the Chip Planner to examine routing congestion, and to view whether nodes in a path are placed close together or far apart.

Related Links

[Exploring Paths in the Chip Planner](#) on page 286

12.4.2.2 Global Network Buffers

You can use routing paths to identify global network buffers that fail timing. Buffer locations are named according to the network they drive.

- CLK_CTRL_Gn—for Global driver
- CLk_CTRL_Rn—for Regional driver

Buffers to access the global networks are located in the center of each side of the device. Buffering to route a core logic signal on a global signal network causes insertion delay. Trade offs to consider for global and non-global routing are source location, insertion delay, fan-out, distance a signal travels, and possible congestion if the signal is demoted to local routing.

12.4.2.2.1 Source Location

If the register feeding the global buffer cannot be moved closer, then consider changing either the design logic or the routing type.

12.4.2.2.2 Insertion Delay

If a global signal is required, consider adding half a cycle to timing by using a negative-edge triggered register to generate the signal (top figure) and use a multicycle setup constraint (bottom figure).

Figure 74. Negative-Edge Triggered Register

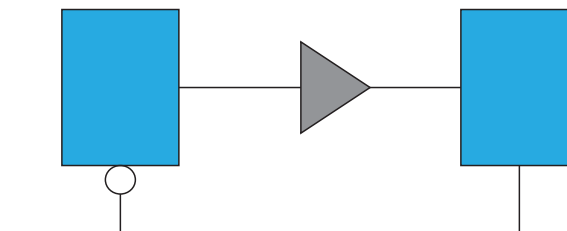
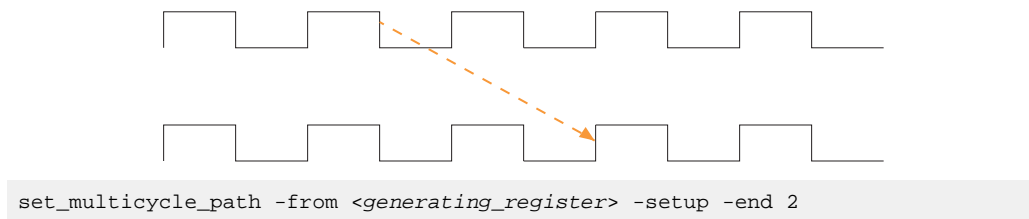




Figure 75. Multicycle Setup Constraint



12.4.2.2.3 Fan-Out

Nodes with very high fan-out that use local routing tend to pull logic that they drive close to the source node. This can make other paths fail timing. Duplicating registers can help reduce the impact of high fan-out paths. Consider manually duplicating and preserving these registers. Using a `MAX_FANOUT` assignment may make arbitrary groups of fan-out nodes, whereas a designer can make more intelligent fan-out groups.

12.4.2.2.4 Global Networks

You can use the Global Signal assignment to control the global signal usage on a per-signal basis. For example, if a signal needs local routing, you set the Global Signal assignment to **OFF**.

Figure 76. Global Signal Assignment

To	Assignment Name /	Value	Enabled
reg_clk	Global Signal	Off	Yes

12.4.2.3 Resets and Global Networks

Reset signals are often routed on global networks. Sometimes, the use of a global network causes recovery failures. Consider reviewing the placement of the register that generates the reset and the routing path of the signal.

12.4.2.4 Suspicious Setup

Suspicious setup failures include paths with very small or very large requirements.

One typical cause is math precision error. For example, $10\text{MHz}/3 = 33.33\text{ ns per period}$. In three cycles, the time is 99.999 ns vs 100.000 ns . Setting a maximum delay can provide an appropriate setup relationship.

Another cause of failure are paths that must be false by design intent, such as:

- Asynchronous paths handled through FIFOs, or
- Slow asynchronous paths that rely on handshaking for data that remain available for multiple clock cycles.

To prevent the Fitter from having to meet unnecessarily restrictive timing requirements, consider adding false or multicycle path statements.

12.4.2.5 Logic Depth

The **Statistics** tab in the Timing Analyzer path report shows the levels of logic in a path. If the path fails timing and the number of logic levels is high, consider adding pipelining in that part of the design.

12.4.2.6 Auto Shift Register Replacement

During Synthesis, the Compiler can convert shift registers or register chains into RAMs to save area. However, conversion to RAM often reduces speed. The names of the converted registers include "altshift_taps".

- If paths that fail timing begin or end in shift registers, consider disabling the **Auto Shift Register Replacement** option. Do not convert registers that are intended for pipelining.
- For shift registers that are converted to a chain, evaluate area/speed trade off of implementing in RAM or logic cells.
- If a design is close to full, you can save area by shifting register conversion to RAM, benefiting non-critical clock domains. You can change the settings from the default **AUTO** to **OFF** globally, or on a register or hierarchy basis.

12.4.2.7 Clocking Architecture

For better timing results, place registers driven by a regional clock in one quadrant of the chip. You can review the clock region boundaries using the Chip Planner.

Timing failure can occur when the I/O interface at the top of the device connects to logic driven by a regional clock which is in one quadrant of the device, and placement restrictions force long paths to and from I/Os to logic across quadrants.

Use a different type of clock source to drive the logic - global, which covers the whole device, or dual-regional which covers half the device. Alternatively, you can reduce the frequency of the I/O interface to accommodate the long path delays. You can also redesign the pinout of the device to place all the specified I/Os adjacent to the regional clock quadrant. This issue can happen when register locations are restricted, such as with Logic Lock (Standard) regions, clocking resources, or hard blocks (memories, DSPs, IPs).

The **Extra Fitter Information** tab in the Timing Analyzer timing report informs you when placement is restricted for nodes in a path.

Related Links

[Viewing Available Clock Networks in the Device](#) on page 284

12.4.2.8 Timing Closure Recommendations

The Report Timing Closure Recommendations task in the Timing Analyzer analyzes paths and provides specific recommendations based on path characteristics.

12.4.3 Adjusting and Recompiling

Look for obvious problems that you can fix with minimal effort. To identify where the Compiler had trouble meeting timing, perform seed sweeping with about five compiles. Doing so shows consistently failing paths. Consider recoding or redesigning that part of the design.



To reach timing closure, a well written RTL can be more effective than changing your compilation settings. Seed sweeping can also be useful if the timing failure is very small, and the design has already been optimized for performance improvements and is close to final release. Additionally, seed sweeping can be used for evaluating changes to compilation settings. Compilation results vary due to the random nature of fitter algorithms. If a compilation setting change produces lower average performance, undo the change.

Sometimes, settings or constraints can cause more problems than they fix. When significant changes to the RTL or design architecture have been made, compile periodically with default settings and without Logic Lock (Standard) regions, and re-evaluate paths that fail timing.

Partitioning often does not help timing closure, and must be done at the beginning of the design process. Adding partitions can increase logic utilization if it prevents cross-boundary optimizations, making timing closure harder and increasing compile times.

12.4.3.1 Using Partitions to Achieve Timing Closure

One technique to achieve timing closure is confining failing paths within individual design partitions, such that there are no failing paths passing between partitions. You can then make changes as necessary to correct the failing paths, and recompile only the affected partitions.

To use this technique:

1. In the Design Partition Planner, load timing data by clicking **View ► Show Timing Data**.
Entities containing nodes on failing paths appear in red in the Design Partition Planner.
2. Extract the entity containing failing paths by dragging it outside of the top-level entity window.
 - If there are no failing paths between the extracted entity and the top-level entity, right-click the extracted entity, and then click **Create Design Partition** to place that entity in its own partition.
3. Keep failing paths within a partition, so that there are no failing paths crossing between partitions.
If you are unable to isolate the failing paths from an extracted entity so that none are crossing partition boundaries, return the entity to its parent without creating a partition.
4. Isolate the partition having the worst slack value.
For more details, refer to *Isolating a Partition Netlist*.
5. Adjust the logic in the partition and rerun the Fitter as necessary until the partition meets the timing requirements.
6. In the Design Partitions window, double-click the partition's **Preservation Level** and set to **Final**.
This action fixes the partition's location, and ensures preservation of the improvements as you continue to optimize other partitions.
7. For each partition with failing paths, implement changes as necessary to achieve the timing requirements

Related Links

- [Isolating a Partition Netlist](#) on page 313
- [Viewing Design Connectivity and Hierarchy](#)
- [Using Block-Based Compilation](#) on page 195

12.5 Design Analysis

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

12.5.1 Ignored Timing Constraints

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

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

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

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

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

Related Links

[Creating I/O Requirements](#)

12.5.2 I/O Timing

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

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

Related Links

[Creating I/O Requirements](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 3*



12.5.3 Register-to-Register Timing Analysis

Your design meets timing requirements when you do not have negative slack on any register-to-register path on any of the clock domains. When timing requirements are not met, a report on the failed paths can uncover more detail.

12.5.3.1 Displaying Path Reports with the Timing Analyzer

The Timing Analyzer generate reports with information about all valid register-to-register paths. To view all timing summaries, run the **Report All Summaries** command by double-clicking **Report All Summaries** in the **Tasks** pane.

If any clock domains have failing paths (highlighted in red in the **Report** pane), right-click the clock name listed in the **Clocks Summary** pane and select **Report Timing** to get more details.

When you select a path in the **Summary of Paths** tab, the path detail pane displays all the path information. The **Extra Fitter Information** tab offers visual representation of the path location on the physical device. This can reveal whether the timing failure is distance related, due to the source and destination node being too close or too far.

The **Data Path** tab displays the Data Arrival Path and the Data Required Path. Using the incremental information you can determine the path segments contributing the most to the timing violations. The **Waveform** tab shows the signals in the time domain, and plots the slack between arrival data and required data.

To assess which areas in your design can benefit from reducing the number of logic levels, you can use the RTL Viewer or Technology Map Viewer to see schematic (gate-level or technology-mapped) representations of your design netlist. To locate a timing path in one of the viewers, right-click a path in the timing report, point to **Locate**, and select either **Locate in RTL Viewer** or **Locate in Technology Map Viewer**. You can also use the Chip Planner to investigate the physical layout of a path in more detail.

Related Links

- [Generating Timing Reports](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*
- [When to Use the Netlist Viewers: Analyzing Design Problems](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

12.5.3.2 Tips for Analyzing Failing Paths

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

- Focus on improving the paths that show the worst slack. The Fitter works hardest on paths with the worst slack. If you fix these paths, the Fitter might be able to improve the other failing timing paths in the design.
- Check for nodes that appear in many failing paths. These nodes are at the top of the list in a timing report panel, along with their minimum slacks. Look for paths that have common source registers, destination registers, or common intermediate combinational nodes. In some cases, the registers are not identical, but are part of the same bus.
- In the timing analysis report panels, click the **From** or **To** column headings to sort the paths by source or destination registers. If you see common nodes, these nodes indicate areas of your design that might be improved through source code changes or Intel Quartus Prime optimization settings. Constraining the placement for just one of the paths might decrease the timing performance for other paths by moving the common node further away in the device.

Related Links

- [Exploring Paths in the Chip Planner](#) on page 286
- [Design Evaluation for Timing Closure](#) on page 202

12.5.3.3 Tips for Analyzing Failing Clock Paths that Cross Clock Domains

When analyzing clock path failures:

- Check whether these paths cross two clock domains.

This is the case if the **From Clock** and **To Clock** in the timing analysis report are different.

Figure 77. Different Value in From Clock and To Clock Field

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

- Check if the design contains paths that involve a different clock in the middle of the path, even if the source and destination register clock are the same.
- Check whether failing paths between these clock domains need to be analyzed synchronously.

If the failing paths are not to be analyzed synchronously, they must be set as false paths.

- When you run `report_timing` on your design, the report shows the launch clock and latch clock for each failing path. Check the relationship between the launch clock and latch clock to make sure it is realistic and what you expect from your knowledge of the design

For example, the path can start at a rising edge and end at a falling edge, which reduces the setup relationship by one half clock cycle.



- Review the clock skew reported in the Timing Report:
A large skew may indicate a problem in your design, such as a gated clock, or a problem in the physical layout (for example, a clock using local routing instead of dedicated clock routing). When you have made sure the paths are analyzed synchronously and that there is no large skew on the path, and that the constraints are correct, you can analyze the data path. These steps help you fine tune your constraints for paths across clock domains to ensure you get an accurate timing report.
- Check if the PLL phase shift is reducing the setup requirement.
You might adjust this by using PLL parameters and settings.
- Ignore paths that cross clock domains if the logic is protected with synchronization logic (for example, FIFOs or double-data synchronization registers), even if the clocks are related.
- Set false path constraints on all unnecessary paths:
Attempting to optimize unnecessary paths can prevent the Fitter from meeting the timing requirements on timing paths that are critical to the design.

Related Links

[report_clock_transfers](#)

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

When analyzing the failing paths in a design, it is often helpful to get a fuller picture of the interactions around the paths.

To understand what may be pulling on a critical path, the following `report_timing` command can be useful.

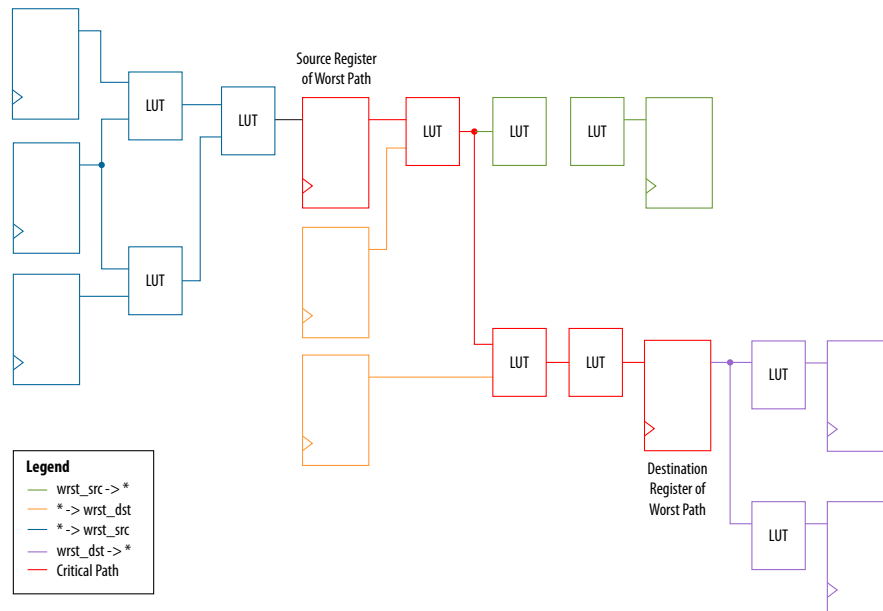
1. In the project directory, run the `report_timing` command to find the nodes in a critical path.
2. Copy the code below in a `.tcl` file, and replace the first two variable with the node names from the **From Node** and **To Node** columns of the worst path. The script analyzes the path between the worst source and destination registers.

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

3. From the **Script** menu, source the `.tcl` file.
4. In the resulting timing panel, locate timing failed paths (highlighted in red) in the Chip Planner, and view information such as distance between the nodes and large fanouts.

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

Figure 78. Timing Report



The critical path of the design is in red. The relation between the .tcl script and the figure is:

- The first two lines show everything inside the two endpoints of the critical path that are pulling them in different directions.
 - The first `report_timing` command analyzes all paths the source is driving, shown in green.
 - The second `report_timing` command analyzes all paths going to the destination, including the critical path, shown in orange.
- The last two `report_timing` commands show everything outside of the endpoints pulling them in other directions.

If any of these neighboring paths have slacks near the critical path, the Fitter is balancing these paths with the critical path, trying to achieve the best slack.

12.5.3.5 Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles

Many designs have the same critical paths show up after each compile. In other designs, critical paths bounce around between different hierarchies, changing with each compile.

This behavior happens in high speed designs where many register-to-register paths have very little slack. Different placements can then result in timing failures in the marginal paths.

1. In the project directory, create a script named `TQ_critical_paths.tcl`.
2. After compilation, review the critical paths and then write a generic `report_timing` command to capture those paths.



For example, if several paths fail in a low-level hierarchy, add a command such as:

```
report_timing -setup -npaths 50 -detail path_only \  
-to "main_system: main_system_inst|app_cpu:cpu|*" \  
-panel_name "Critical Paths||s: * -> app_cpu"
```

3. If there is a specific path, such as a bit of a state-machine going to other `*count_sync*` registers, you can add a command similar to:

```
report_timing -setup -npaths 50 -detail path_only \  
-from "main_system: main_system_inst|egress_count_sm:egress_inst|  
update" \  
-to "*count_sync*" -panel_name "Critical Paths||s: egress_sm|  
update -> count_sync"
```

4. Execute this script in the Timing Analyzer after every compilation, and add new `report_timing` commands as new critical paths appear.

This helps you monitor paths that consistently fail and paths that are only marginal, so you can prioritize effectively

12.5.3.6 Global Routing Resources

Global routing resources are designed to distribute high fan-out, low-skew signals (such as clocks) without consuming regular routing resources. Depending on the device, these resources can span the entire chip or a smaller portion, such as a quadrant. The Intel Quartus Prime software attempts to assign signals to global routing resources automatically, but you might be able to make more suitable assignments manually.

For details about the number and types of global routing resources available, refer to the relevant device handbook.

Check the global signal utilization in your design to ensure that the appropriate signals have been placed on the global routing resources. In the Compilation Report, open the Fitter report and click **Resource Section**. Analyze the Global & Other Fast Signals and Non-Global High Fan-out Signals reports to determine whether any changes are required.

You might be able to reduce skew for high fan-out signals by placing them on global routing resources. Conversely, you can reduce the insertion delay of low fan-out signals by removing them from global routing resources. Doing so can improve clock enable timing and control signal recovery/removal timing, but increases clock skew. Use the **Global Signal** setting in the Assignment Editor to control global routing resources.

12.6 Timing Optimization

Use the following guidelines if your design does not meet its timing requirements.

12.6.1 Displaying Timing Closure Recommendations for Failing Paths

Use the **Timing Closure Recommendations** report to get specific recommendations about failing paths in your design and changes that you can make to potentially fix the failing paths.



1. In the **Tasks** pane of the Timing Analyzer, select the **Report Timing Closure Recommendations** task to open the **Report Timing Closure Recommendations** dialog box.
2. Select paths based on the clock domain, filter by nodes on path, and choose the number of paths to analyze.
3. After running the **Report Timing Closure Recommendations** task in the Timing Analyzer, examine the reports in the **Report Timing Closure Recommendations** folder in the **Report** pane of the Timing Analyzer GUI. Each recommendation has star symbols (*) associated with it. Recommendations with more stars are more likely to help you close timing on your design.

The reports give you the most probable causes of failure for each analyzed path, and show recommendations that may help you fix the failing paths.

The reports are organized into sections, depending on the type of issues found in the design, such as large clock skew, restricted optimizations, unbalanced logic, skipped optimizations, coding style that has too many levels of logic between registers, or region or partition constraints specific to your project.

For detailed analysis of the critical paths, run the `report_timing` command on specified paths. In the **Extra Fitter Information** tab of the **Path** report panel, you can see detailed fitter-related information that may help you visualize the issue.

Related Links

- [Report Timing Closure Recommendations Dialog Box](#)
- [Fast Forward Timing Closure Recommendations](#)

12.6.2 Timing Optimization Advisor

While the Timing Analyzer **Report Timing Closure Recommendations** task gives specific recommendations to fix failing paths, the Timing Optimization Advisor gives more general recommendations to improve timing performance for a design.

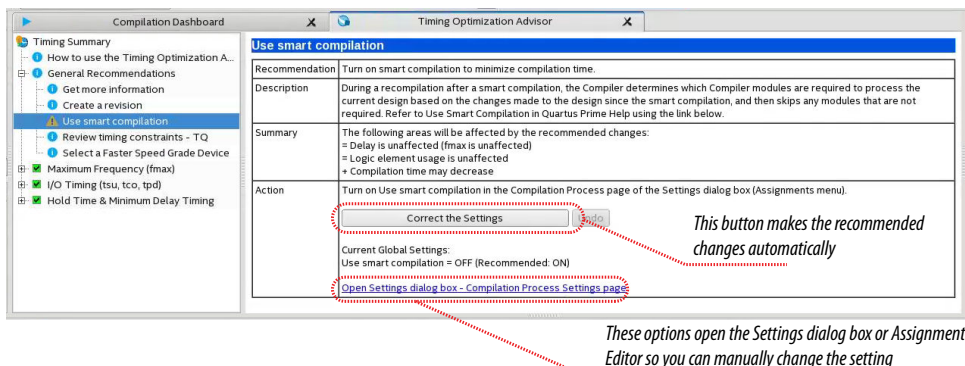
The Timing Optimization Advisor guides you in making settings that optimize your design to meet your timing requirements. To run the Timing Optimization Advisor click **Tools > Advisors > Timing Optimization Advisor**. This advisor describes many of the suggestions made in this section.

When you open the Timing Optimization Advisor after compilation, you can find recommendations to improve the timing performance of your design. If suggestions in these advisors contradict each other, evaluate these options and choose the settings that best suit the given requirements.

The example shows the Timing Optimization Advisor after compiling a design that meets its frequency requirements, but requires setting changes to improve the timing.



Figure 79. Timing Optimization Advisor



When you expand one of the categories in the Timing Optimization Advisor, such as **Maximum Frequency (fmax)** or **I/O Timing (tsu, tco, tpd)**, the recommendations appear in stages. These stages show the order in which to apply the recommended settings.

The first stage contains the options that are easiest to change, make the least drastic changes to your design optimization, and have the least effect on compilation time.

Icons indicate whether each recommended setting has been made in the current project. In the figure, the checkmark icons in the list of recommendations for Stage 1 indicates recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icons indicate general suggestions. For these entries, the advisor does not report whether these recommendations were followed, but instead explains how you can achieve better performance. For a legend that provides more information for each icon, refer to the “How to use” page in the Timing Optimization Advisor.

Each recommendation provides a link to the appropriate location in the Intel Quartus Prime GUI where you can change the settings. For example, consider the **Synthesis Netlist Optimizations** page of the **Settings** dialog box or the **Global Signals category** in the Assignment Editor. This approach provides the most control over which settings are made and helps you learn about the settings in the software. When available, you can also use the **Correct the Settings** button to automatically make the suggested change to global settings.

For some entries in the Timing Optimization Advisor, a button allows you to further analyze your design and see more information. The advisor provides a table with the clocks in the design, indicating whether they have been assigned a timing constraint.

12.6.3 Optional Fitter Settings

This section focuses only on the optional timing-optimization Fitter settings, which are the **Optimize Hold Timing**, **Optimize Multi-Corner Timing**, and **Fitter Aggressive Routability Optimization**.

Caution: The settings that best optimize different designs might vary. The group of settings that work best for one design does not necessarily produce the best result for another design.

Related Links

[Advanced Fitter Setting Dialog Box](#)

12.6.3.1 Optimize Hold Timing

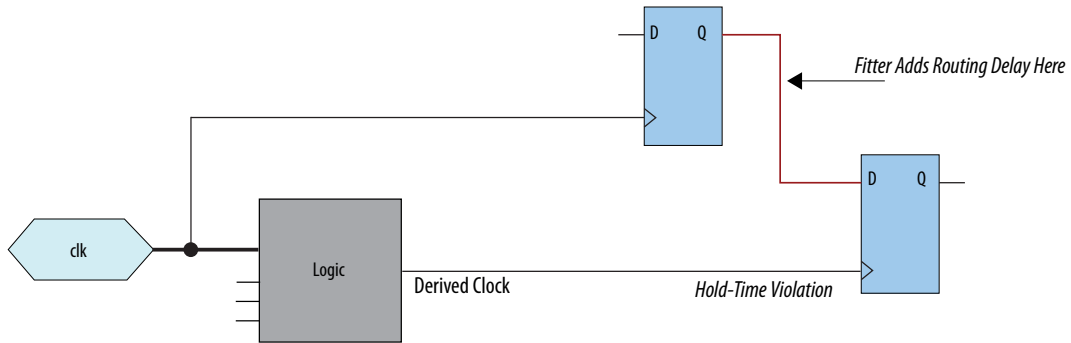
The **Optimize Hold Timing** option directs the Intel Quartus Prime software to optimize minimum delay timing constraints. Check your device information to determine whether the Intel Quartus Prime software optimizes hold timing for all paths or only for I/O paths and minimum t_{PD} paths.

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

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

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

Figure 80. Optimize Hold Timing Option Fixing an Internal Hold Time Violation



However, if your design still has internal hold time violations between registers, you can manually add delays by instantiating LCELL primitives, or by making changes to your design, such as using a clock enable signal instead of a derived or gated clock.

Related Links

[Recommended Design Practices](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 1*

12.6.3.2 Fitter Aggressive Routability Optimization

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

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



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

Table 31. Fitter Aggressive Routability Optimizations Logic Option Settings

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

12.6.4 I/O Timing Optimization Techniques

This stage of design optimization focuses on I/O timing, including setup delay (t_{SU}), hold time (t_H), and clock-to-output (t_{CO}) parameters.

Before proceeding with I/O timing optimization, ensure that:

- The design's assignments follow the suggestions in the *Initial Compilation: Required Settings* section of the *Design Optimization Overview* chapter.
- Resource utilization is satisfactory.

You can apply the suggestions this section to all Intel FPGA families and to the family of CPLDs.

Note: Complete this stage before proceeding to the register-to-register timing optimization stage. Changes to the I/O paths affect the internal register-to-register timing.

Summary of Techniques for Improving Setup and Clock-to-Output Times

The table lists the recommended order of techniques to reduce t_{SU} and t_{CO} times. Reducing t_{SU} times increases hold (t_H) times.

Note: Verify which options are available to each device family

Table 32. Improving Setup and Clock-to-Output Times

Order	Technique	Affects t_{SU}	Affects t_{CO}
1	Verify of that the appropriate constraints are set for the failing I/Os (refer to <i>Initial Compilation: Required Settings</i>)	Yes	Yes
2	Use timing-driven compilation for I/O (refer to <i>Fast Input, Output, and Output Enable Registers</i>)	Yes	Yes
3	Use fast input register (refer to <i>Programmable Delays</i>)	Yes	N/A
4	Use fast output register, fast output enable register, and fast OCT register (refer to <i>Programmable Delays</i>)	N/A	Yes
5	Decrease the value of Input Delay from Pin to Input Register or set Decrease Input Delay to Input Register = ON	Yes	N/A
6	Decrease the value of Input Delay from Pin to Internal Cells or set Decrease Input Delay to Internal Cells = ON	Yes	N/A
<i>continued...</i>			



Order	Technique	Affects t_{SU}	Affects t_{CO}
7	Decrease the value of Delay from Output Register to Output Pin or set Increase Delay to Output Pin = OFF (refer to <i>Fast Input, Output, and Output Enable Registers</i>)	N/A	Yes
8	Increase the value of Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations (refer to <i>Fast Input, Output, and Output Enable Registers</i>)	Yes	N/A
9	Use PLLs to shift clock edges	Yes	Yes
10	Use the Fast Regional Clock (refer to <i>Change How Hold Times are Optimized for MAX II Devices</i>)	N/A	Yes
11	For MAX II or MAX V family devices, set Guarantee I/O Paths Have Zero Hold Time at Fast Corner to OFF , or When T_{SU} and T_{PD} Constraints Permit (refer to <i>Change How Hold Times are Optimized for MAX II Devices</i>)	Yes	N/A
12	Increase the value of Delay to output enable pin or set Increase delay to output enable pin (refer to <i>Use PLLs to Shift Clock Edges</i>)	N/A	Yes

[Optimize IOC Register Placement for Timing Logic Option](#) on page 226

[Fast Input, Output, and Output Enable Registers](#) on page 227

[Programmable Delays](#) on page 227

[Use PLLs to Shift Clock Edges](#) on page 228

[Use Fast Regional Clock Networks and Regional Clocks Networks](#) on page 228

[Spine Clock Limitations](#) on page 229

[Change How Hold Times are Optimized for Devices](#) on page 229

Related Links

[Initial Compilation: Required Settings](#) on page 184

12.6.4.1 Optimize IOC Register Placement for Timing Logic Option

This option moves registers into I/O elements to meet t_{SU} or t_{CO} assignments, duplicating the register if necessary (as in the case in which a register fans out to multiple output locations). This option is turned on by default and is a global setting.

Note: The option does not apply to series devices because they do not contain I/O registers.

The **Optimize IOC Register Placement for Timing** logic option affects only pins that have a t_{SU} or t_{CO} requirement. Using the I/O register is possible only if the register directly feeds a pin or is fed directly by a pin. Therefore, this logic option does not affect registers with any of the following characteristics:

Note: To optimize registers with these characteristics, use other Intel Quartus Prime Fitter optimizations.

- Have combinational logic between the register and the pin
- Are part of a carry or cascade chain
- Have an overriding location assignment
- Use the asynchronous load port and the value is not 1 (in device families where the port is available)



Related Links

[Optimize IOC Register Placement for Timing Logic Option](#)

12.6.4.2 Fast Input, Output, and Output Enable Registers

You can place individual registers in I/O cells manually by making fast I/O assignments with the Assignment Editor. By default, with correct timing assignments, the Fitter places the I/O registers in the correct I/O cell or in the core, to meet the performance requirement.

In series devices, which have no I/O registers, these assignments lock the register into the LAB adjacent to the I/O pin if there is a pin location assignment for that I/O pin.

If the fast I/O setting is on, the register is always placed in the I/O element. If the fast I/O setting is off, the register is never placed in the I/O element. This is true even if the **Optimize IOC Register Placement for Timing** option is turned on. If there is no fast I/O assignment, the Intel Quartus Prime software determines whether to place registers in I/O elements if the **Optimize IOC Register Placement for Timing** option is turned on.

You can also use the four fast I/O options (**Fast Input Register**, **Fast Output Register**, **Fast Output Enable Register**, and **Fast OCT Register**) to override the location of a register that is in a Logic Lock (Standard) region and force it into an I/O cell. If you apply this assignment to a register that feeds multiple pins, the Fitter duplicates the register and places it in all relevant I/O elements.

In series devices, the Fitter duplicates the register and places it in each distinct LAB location that is next to an I/O pin with a pin location assignment.

For more information about the **Fast Input Register** option, **Fast Output Register** option, **Fast Output Enable Register** option, and **Fast OCT (on-chip termination) Register** option, refer to Intel Quartus Prime Help.

Related Links

- [Fast Input Register logic option](#)
- [Fast Output Register logic option](#)
- [Fast Output Enable Register logic option](#)
- [Fast OCT Register logic option](#)

12.6.4.3 Programmable Delays

You can use various programmable delay options to minimize the t_{SU} and t_{CO} times. Programmable delays are advanced options that you use only after you compile a project, check the I/O timing, and determine that the timing is unsatisfactory.

For Arria, Cyclone, MAX II, MAX V, and Stratix series devices, the Intel Quartus Prime software automatically adjusts the applicable programmable delays to help meet timing requirements. For detailed information about the effect of these options, refer to the device family handbook or data sheet.

After you have made a programmable delay assignment and compiled the design, you can view the implemented delay values for every delay chain and every I/O pin in the **Delay Chain Summary** section of the Compilation Report.

You can assign programmable delay options to supported nodes with the Assignment Editor. You can also view and modify the delay chain setting for the target device with the Chip Planner and Resource Property Editor. When you use the Resource Property Editor to make changes after performing a full compilation, recompiling the entire design is not necessary; you can save changes directly to the netlist. Because these changes are made directly to the netlist, the changes are not made again automatically when you recompile the design. The change management features allow you to reapply the changes on subsequent compilations.

Although the programmable delays in newer devices are user-controllable, Intel recommends their use for advanced users only. However, the Intel Quartus Prime software might use the programmable delays internally during the Fitter phase.

For details about the programmable delay logic options available for Intel devices, refer to the following Intel Quartus Prime Help topics:

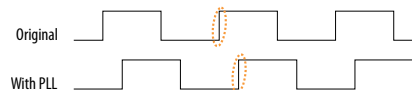
Related Links

- [Input Delay from Pin to Input Register logic option](#)
- [Input Delay from Pin to Internal Cells logic option](#)
- [Output Enable Pin Delay logic option](#)
- [Delay from Output Register to Output Pin logic option](#)
- [Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations logic option](#)

12.6.4.4 Use PLLs to Shift Clock Edges

Using a PLL typically improves I/O timing automatically. If the timing requirements are still not met, most devices allow the PLL output to be phase shifted to change the I/O timing. Shifting the clock backwards gives a better t_H at the expense of t_{SU} , while shifting it forward gives a better t_{SU} at the expense of t_H . You can use this technique only in devices that offer PLLs with the phase shift option.

Figure 81. Shift Clock Edges Forward to Improve t_{SU} at the Expense of t_H



You can achieve the same type of effect in certain devices by using the programmable delay called **Input Delay from Dual Purpose Clock Pin to Fan-Out Destinations**.

12.6.4.5 Use Fast Regional Clock Networks and Regional Clocks Networks

Regional clocks provide the lowest clock delay and skew for logic contained in a single quadrant. In general, fast regional clocks have less delay to I/O elements than regional and global clocks, and are used for high fan-out control signals. Placing clocks on these low-skew and low-delay clock nets provides better t_{CO} performance.

Intel devices have a variety of hierarchical clock structures. These include dedicated global clock networks, regional clock networks, fast regional clock networks, and periphery clock networks. The available resources differ between the various Intel device families.

For the number of clocking resources available in your target device, refer to the appropriate device handbook.



12.6.4.6 Spine Clock Limitations

In projects with high clock routing demands, limitations in the Intel Quartus Prime software can cause spine clock errors. These errors are often seen with designs using multiple memory interfaces and high-speed serial interface (HSSI) channels, especially PMA Direct mode.

Global clock networks, regional clock networks, and periphery clock networks have an additional level of clock hierarchy known as spine clocks. Spine clocks drive the final row and column clocks to their registers; thus, the clock to every register in the chip is reached through spine clocks. Spine clocks are not directly user controllable.

To reduce these spine clock errors, constrain your design to use your regional clock resources better:

- If your design does not use Logic Lock (Standard) regions, or if the Logic Lock (Standard) regions are not aligned to your clock region boundaries, create additional Logic Lock (Standard) regions and further constrain your logic.
- If Periphery features ignore Logic Lock (Standard) region assignment, possibly because the global promotion process is not functioning properly. To ensure that the global promotion process uses the correct locations, assign specific pins to the I/Os using these periphery features.
- By default, some Intel FPGA IP functions apply a global signal assignment with a value of dual-regional clock. If you constrain your logic to a regional clock region and set the global signal assignment to **Regional** instead of **Dual-Regional**, you can reduce clock resource contention.

Related Links

- [Viewing Available Clock Networks in the Device](#) on page 284
- [Layers Settings and Editing Modes](#) on page 281
- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)
In Intel Quartus Prime Help

12.6.4.7 Change How Hold Times are Optimized for Devices

For devices, you can use the **Guarantee I/O Paths Have Zero Hold Time at Fast Corner** option to control how hold time is optimized by the Intel Quartus Prime software.

12.6.5 Register-to-Register Timing Optimization Techniques

The next stage of design optimization seeks to improve register-to-register (f_{MAX}) timing. The following sections provide available options if the performance requirements are not achieved after compilation.

Coding style affects the performance of your design to a greater extent than other changes in settings. Always evaluate your code and make sure to use synchronous design practices.

Note: When using the Timing Analyzer, register-to-register timing optimization is the same as maximizing the slack on the clock domains in your design. You can use the techniques described in this section to improve the slack on different timing paths in your design.



Before optimizing your design, understand the structure of your design as well as the type of logic affected by each optimization. An optimization can decrease performance if the optimization does not benefit your logic structure.

Related Links

[Recommended Design Practices](#)

In *Intel Quartus Prime Standard Edition Handbook Volume 1*

12.6.5.1 Optimize Source Code

In many cases, optimizing the design's source code can have a very significant effect on your design performance. In fact, optimizing your source code is typically the most effective technique for improving the quality of your results and is often a better choice than using Logic Lock (Standard) or location assignments.

Be aware of the number of logic levels needed to implement your logic while you are coding. Too many levels of logic between registers might result in critical paths failing timing. Try restructuring the design to use pipelining or more efficient coding techniques. Also, try limiting high fan-out signals in the source code. When possible, duplicate and pipeline control signals. Make sure the duplicate registers are protected by a preserve attribute, to avoid merging during synthesis.

If the critical path in your design involves memory or DSP functions, check whether you have code blocks in your design that describe memory or functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to cause these functions to be placed into high-performance dedicated memory or resources in the target device. When using RAM/DSP blocks, enable the optional input and output registers.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Intel Quartus Prime software, you can check the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including state encoding for each state machine that was recognized during compilation. If your state machine is not recognized, you might have to change your source code to enable it to be recognized.

Related Links

- [Recommended HDL Coding Styles](#)
In *Intel Quartus Prime Pro Edition Handbook Volume 1*
- [AN 584: Timing Closure Methodology for Advanced FPGA Designs](#)

12.6.5.2 Improving Register-to-Register Timing

The choice of options and settings to improve the timing margin (slack) or to improve register-to-register timing depends on the failing paths in the design. To achieve the results that best approximate your performance requirements, apply the following techniques and compile the design after each step:



1. Ensure that your timing assignments are complete and correct. For details, refer to the *Initial Compilation: Required Settings* section in the *Design Optimization Overview* chapter.
2. Review all warning messages from your initial compilation and check for ignored timing assignments.
3. Apply netlist synthesis optimization options.
4. To optimize for speed, apply the following synthesis options:
 - Optimize Synthesis for Speed, Not Area
 - Flatten the Hierarchy During Synthesis
 - Set the Synthesis Effort to High
 - Change State Machine Encoding
 - Prevent Shift Register Inference
 - Use Other Synthesis Options Available in Your Synthesis Tool
5. To optimize for performance using physical synthesis, apply the following options:
 - Enable physical synthesis
 - Perform automatic asynchronous signal pipelining
 - Perform register duplication
 - Perform register retiming
 - Perform logic to memory mapping
6. Try different Fitter seeds. If only a small number of paths are failing by small negative slack, then you can try with a different seed to find a fit that meets constraints in the Fitter seed noise.

Note: Omit this step if a large number of critical paths are failing, or if the paths are failing by a long margin.
7. To control placement, make Logic Lock (Standard) assignments.
8. Modify your design source code to fix areas of the design that are still failing timing requirements by significant amounts.
9. Make location assignments, or as a last resort, perform manual placement by back-annotating the design.

You can use Design Space Explorer II (DSE) to automate the process of running different compilations with different settings.

If these techniques do not achieve performance requirements, additional design source code modifications might be required.

Related Links

- [Launch Design Space Explorer Command \(Tools Menu\)](#)
- [Initial Compilation: Required Settings](#)

12.6.5.3 Physical Synthesis Optimizations

The Intel Quartus Prime software offers physical synthesis optimizations that can help improve the performance of many designs, regardless of the synthesis tool you use. You can apply physical synthesis optimizations both during synthesis and during fitting.



During the synthesis stage of the Intel Quartus Prime compilation, physical synthesis optimizations operate either on the output from another EDA synthesis tool, or as an intermediate step in synthesis. These optimizations modify the synthesis netlist to improve either area or speed, depending on the technique and effort level you select.

To view and modify the synthesis netlist optimization options, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**.

If you use a third-party EDA synthesis tool and want to determine if the Intel Quartus Prime software can remap the circuit to improve performance, use the **Perform WYSIWYG Primitive Resynthesis** option. This option directs the Intel Quartus Prime software to unmap the LEs in an atom netlist to logic gates, and then map the gates back to Intel-specific primitives. Using Intel-specific primitives enables the Fitter to remap the circuits using architecture-specific techniques.

The Intel Quartus Prime technology mapper optimizes the design to achieve maximum speed performance, minimum area usage, or balances high performance and minimal logic usage, according to the setting of the **Optimization Technique** option. Set this option to **Speed** or **Balanced**.

During the Fitter stage of the Intel Quartus Prime compilation, physical synthesis optimizations make placement-specific changes to the netlist that improve speed performance results for the specific Intel device.

Note: If you want the performance gain from physical synthesis only on parts of your design, you can apply the physical synthesis options on specific instances with the Assignment Editor.

Related Links

- [Perform WYSIWYG Primitive Resynthesis Logic Option](#)
- [Optimization Technique Logic Option](#)

12.6.5.4 Turn Off Extra-Effort Power Optimization Settings

If power optimization settings are set to **Extra Effort**, your design performance can be affected. If timing performance is more important than power, set the power optimization setting to **Normal**.

Related Links

- [Power Optimization Logic Option](#)
- [Power Optimization](#) on page 247

12.6.5.5 Optimize Synthesis for Speed, Not Area

Design performance varies depending on coding style, synthesis tool used, and options you specify when synthesizing. Change your synthesis options if a large number of paths are failing, or if specific paths fail by a great margin and have many levels of logic.

Identify the default optimization targets of your Synthesis tool, and set your device and timing constraints accordingly. For example, if you do not specify a target frequency, some synthesis tools optimize for area.



You can specify logic options for specific modules in your design with the Assignment Editor while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families) or **Area** (if area is an important concern). You can also use the **Speed Optimization Technique for Clock Domains** option in the Assignment Editor to specify that all combinational logic in or between the specified clock domains are optimized for speed.

To achieve best performance with push-button compilation, follow the recommendations in the following sections for other synthesis settings. You can use DSE II to experiment with different Intel Quartus Prime synthesis options to optimize your design for the best performance.

Related Links

[Optimization Technique Logic Option](#)

12.6.5.6 Flatten the Hierarchy During Synthesis

Synthesis tools typically let you preserve hierarchical boundaries, which can be useful for verification or other purposes. However, the best optimization results generally occur when the synthesis tool optimizes across hierarchical boundaries, because doing so often allows the synthesis tool to perform the most logic minimization, which can improve performance. Whenever possible, flatten your design hierarchy to achieve the best results.

Note: If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.

12.6.5.7 Set the Synthesis Effort to High

Synthesis tools offer varying synthesis effort levels to trade off compilation time with synthesis results. Set the synthesis effort to **high** to achieve best results when applicable.

12.6.5.8 Change State Machine Encoding

State machines can be encoded using various techniques. One-hot encoding, which uses one register for every state bit, usually provides the best performance. If your design contains state machines, changing the state machine encoding to one-hot can improve performance at the cost of area.

Related Links

[State Machine Processing Logic Option online help](#)

12.6.5.9 Duplicate Logic for Fan-Out Control

Oftentimes, timing failures occur not because of the high fan-out registers, but because of the location of those registers. Duplicating registers, where source and destination registers are physically close, can help improve slack on critical paths.

Synthesis tools support options or attributes that specify the maximum fan-out of a register. When using Intel Quartus Prime integrated synthesis, you can set the **Maximum Fan-Out** logic option in the Assignment Editor to control the number of

destinations for a node so that the fan-out count does not exceed a specified value. You can also use the `maxfan` attribute in your HDL code. The software duplicates the node as required to achieve the specified maximum fan-out.

Logic duplication using **Maximum Fan-Out** assignments normally increases resource utilization, and can potentially increase compilation time, depending on the placement and the total resource usage within the selected device.

The improvement in timing performance that results from **Maximum Fan-Out** assignments is design-specific. This is because when you use the **Maximum Fan-Out** assignment, the Fitter duplicates the source logic to limit the fan-out, but does not control the destinations that each of the duplicated sources drive. Therefore, it is possible for duplicated source logic to be driving logic located all around the device. To avoid this situation, you can use the **Manual Logic Duplication** logic option.

If you are using **Maximum Fan-Out** assignments, benchmark your design with and without these assignments to evaluate whether they give the expected improvement in timing performance. Use the assignments only when you get improved results.

You can manually duplicate registers in the Intel Quartus Prime software regardless of the synthesis tool used. To duplicate a register, apply the **Manual Logic Duplication** logic option to the register with the Assignment Editor.

Note: Some Fitter optimizations may cause a small violation to the **Maximum Fan-Out** assignments to improve timing.

12.6.5.10 Prevent Shift Register Inference

Turning off the inference of shift registers can increase performance. This setting forces the software to use logic cells to implement the shift register, instead of using the `ALTSHIFT_TAPS` IP core to implement the registers in memory block. If you implement shift registers in logic cells instead of memory, logic utilization increases.

12.6.5.11 Use Other Synthesis Options Available in Your Synthesis Tool

With your synthesis tool, experiment with the following options if they are available:

- Turn on register balancing or retiming
- Turn on register pipelining
- Turn off resource sharing

These options can increase performance, but typically increase the resource utilization of your design.

12.6.5.12 Fitter Seed

The Fitter seed affects the initial placement configuration of the design. Any change in the initial conditions changes the Fitter results; accordingly, each seed value results in a somewhat different fit. You can experiment with different seeds to attempt to obtain better fitting results and timing performance.

Changes in your design impact performance between compilations. This random variation is inherent in placement and routing algorithms—it is impossible to try all seeds and get the absolute best result.



Note: Any design change that directly or indirectly affects the Fitter has the same type of random effect as changing the seed value. This includes any change in source files, **Compiler Settings** or **Timing Analyzer Settings**. The same effect can appear if you use a different computer processor type or different operating system, because different systems can change the way floating point numbers are calculated in the Fitter.

If a change in optimization settings marginally affects the register-to-register timing or number of failing paths, you cannot always be certain that your change caused the improvement or degradation, or whether it is due to random effects in the Fitter. If your design is still changing, running a seed sweep (compiling your design with multiple seeds) determines whether the average result improved after an optimization change, and whether a setting that increases compilation time has benefits worth the increased time, such as with physical synthesis settings. The sweep also shows the amount of random variation to expect for your design.

If your design is finalized you can compile your design with different seeds to obtain one optimal result. However, if you subsequently make any changes to your design, you might need to perform seed sweep again.

Click **Assignments** ► **Compiler Settings** to control the initial placement with the seed. You can use the DSE II to perform a seed sweep easily.

To specify a Fitter seed use the following Tcl command :

```
set_global_assignment -name SEED <value>
```

Related Links

[Launch Design Space Explorer Command \(Tools Menu\)](#)

12.6.5.13 Set Maximum Router Timing Optimization Level

To improve routability in designs where the router did not pick up the optimal routing lines, set the **Router Timing Optimization Level** to **Maximum**. This setting determines how aggressively the router tries to meet the timing requirements. Setting this option to **Maximum** can marginally increase design speed at the cost of increased compilation time. Setting this option to **Minimum** can reduce compilation time at the cost of marginally reduced design speed. The default value is **Normal**.

Related Links

[Router Timing Optimization Level Logic Option](#)

12.6.6 Logic Lock (Standard) Assignments

Using Logic Lock (Standard) assignments to improve timing performance is only recommended for older devices, such as the MAX II family. For other device families, especially for larger devices such as Arria and Stratix series devices, do not use Logic Lock (Standard) assignments to improve timing performance. For these devices, use the feature for performance preservation and to floorplan your design.

Logic Lock (Standard) assignments do not always improve the performance of the design. In many cases, you cannot improve upon results from the Fitter by making location assignments. If there are existing Logic Lock (Standard) assignments in your design, remove the assignments if your design methodology permits it. Recompile the design, and then check if the assignments are making the performance worse.

When making Logic Lock (Standard) assignments, it is important to consider how much flexibility to give the Fitter. Logic Lock (Standard) assignments provide more flexibility than hard location assignments. Assignments that are more flexible require higher Fitter effort, but reduce the chance of design overconstraint.

The following types of Logic Lock (Standard) assignments are available, listed in the order of decreasing flexibility:

- Auto size, floating location regions
- Fixed size, floating location regions
- Fixed size, locked location regions

If you are unsure about the best size or location of a Logic Lock (Standard) region, the **Auto/Floating** options are useful for your first pass. After you determine where a Logic Lock (Standard) region must go, modify the Fixed/Locked regions, as Auto/Floating Logic Lock (Standard) regions can hurt your overall performance. To determine what to put into a Logic Lock (Standard) region, refer to the timing analysis results and analyze the critical paths in the Chip Planner. The register-to-register timing paths in the Timing Analyzer section of the Compilation Report help you recognize patterns.

Related Links

[Analyzing and Optimizing the Design Floorplan](#) on page 280

12.6.6.1 Hierarchy Assignments

For a design with the hierarchy shown in the figure, which has failing paths in the timing analysis results similar to those shown in the table, `mod_A` is probably a problem module. In this case, a good strategy to fix the failing paths is to place the `mod_A` hierarchy block in a Logic Lock (Standard) region so that all the nodes are closer together in the floorplan.

Figure 82. Design Hierarchy

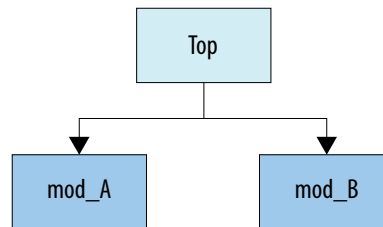


Table 33. Failing Paths in a Module Listed in Timing Analysis

From	To
mod_A reg1	mod_A reg9
mod_A reg3	mod_A reg5
mod_A reg4	mod_A reg6
mod_A reg7	mod_A reg10
mod_A reg0	mod_A reg2



Hierarchical Logic Lock (Standard) regions are also important if you are using an incremental compilation flow. Place each design partition for incremental compilation in a separate Logic Lock (Standard) region to reduce conflicts and ensure good results as the design develops. You can use the auto size and floating location regions to find a good design floorplan, but fix the size and placement to achieve the best results in future compilations.

Related Links

[Analyzing and Optimizing the Design Floorplan](#) on page 280

12.6.7 Location Assignments

If a small number of paths are failing to meet their timing requirements, you can use hard location assignments to optimize placement.

Location assignments are less flexible for the Intel Quartus Prime Fitter than Logic Lock (Standard) assignments. Additionally, if you are familiar with your design, you can enter location constraints in a way that produces better results.

Note: Improving fitting results, especially for larger devices, such as Arria and Stratix series devices, can be difficult. Location assignments do not always improve the performance of the design. In many cases, you cannot improve upon the results from the Fitter by making location assignments.

12.6.8 Metastability Analysis and Optimization Techniques

Metastability problems can occur when a signal is transferred between circuitry in unrelated or asynchronous clock domains, because the designer cannot guarantee that the signal meets its setup and hold time requirements. The mean time between failures (MTBF) is an estimate of the average time between instances when metastability could cause a design failure.

You can use the Intel Quartus Prime software to analyze the average MTBF due to metastability when a design synchronizes asynchronous signals and to optimize the design to improve the MTBF. These metastability features are supported only for designs constrained with the Timing Analyzer, and for select device families.

If the MTBF of your design is low, refer to the Metastability Optimization section in the Timing Optimization Advisor, which suggests various settings that can help optimize your design in terms of metastability.

This chapter describes how to enable metastability analysis and identify the register synchronization chains in your design, provides details about metastability reports, and provides additional guidelines for managing metastability.

Related Links

- [Understanding Metastability in FPGAs](#)
- [Managing Metastability with the Intel Quartus Prime Software](#)
In Intel Quartus Prime Standard Edition Handbook Volume 1

12.7 Periphery to Core Register Placement and Routing Optimization

The Periphery to Core Register Placement and Routing Optimization (P2C) option specifies whether the Fitter performs targeted placement and routing optimization on direct connections between periphery logic and registers in the FPGA core. P2C is an optional pre-routing-aware placement optimization stage that enables you to more reliably achieve timing closure.

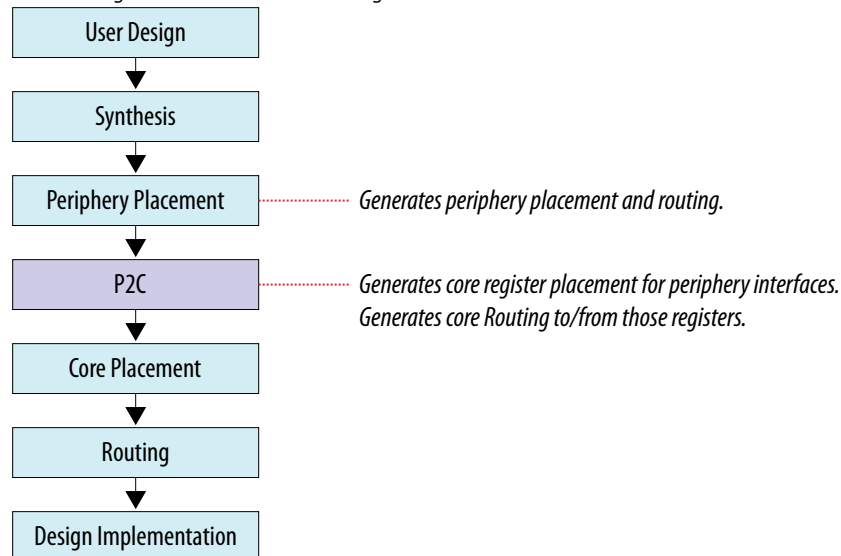
Note: The **Periphery to Core Register Placement and Routing Optimization** option applies in both directions, periphery to core and core to periphery.

Transfers between external interfaces (for example, high-speed I/O or serial interfaces) and the FPGA often require routing many connections with tight setup and hold timing requirements. When this option is turned on, the Fitter performs P2C placement and routing decisions before those for core placement and routing. This reserves the necessary resources to ensure that your design achieves its timing requirements and avoids routing congestion for transfers with external interfaces.

This option is available as a global assignment, or can be applied to specific instances within your design.

Figure 83. Periphery to Core Register Placement and Routing Optimization (P2C) Flow

P2C runs after periphery placement, and generates placement for core registers on corresponding P2C/C2P paths, and core routing to and from these core registers.



[Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box on page 239](#)

[Setting Periphery to Core Optimizations in the Assignment Editor on page 239](#)

[Viewing Periphery to Core Optimizations in the Fitter Report on page 240](#)



12.7.1 Setting Periphery to Core Optimizations in the Advanced Fitter Setting Dialog Box

The **Periphery to Core Placement and Routing Optimization** setting specifies whether the Fitter optimizes targeted placement and routing on direct connections between periphery logic and registers in the FPGA core.

You can optionally perform periphery to core optimizations by instance with settings in the Assignment Editor.

1. In the Intel Quartus Prime software, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.
2. In the **Advanced Fitter Settings** dialog box, for the **Periphery to Core Placement and Routing Optimization** option, select one of the following options depending on how you want to direct periphery to core optimizations in your design:
 - a. Select **Auto** to direct the software to automatically identify transfers with tight timing windows, place the core registers, and route all connections to or from the periphery.
 - b. Select **On** to direct the software to globally optimize all transfers between the periphery and core registers, regardless of timing requirements.

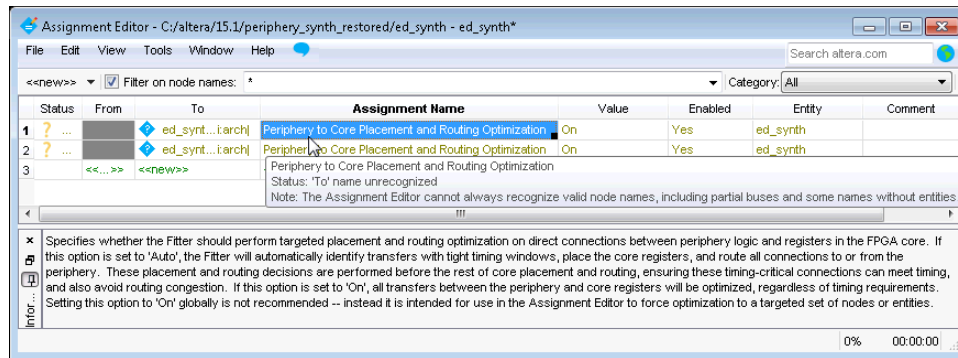
Note: Setting this option to **On** in the **Advanced Fitter Settings** is not recommended. The intended use for this setting is in the Assignment Editor to force optimization for a targeted set of nodes or instance.
 - c. Select **Off** to disable periphery to core path optimization in your design.

12.7.2 Setting Periphery to Core Optimizations in the Assignment Editor

When you turn on the **Periphery to Core Placement and Routing Optimization (P2C/C2P)** setting in the Assignment Editor, the Intel Quartus Prime software performs periphery to core, or core to periphery optimizations on selected instances in your design.

You can optionally perform periphery to core optimizations by instance with settings in the **Advanced Fitter Settings** dialog box.

1. In the Intel Quartus Prime software, click **Assignments** > **Assignment Editor**.
2. For the selected path, double-click the **Assignment Name** column, and then click the **Periphery to core register placement and routing optimization** option in the drop-down list.
3. In the **To** column, choose either a periphery node or core register node on a P2C/C2P path you want to optimize. Leave the **From** column empty. For paths to appear in the Assignments Editor, you must first run Analysis & Synthesis on your design.



12.7.3 Viewing Periphery to Core Optimizations in the Fitter Report

The Intel Quartus Prime software generates a periphery to core placement and routing optimization summary in the **Fitter (Place & Route)** report after compilation.

1. Compile your Intel Quartus Prime project.
2. In the **Tasks** pane, select **Compilation**.
3. Under **Fitter (Place & Route)**, double-click **View Report**.
4. In the **Fitter** folder, expand the **Place Stage** folder.
5. Double-click **Periphery to Core Transfer Optimization Summary**.

Table 34. Fitter Report - Periphery to Core Transfer Optimization (P2C) Summary

From Path	To Path	Status
Node 1	Node 2	Placed and Routed —Core register is locked. Periphery to core/core to periphery routing is committed.
Node 3	Node 4	Placed but not Routed —Core register is locked. Routing is not committed. This occurs when P2C is not able to optimize all targeted paths within a single group, for example, the same delay/wire requirement, or the same control signals. Partial P2C routing commitments may cause unresolvable routing congestion.
Node 5	Node 6	Not Optimized —This occurs when P2C is set to Auto and the path is not optimized due to one of the following issues: <ol style="list-style-type: none"> a. The delay requirement is impossible to achieve. b. The minimum delay requirement (for hold timing) is too large. The P2C algorithm cannot efficiently handle cases when many wires need to be added to meet hold timing. c. P2C encountered unresolvable routing congestion for this particular path.

Periphery to Core Transfer Optimization Summary			
	From	To	Status
1	Periphery to Core Transfer		
2	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(1)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_single_port_afi_rdata_valid_regs_srs_out[0]	Placed but Not Routed
3	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(2)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[63]	Placed but Not Routed
4	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(1)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[11]	Placed but Not Routed
5	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[7]	Placed but Not Routed
6	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[3]	Placed but Not Routed
7	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[0]	Placed but Not Routed
8	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[79]	Placed but Not Routed
9	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[75]	Placed but Not Routed
10	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(0)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[73]	Placed but Not Routed
11	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[71]	Placed but Not Routed
12	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(2)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[69]	Placed but Not Routed
13	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(1)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[35]	Placed but Not Routed
14	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[31]	Placed but Not Routed
15	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(3)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[27]	Placed but Not Routed
16	dutjarchlarch_instjio_tiles_wrap_instjio_tiles_insttile_gen(0)lane_gen(2)lane_inst	dutjarchlarch_instjio_if_instjio_sp_bidir_data.mem_dq_afi_regs_srs_out[23]	Placed but Not Routed



12.8 Scripting Support

You can run procedures and make settings described in this manual in a Tcl script. You can also run procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> -to <instance name>
```

Note: If the <value> field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

Related Links

- [Tcl Scripting](#) on page 81
- [Intel Quartus Prime Settings Reference File Manual](#)
- [Command Line Scripting](#) on page 67

12.8.1 Initial Compilation Settings

Use the Intel Quartus Prime Settings File (.qsf) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

The top table lists the .qsf variable name and applicable values for the settings described in the *Initial Compilation: Required Settings* section in the *Design Optimization Overview* chapter. The bottom table lists the advanced compilation settings.

Table 35. Initial Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING	ON, OFF	Global
Optimize Hold Timing	OPTIMIZE_HOLD_TIMING	OFF, IO PATHS AND MINIMUM TPD PATHS, ALL PATHS	Global

Table 36. Advanced Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Router Timing Optimization level	ROUTER_TIMING_OPTIMIZATION_LEVEL	NORMAL, MINIMUM, MAXIMUM	Global



Related Links

Design Optimization Overview on page 184

12.8.2 Resource Utilization Optimization Techniques

This table lists the .qsf file variable name and applicable values for Resource Utilization Optimization settings.

Table 37. Resource Utilization Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers ⁽¹⁾	QII_AUTO_PACKED_REGISTERS	AUTO, OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, SPARSE, SPARSE AUTO	Global, Instance
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Perform Physical Synthesis for Combinational Logic for Reducing Area (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global, Instance
Perform Physical Synthesis for Mapping Logic to Memory (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA	ON, OFF	Global, Instance
Optimization Technique	<device family name>_OPTIMIZATION_TECHNIQUE	AREA, SPEED, BALANCED	Global, Instance
Speed Optimization Technique for Clock Domains	SYNTH_CRITICAL_CLOCK	ON, OFF	Instance
State Machine Encoding	STATE_MACHINE_PROCESSING	AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER-ENCODE	Global, Instance
Auto RAM Replacement	AUTO_RAM_RECOGNITION	ON, OFF	Global, Instance
Auto ROM Replacement	AUTO_ROM_RECOGNITION	ON, OFF	Global, Instance
Auto Shift Register Replacement	AUTO_SHIFT_REGISTER_RECOGNITION	ON, OFF	Global, Instance
Auto Block Replacement	AUTO_DSP_RECOGNITION	ON, OFF	Global, Instance
Number of Processors for Parallel Compilation	NUM_PARALLEL_PROCESSORS	Integer between 1 and 16 inclusive, or ALL	Global

⁽¹⁾ Allowed values for this setting depend on the device family that you select.



12.8.3 I/O Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the I/O timing optimization settings.

Table 38. I/O Timing Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Optimize IOC Register Placement For Timing	OPTIMIZE_IOC_REGISTER_PLACEMENT_FOR_TIMING	ON, OFF	Global
Fast Input Register	FAST_INPUT_REGISTER	ON, OFF	Instance
Fast Output Register	FAST_OUTPUT_REGISTER	ON, OFF	Instance
Fast Output Enable Register	FAST_OUTPUT_ENABLE_REGISTER	ON, OFF	Instance
Fast OCT Register	FAST_OCT_REGISTER	ON, OFF	Instance

12.8.4 Register-to-Register Timing Optimization Techniques

The table lists the .qsf file variable name and applicable values for the settings described in *Register-to-Register Timing Optimization Techniques*.

Table 39. Register-to-Register Timing Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Perform Physical Synthesis for Combinational Logic (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC	ON, OFF	Global, Instance
Perform Register Duplication (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION	ON, OFF	Global, Instance
Perform Register Retiming (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_RETIMING	ON, OFF	Global, Instance
Perform Automatic Asynchronous Signal Pipelining (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING	ON, OFF	Global, Instance
Physical Synthesis Effort (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_EFFORT	NORMAL, EXTRA, FAST	Global
Fitter Seed	SEED	<integer>	Global
Maximum Fan-Out	MAX_FANOUT	<integer>	Instance
Manual Logic Duplication	DUPLICATE_ATOM	<node name>	Instance
Optimize Power during Synthesis	OPTIMIZE_POWER_DURING_SYNTHESIS	NORMAL, OFF, EXTRA_EFFORT	Global
Optimize Power during Fitting	OPTIMIZE_POWER_DURING_FITTING	NORMAL, OFF, EXTRA_EFFORT	Global



Related Links

Register-to-Register Timing Optimization Techniques on page 229

12.9 Document Revision History

Table 40. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> Moved Topic: Design Evaluation for Timing Closure after Initial Compilation: Optional Fitter Settings. Updated logic options about resource utilization optimization settings.
2017.05.08	17.0.0	<ul style="list-style-type: none"> Added topic: <i>Critical Paths</i>. Updated <i>Register-to-Register Timing</i> and renamed to <i>Register-to-Register Timing Analysis</i>. Renamed topic: <i>Timing Analysis with the Timing Analyzer to Displaying Path Reports with the Timing Analyzer</i>. Removed (LUT-Based Devices) remark from topic titles. Renamed topic: <i>Optimizing Timing (LUT-Based Devices)</i> to <i>Timing Optimization</i>. Renamed topic: <i>Debugging Timing Failures in the Timing Analyzer to Displaying Timing Closure Recommendations for Failing Paths</i>. Renamed topic: <i>Improving Register-to-Register Timing Summary to Improving Register-to-Register Timing</i>.
2016.05.02	16.0.0	<ul style="list-style-type: none"> Stated limitations about deprecated physical synthesis options. Added information about monitoring clustering difficulty.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Added: <i>Periphery to Core Register Placement and Routing Optimization</i>. Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. Updated DSE II content.
June 2014	14.0.0	<ul style="list-style-type: none"> Dita conversion. Removed content about obsolete devices that are no longer supported in QII software v14.0: Arria GX, Arria II, Cyclone III, Stratix II, Stratix III. Replaced Megafunction content with IP core content.
November 2013	13.1.0	<ul style="list-style-type: none"> Added Design Evaluation for Timing Closure section. Removed Optimizing Timing (Macrocell-Based CPLDs) section. Updated Optimize Multi-Corner Timing and Fitter Aggressive Routability Optimization. Updated Timing Analysis with the Timing Analyzer to show how to access the Report All Summaries command. Updated Ignored Timing Constraints to include a help link to <i>Fitter Summary Reports</i> with the Ignored Assignment Report information.
<i>continued...</i>		



Date	Version	Changes
May 2013	13.0.0	<ul style="list-style-type: none"> Renamed chapter title from Area and Timing Optimization to Timing Closure and Optimization. Removed design and area/resources optimization information. Added the following sections: Fitter Aggressive Routability Optimization. Tips for Analyzing Paths from/to the Source and Destination of Critical Path. Tips for Locating Multiple Paths to the Chip Planner. Tips for Creating a .tcl Script to Monitor Critical Paths Across Compiles.
November 2012	12.1.0	<ul style="list-style-type: none"> Updated "Initial Compilation: Optional Fitter Settings" on page 13-2, "I/O Assignments" on page 13-2, "Initial Compilation: Optional Fitter Settings" on page 13-2, "Resource Utilization" on page 13-9, "Routing" on page 13-21, and "Resolving Resource Utilization Problems" on page 13-43.
June 2012	12.0.0	<ul style="list-style-type: none"> Updated "Optimize Multi-Corner Timing" on page 13-6, "Resource Utilization" on page 13-10, "Timing Analysis with the Timing Analyzer" on page 13-12, "Using the Resource Optimization Advisor" on page 13-15, "Increase Placement Effort Multiplier" on page 13-22, "Increase Router Effort Multiplier" on page 13-22 and "Debugging Timing Failures in the Timing Analyzer" on page 13-24. Minor text edits throughout the chapter.
November 2011	11.1.0	<ul style="list-style-type: none"> Updated the "Timing Requirement Settings", "Standard Fit", "Fast Fit", "Optimize Multi-Corner Timing", "Timing Analysis with the Timing Analyzer", "Debugging Timing Failures in the Timing Analyzer", "Logic Lock (Standard) Assignments", "Tips for Analyzing Failing Clock Paths that Cross Clock Domains", "Flatten the Hierarchy During Synthesis", "Fast Input, Output, and Output Enable Registers", and "Hierarchy Assignments" sections Updated Table 13-6 Added the "Spine Clock Limitations" section Removed the Change State Machine Encoding section from page 19 Removed Figure 13-5 Minor text edits throughout the chapter
May 2011	11.0.0	<ul style="list-style-type: none"> Reorganized sections in "Initial Compilation: Optional Fitter Settings" section Added new information to "Resource Utilization" section Added new information to "Duplicate Logic for Fan-Out Control" section Added links to Help Additional edits and updates throughout chapter
December 2010	10.1.0	<ul style="list-style-type: none"> Added links to Help Updated device support Added "Debugging Timing Failures in the Timing Analyzer" section Removed Classic Timing Analyzer references Other updates throughout chapter
August 2010	10.0.1	Corrected link
July 2010	10.0.0	<ul style="list-style-type: none"> Moved Compilation Time Optimization Techniques section to new <i>Reducing Compilation Time</i> chapter Removed references to Timing Closure Floorplan Moved Smart Compilation Setting and Early Timing Estimation sections to new <i>Reducing Compilation Time</i> chapter Added Other Optimization Resources section Removed outdated information



Date	Version	Changes
		<ul style="list-style-type: none">• Changed references to DSE chapter to Help links• Linked to Help where appropriate• Removed Referenced Documents section

Related Links

[Documentation Archive](#)

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



13 Power Optimization

The Intel Quartus Prime software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing your design's total power consumption using power-driven synthesis and power-driven place-and-route.

This chapter describes the power-driven compilation feature and flow in detail, as well as low power design techniques that can further reduce power consumption in your design. The techniques primarily target Arria, Stratix, and Cyclone series of devices. These devices utilize a low-k dielectric material that dramatically reduces dynamic power and improves performance. Arria series, Stratix IV, and Stratix V device families include efficient logic structures called adaptive logic modules (ALMs) that obtain maximum performance while minimizing power consumption. Cyclone device families offer the optimal blend of high performance and low power in a low-cost FPGA.

Intel provides the Intel Quartus Prime Power Analyzer to aid you during the design process by delivering fast and accurate estimations of power consumption. You can minimize power consumption, while taking advantage of the industry's leading FPGA performance, by using the tools and techniques described in this chapter.

Total FPGA power consumption is comprised of I/O power, core static power, and core dynamic power. This chapter focuses on design optimization options and techniques that help reduce core dynamic power and I/O power. In addition to these techniques, there are additional power optimization techniques available for specific devices. These techniques include:

- Programmable Power Technology
- Device Speed Grade Selection

Related Links

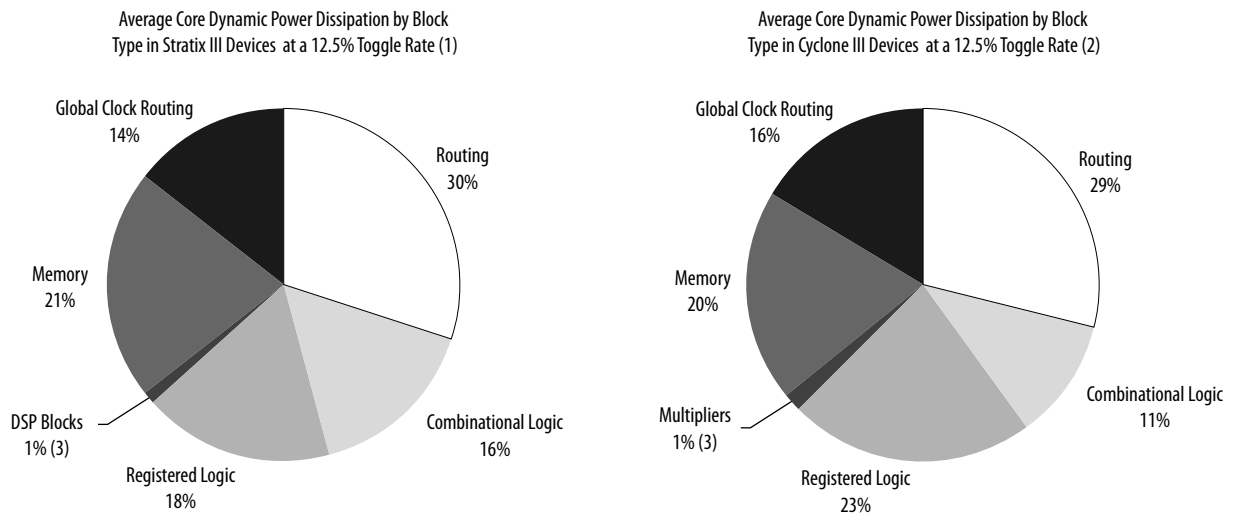
- [Literature and Technical Documentation](#)
- [Power Analysis](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*
- [AN 514: Power Optimization in Stratix IV FPGAs](#)

13.1 Power Dissipation

You can refine techniques that reduce power consumption in your design by understanding the sources of power dissipation.

The following figure shows the power dissipation of Stratix and Cyclone devices in different designs. All designs were analyzed at a fixed clock rate of 100 MHz and exhibited varied logic resource utilization across available resources.

Figure 84. Average Core Dynamic Power Dissipation



Notes:

1. 103 different designs were used to obtain these results.
2. 96 different designs were used to obtain these results.
3. In designs using DSP blocks, DSPs consumed 5% of core dynamic power.

In Stratix and Cyclone device families, a series of column and row interconnect wires of varying lengths provide signal interconnections between logic array blocks (LABs), memory block structures, and digital signal processing (DSP) blocks or multiplier blocks. These interconnects dissipate the largest component of device power.

FPGA combinational logic is another source of power consumption. The basic building block of logic in the latest Stratix series devices is the ALM, and in Cyclone IV GX devices, it is the logic element (LE).

For more information about ALMs and LEs in Cyclone or Stratix devices, refer to the respective device handbook.

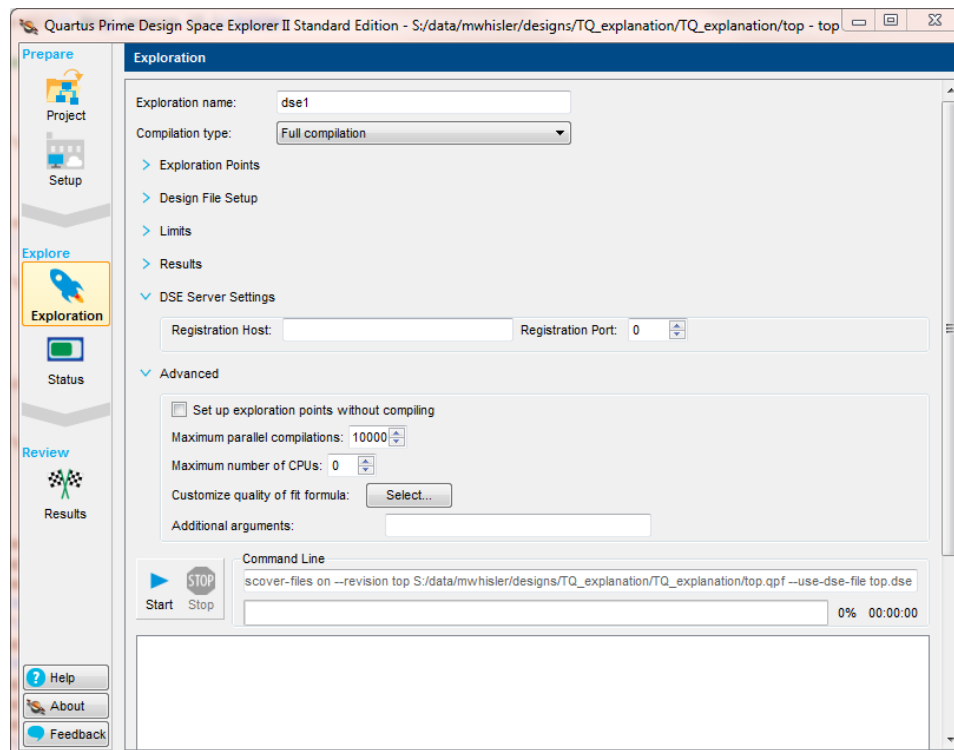
Memory and clock resources are other major consumers of power in FPGAs. Stratix devices feature the TriMatrix memory architecture. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are configurable to support many features. Stratix IV TriMatrix on-chip memory is an enhancement based upon the Stratix II FPGA TriMatrix memory and includes three sizes of memory blocks: MLAB blocks, M9K blocks, and M144K blocks. Stratix IV and Stratix V devices feature Programmable Power Technology, an advanced architecture that enables a smooth trade-off between speed and power. The core of each Stratix IV and Stratix V device is divided into tiles, each of which may be put into a high-speed or low-power mode. The primary benefit of Programmable Power Technology is to reduce static power, with a secondary benefit being a small reduction in dynamic power. Cyclone IV GX devices have 9-Kbit M9K memory blocks.



13.2 Design Space Explorer II

Design Space Explorer II (DSE) is an easy-to-use, self-guided design optimization utility that is included in the Intel Quartus Prime software. DSE II explores and reports optimal Intel Quartus Prime software options for your design, targeting either power optimization, design performance, or area utilization improvements. You can use DSE II to implement the techniques described in this chapter.

Figure 85. Design Space Explorer II User Interface



The power optimizations, under **Exploration mode**, target overall design power improvements. These settings focus on applying different options that specifically reduce total design thermal power.

By default, the Intel Quartus Prime Power Analyzer is run for every exploration performed by DSE II when power optimizations are selected. This helps you debug your design and determine trade-offs between power requirements and performance optimization.

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

- Name your DSE II session and specify the type of compilation to perform.
- Set **Exploration Points** and specify Exploration mode and the number and types of **Seeds** to use.
- Specify the **Design File Setup** including the use of a specified Quartus Archive File (.qar) or create a new one.
- Specify **Limits** to the operation of DSE II.
- Specify the type of **Results** to save.
- When using a remote compute farm, DSE II uses the values in the **DSE Server Settings** box to specify a registration host and network ports to connect.
- Options in the **Advanced** settings allow you to specify options such as:
 - Turn on the option to specify exploration points without compiling.
 - Specify the **Maximum number of parallel compilations** used by DSE II.
 - Specify the **Maximum number of CPUs** that can be used by DSE II.
 - Specify a quality of fit formula.

When you have completed your configuration, you can perform an exploration by clicking **Start**.

Related Links

- [Optimizing with Design Space Explorer II](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*
- [Launch Design Space Explorer Command \(Tools Menu\)](#)
in Intel Quartus Prime Help

13.3 Power-Driven Compilation

The standard Intel Quartus Prime compilation flow consists of Analysis and Synthesis, placement and routing, Assembly, and Timing Analysis. Power-driven compilation takes place at the Analysis and Synthesis and Place-and-Route stages.

Intel Quartus Prime software settings that control power-driven compilation are located in the **Power optimization during synthesis** list in the **Advanced Settings (Synthesis)** dialog box, and the **Power optimization during fitting** list on the **Advanced Fitter Settings** dialog box. The following sections describes these power optimization options at the Analysis and Synthesis and Fitter levels.

13.3.1 Power-Driven Synthesis

Synthesis netlist optimization occurs during the synthesis stage of the compilation flow. The optimization technique modifies the synthesis netlist to optimize your design according to the selection of area, speed, or power optimization. This section describes power optimization techniques at the synthesis level.

To access the Power Optimization During Synthesis option, click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**.

You can apply these settings on a project or entity level.



Table 41. Optimize Power During Synthesis Options

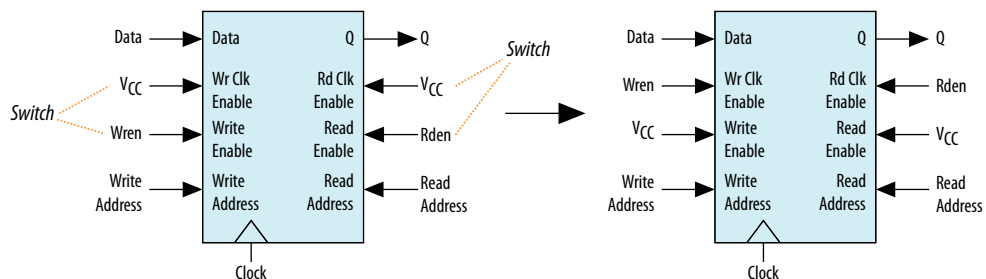
Settings	Description
Off	No netlist, placement, or routing optimizations are performed to minimize power.
Normal compilation (Default)	Low compute effort algorithms are applied to minimize power through netlist optimizations as long as they are not expected to reduce design performance.
Extra effort	High compute effort algorithms are applied to minimize power through netlist optimizations. Max performance might be impacted.

The **Normal compilation** setting is turned on by default. This setting performs memory optimization and power-aware logic mapping during synthesis.

Memory blocks can represent a large fraction of total design dynamic power. Minimizing the number of memory blocks accessed during each clock cycle can significantly reduce memory power. Memory optimization involves effective movement of user-defined read/write enable signals to associated read-and-write clock enable signals for all memory types.

A default implementation of a simple dual-port memory block in which write-clock enable signals and read-clock enable signals are connected to V_{CC}, making both read and write memory ports active during each clock cycle.

Figure 86. Memory Transformation



Memory transformation effectively moves the read-enable and write-enable signals to the respective read-clock enable and write-clock enable signals. By using this technique, memory ports are shut down when they are not accessed. This significantly reduces your design's memory power consumption. For Stratix IV and Stratix V devices, the memory transformation takes place at the Fitter level by selecting the **Normal compilation** settings for the power optimization option.

In Cyclone IV GX and StratixIV devices, the read-during-write behavior can significantly impact the power of single-port and bidirectional dual-port RAMs. It is best to set the read-during-write parameter to "Don't care" (at the HDL level), as it allows an optimization whereby the `read-enable` signal can be set to the inversion of the existing `write-enable` signal (if one exists). This allows the core of the RAM to shut down, which prevents switching, saving a significant amount of power.

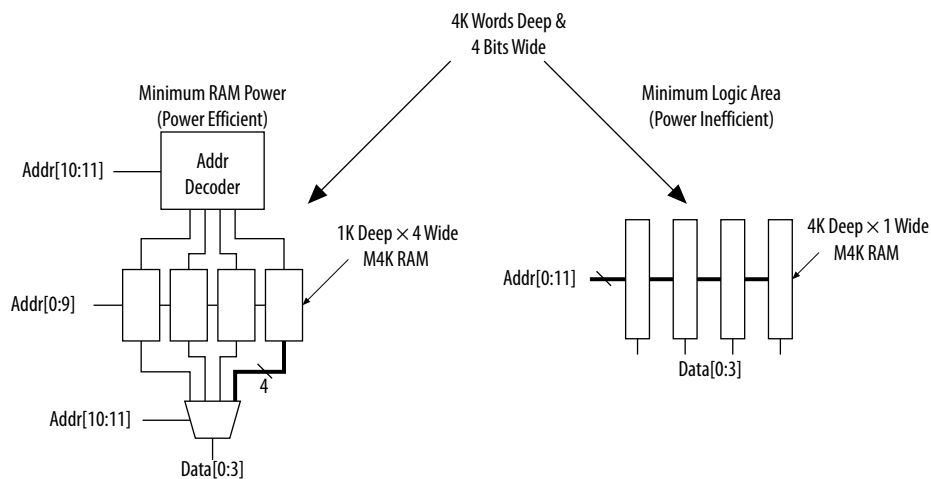
The other type of power optimization that takes place with the **Normal compilation** setting is power-aware logic mapping. The power-aware logic mapping reduces power by rearranging the logic during synthesis to eliminate nets with high switching rates.

The **Extra effort** setting performs the functions of the **Normal compilation** setting and other memory optimizations to further reduce memory power by shutting down memory blocks that are not accessed. This level of memory optimization can require extra logic, which can reduce design performance.

The **Extra effort** setting also performs power-aware memory balancing. Power-aware memory balancing automatically chooses the best memory configuration for your memory implementation and provides optimal power saving by determining the number of memory blocks, decoder, and multiplexer circuits required. If you have not previously specified target-embedded memory blocks for your design's memory functions, the power-aware balancer automatically selects them during memory implementation.

The following figure is an example of a 4k × 4 (4k deep and 4 bits wide) memory implementation in two different configurations using M4K memory blocks available in some Stratix devices.

Figure 87. 4K × 4 Memory Implementation Using Multiple M4K Blocks



The minimum logic area implementation uses M4K blocks configured as 4k × 1. This implementation is the default in the Intel Quartus Prime software because it has the minimum logic area (0 logic cells) and the highest speed. However, all four M4K blocks are active on each memory access in this implementation, which increases RAM power. The minimum RAM power implementation is created by selecting **Extra effort** in the **Power optimization** list. This implementation automatically uses four M4K blocks configured as 1k × 4 for optimal power saving. An address decoder is implemented by the RAM megafunction to select which of the four M4K blocks should be activated on a given cycle, based on the state of the top two user address bits. The RAM megafunction automatically implements a multiplexer to feed the downstream logic by choosing the appropriate M4K output. This implementation reduces RAM power because only one M4K block is active on any cycle, but it requires extra logic cells, costing logic area and potentially impacting design performance.

There is a trade-off between power saved by accessing fewer memories and power consumed by the extra decoder and multiplexer logic. The Intel Quartus Prime software automatically balances the power savings against the costs to choose the lowest power configuration for each logical RAM. The benchmark data shows that the power-driven synthesis can reduce memory power consumption by as much as 60% in Stratix devices.



Memory optimization options can also be controlled by the `Low_Power_Mode` parameter in the **Default Parameters** page of the **Settings** dialog box. The settings for this parameter are **None**, **Auto**, and **ALL**. **None** corresponds to the **Off** setting in the **Power optimization** list. **Auto** corresponds to the **Normal compilation** setting and **ALL** corresponds to the **Extra effort** setting, respectively. You can apply power optimization either on a compiler basis or on individual entities. The `Low_Power_Mode` parameter always takes precedence over the **Optimize Power for Synthesis** option for power optimization on memory.

You can also set the `MAXIMUM_DEPTH` parameter manually to configure the memory for low power optimization. This technique is the same as the power-aware memory balancer, but it is manual rather than automatic like the **Extra effort** setting in the **Power optimization** list. You can set the `MAXIMUM_DEPTH` parameter for memory modules manually in the megafunction instantiation or in the IP Catalog for power optimization. The `MAXIMUM_DEPTH` parameter always takes precedence over the **Optimize Power for Synthesis** options for power optimization on memory optimization.

Related Links

[Reducing Memory Power Consumption](#) on page 257

13.3.2 Power-Driven Fitter

The **Compiler Settings** page provides access to **Power optimization** settings.

You can apply these settings only on a project-wide basis. The **Extra effort** setting for the Fitter requires extensive effort to optimize the design for power and can increase the compilation time.

Table 42. Power-Driven Fitter Option

Settings	Description
Off	No netlist, placement, or routing optimizations are performed to minimize power.
Normal compilation (Default)	Low compute effort algorithms are applied to minimize power through placement and routing optimizations as long as they are not expected to reduce design performance.
Extra effort	High compute effort algorithms are applied to minimize power through placement and routing optimizations. Max performance might be impacted.

The **Normal compilation** setting is selected by default and performs DSP optimization by creating power-efficient DSP block configurations for your DSP functions. For Stratix IV and Stratix V devices, this setting, which is based on timing constraints entered for the design, enables the Programmable Power Technology to configure tiles as high-speed mode or low-power mode. Programmable Power Technology is always turned **ON** even when the **OFF** setting is selected for the **Power optimization** option. Tiles are the combination of LAB and MLAB pairs (including the adjacent routing associated with LAB and MLAB), which can be configured to operate in high-speed or low-power mode. This level of power optimization does not have any affect on the fitting, timing results, or compile time.

The **Extra effort** setting performs the functions of the **Normal compilation** setting and other place-and-route optimizations during fitting to fully optimize the design for power. The Fitter applies an extra effort to minimize power even after timing

requirements have been met by effectively moving the logic closer during placement to localize high-toggling nets, and using routes with low capacitance. However, this effort can increase the compilation time.

The **Extra effort** setting uses a Value Change Dump File (.vcd) that guides the Fitter to fully optimize the design for power, based on the signal activity of the design. The best power optimization during fitting results from using the most accurate signal activity information. If you do not have a .vcd file, the Intel Quartus Prime software uses assignments, clock assignments, and vectorless estimation values (Power Analyzer Tool settings) to estimate the signal activities. This information is used to optimize your design for power during fitting. The benchmark data shows that the power-driven Fitter technique can reduce power consumption by as much as 19% in Stratix devices. On average, you can reduce core dynamic power by 16% with the Extra effort synthesis and Extra effort fitting settings, as compared to the **Off** settings in both synthesis and Fitter options for power-driven compilation.

Note:

Only the **Extra effort** setting in the **power optimization** list for the Fitter option uses the signal activities (from .vcd files) during fitting. The settings made in the **Power Analyzer Settings** page in the **Settings** dialog box are used to calculate the signal activity of your design.

Related Links

- [AN 514: Power Optimization in Stratix IV FPGAs](#)
- [Power Analysis](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 3*

13.3.3 Area-Driven Synthesis

Using area optimization rather than timing or delay optimization during synthesis saves power because you use fewer logic blocks. Using less logic usually means less switching activity.

The Intel Quartus Prime software provides **Speed**, **Balanced**, or **Area** for the **Optimization Technique** technique option. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families). The **Speed Optimization Technique** can increase the resource usage of your design if the constraints are too aggressive, and can also result in increased power consumption.

The benchmark data shows that the area-driven technique can reduce power consumption by as much as 31% in Stratix devices and as much as 15% in Cyclone devices.

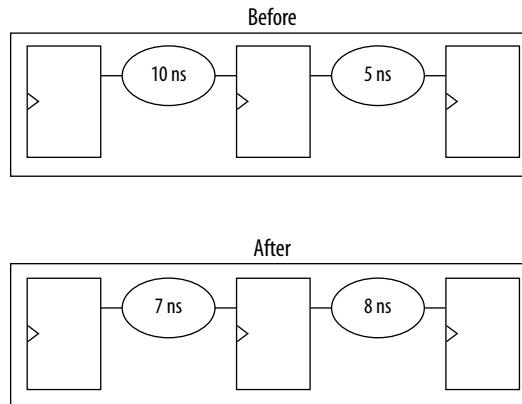
13.3.4 Gate-Level Register Retiming

You can also use gate-level register retiming to reduce circuit switching activity. Retiming shuffles registers across combinational blocks without changing design functionality.

The **Perform gate-level register retiming** option in the Intel Quartus Prime software enables the movement of registers across combinational logic to balance timing, allowing the software to trade off the delay between critical and noncritical paths.

Retiming uses fewer registers than pipelining. In this example of gate-level register retiming, the 10 ns critical delay is reduced by moving the register relative to the combinational logic, resulting in the reduction of data depth and switching activity.

Figure 88. Gate-Level Register Retiming



Gate-level register retiming makes changes at the gate level. If you are using an atom netlist from a third-party synthesis tool, you must also select the **Perform WYSIWYG primitive resynthesis** option to undo the atom primitives to gates mapping (so that register retiming can be performed), and then to remap gates to Intel primitives.

When using Intel Quartus Prime integrated synthesis, retiming occurs during synthesis before the design is mapped to Intel primitives. The benchmark data shows that the combination of WYSIWYG remapping and gate-level register retiming techniques can reduce power consumption by as much as 6% in Stratix devices and as much as 21% in Cyclone devices.

Related Links

[Netlist Optimizations and Physical Synthesis](#) on page 306

13.4 Design Guidelines

Several low-power design techniques can reduce power consumption when applied during FPGA design implementation. This section provides detailed design techniques for Cyclone IV GX devices that affect overall design power. The results of these techniques might be different from design to design.

13.4.1 Clock Power Management

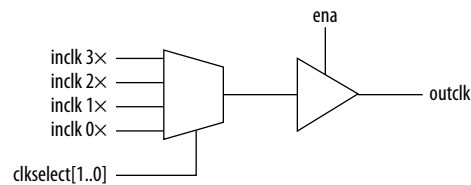
Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. Actual clock-related power consumption is higher, because the power consumption of a block includes local clock distribution within logic, memory, and DSP or multiplier blocks.

The Intel Quartus Prime software optimizes clock routing power automatically, enabling only those portions of the clock network that are necessary to feed downstream registers. Another technique for power reduction is gating clocks when the logic does not require them. Even though you can build clock-gating logic, this approach can generate clock glitches in FPGAs using ALMs or LEs.

Cyclone IV , Stratix IV, and Stratix V devices use clock control blocks that include an enable signal. A clock control block is a clock buffer that lets you dynamically enable or disable the clock network, and dynamically switch between multiple sources to drive the clock network. You can use the Intel Quartus Prime IP Catalog to create this clock control block with the ALTCLKCTRL IP core. Cyclone IV , Stratix IV, and Stratix V devices provide clock control blocks for global clock networks. In addition, Stratix IV, and Stratix V devices have clock control blocks for regional clock networks.

The dynamic clock enable feature lets internal logic control the clock network. With a powered down clock network you prevent switching of all the logic that the clock network feeds, thereby reducing the overall power consumption of the device. For example, the following shows a 4-input clock control block diagram.

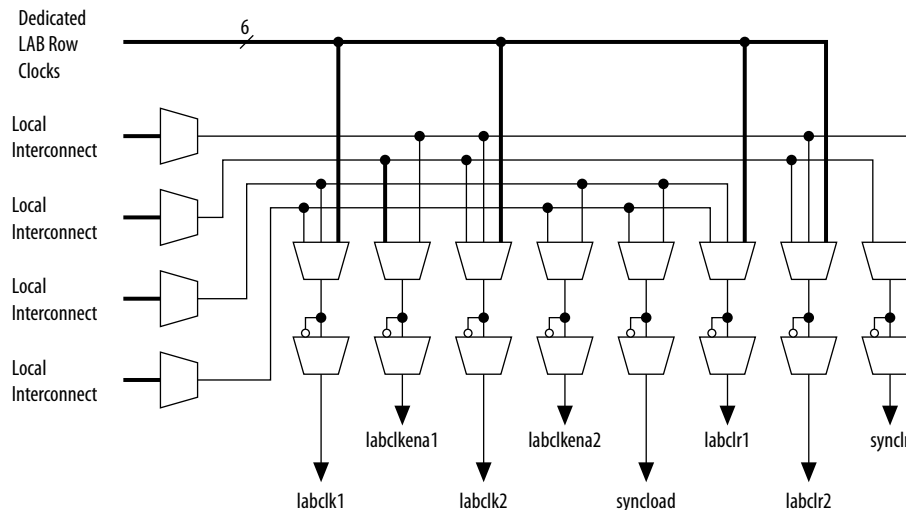
Figure 89. Clock Control Block Diagram



The enable signal is applied to the clock signal before being distributed to global routing. Therefore, the enable signal can either have a significant timing slack (at least as large as the global routing delay) or it can reduce the f_{MAX} of the clock signal.

Another contributor to clock power consumption is the LAB clock that distributes a clock to the registers within a LAB. LAB clock power can be the dominant contributor to overall clock power. For example, in Cyclone devices, each LAB can use two clocks and two clock enable signals, as shown in the following figure. Each LAB's clock signal and clock enable signal are linked. For example, an LE in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal.

Figure 90. LAB-Wide Control Signals



To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Intel Quartus Prime software automatically promotes register-level clock enable signals to the LAB-level. All registers within an LAB that share a common clock and clock enable are controlled by a shared gated clock. To take advantage of these clock enables, use a clock enable construct in the relevant HDL code for the registered logic.

Related Links

[Clock Control Block Megafunction User Guide \(ALTCLKCTRL\)](#)

13.4.1.1 LAB-Wide Clock Enable Example

This VHDL code makes use of a LAB-wide clock enable. This clock-gating logic is automatically turned into an LAB-level clock enable signal.

```

IF clk'event AND clock = '1' THEN
    IF logic_is_enabled = '1' THEN
        reg <= value;
    ELSE
        reg <= reg;
    END IF;
END IF;

```

13.4.1.2 Reducing Memory Power Consumption

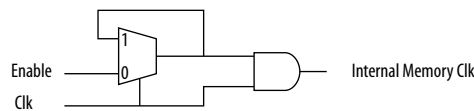
The memory blocks in FPGA devices can represent a large fraction of typical core dynamic power. Memory consumes approximately 20% of the core dynamic power in typical some device designs. Memory blocks are unlike most other blocks in the device because most of their power is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines.

When a memory block is clocked, there is a sequence of timed events that occur within the block to execute a read or write. The circuitry controlled by the clock consumes the same amount of power regardless of whether or not the address or data has changed from one cycle to the next. Thus, the toggle rate of input data and the address bus have no impact on memory power consumption.

The key to reducing memory power consumption is to reduce the number of memory clocking events. You can achieve this through clock network-wide gating, or on a per-memory basis through use of the clock enable signals on the memory ports.

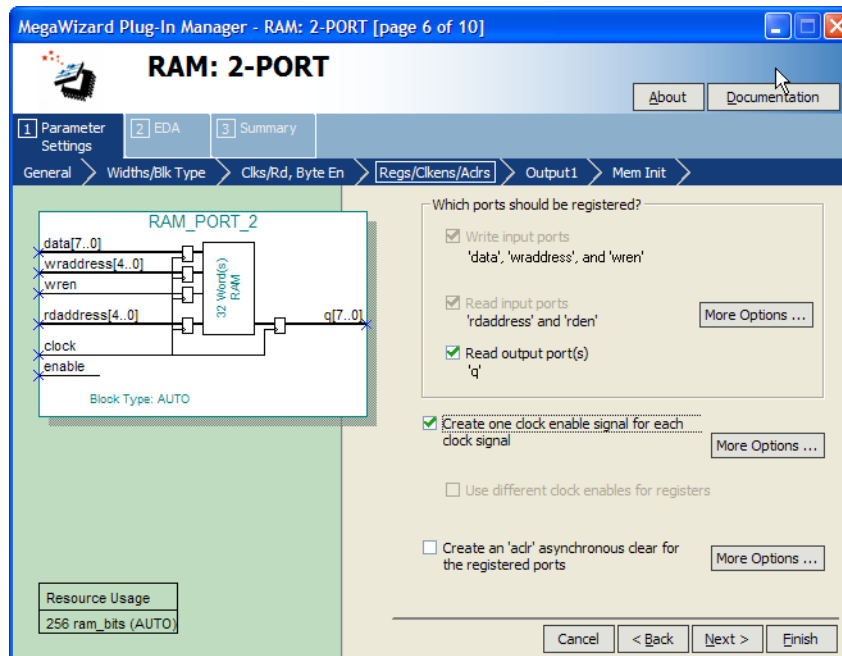
The logical view of the internal clock of the memory block. Use the appropriate enable signals on the memory to make use of the clock enable signal instead of gating the clock.

Figure 91. Memory Clock Enable Signal



Using the clock enable signal enables the memory only when necessary and shuts it down for the rest of the time, reducing the overall memory power consumption. You can create these enable signals by selecting the **Clock enable signal** option for the appropriate port when generating the memory block function.

Figure 92. RAM 2-Port Clock Enable Signal Selectable Option



For example, consider a design that contains a 32-bit-wide M4K memory block in ROM mode that is running at 200 MHz. Assuming that the output of this block is only required approximately every four cycles, this memory block consumes 8.45 mW of

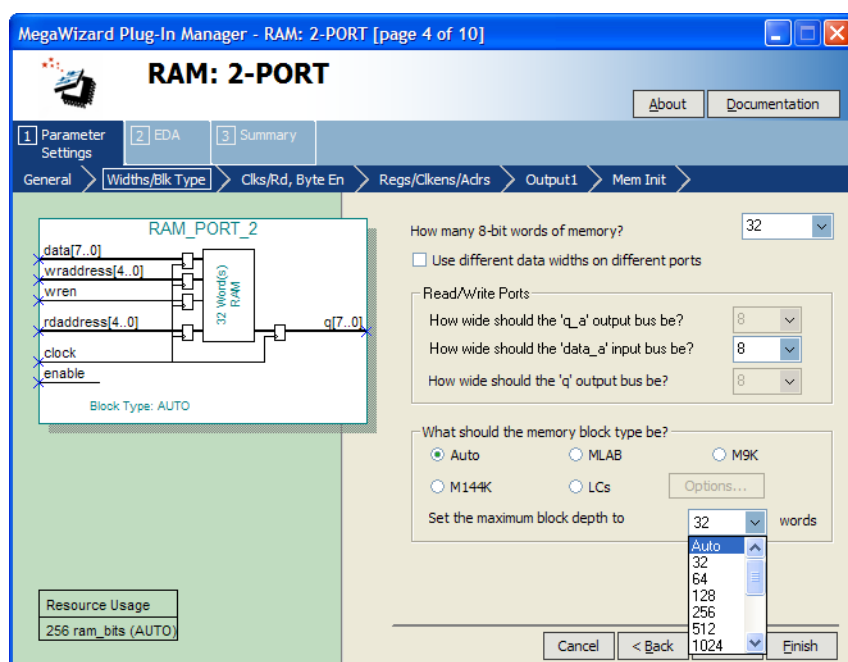


dynamic power according to the demands of the downstream logic. By adding a small amount of control logic to generate a read clock enable signal for the memory block only on the relevant cycles, the power can be cut 75% to 2.15 mW.

You can also use the `MAXIMUM_DEPTH` parameter in your memory megafunction to save power in Cyclone IV GX, Stratix IV, and Stratix V devices; however, this approach might increase the number of LEs required to implement the memory and affect design performance.

You can set the `MAXIMUM_DEPTH` parameter for memory modules manually in the megafunction instantiation. The Intel Quartus Prime software automatically chooses the best design memory configuration for optimal power.

Figure 93. RAM 2-Port Maximum Depth Selectable Option



Related Links

- [Power-Driven Compilation](#) on page 250
- [Clock Power Management](#) on page 255

13.4.1.3 Memory Power Reduction Example

Power usage measurements for a 4K × 36 simple dual-port memory implemented using multiple M4K blocks in a Stratix device. For each implementation, the M4K blocks are configured with a different memory depth.

Table 43. 4K × 36 Simple Dual-Port Memory Implemented Using Multiple M4K Blocks

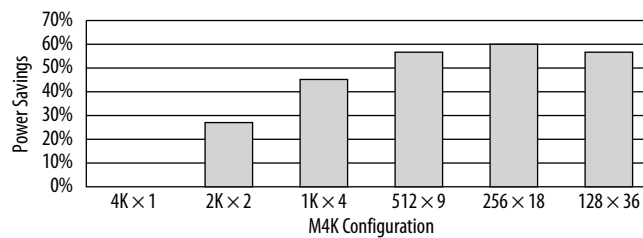
M4K Configuration	Number of M4K Blocks	ALUTs
4K × 1 (Default setting)	36	0
2K × 2	36	40

continued...

M4K Configuration	Number of M4K Blocks	ALUTs
1K × 4	36	62
512 × 9	32	143
256 × 18	32	302
128 × 36	32	633

Using the `MAXIMUM_DEPTH` parameter can save power. For all implementations, a user-provided read enable signal is present to indicate when read data is required. Using this power-saving technique can reduce power consumption by as much as 60%.

Figure 94. Power Savings Using the `MAXIMUM_DEPTH` Parameter



As the memory depth becomes more shallow, memory dynamic power decreases because unaddressed M4K blocks can be shut off using a decoded combination of address bits and the read enable signal. For a 128-deep memory block, power used by the extra LEs starts to outweigh the power gain achieved by using a more shallow memory block depth. The power consumption of the memory blocks and associated LEs depends on the memory configuration.

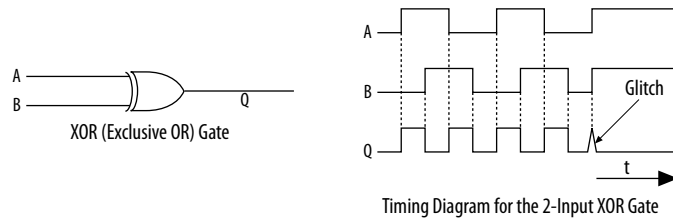
Note: The SOPC Builder and Platform Designer (Standard) system do not offer specific power savings control for on-chip memory block. There is no read enable, write enable, or clock enable that you can enable in the on-chip RAM megafunction to shut down the RAM block in the SOPC Builder and Platform Designer (Standard) system.

13.4.2 Pipelining and Retiming

Designs with many glitches consume more power because of faster switching activity. Glitches cause unnecessary and unpredictable temporary logic switches at the output of combinational logic. A glitch usually occurs when there is a mismatch in input signal timing leading to unequal propagation delay.

For example, consider an input change on one input of a 2-input XOR gate from 1 to 0, followed a few moments later by an input change from 0 to 1 on the other input. For a moment, both inputs become 1 (high) during the state transition, resulting in 0 (low) at the output of the XOR gate. Subsequently, when the second input transition takes place, the XOR gate output becomes 1 (high). During signal transition, a glitch is produced before the output becomes stable.

Figure 95. XOR Gate Showing Glitch at the Output

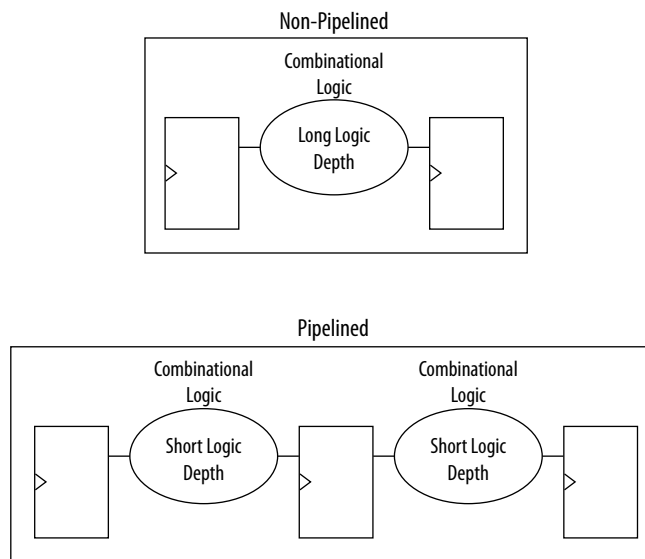


This glitch can propagate to subsequent logic and create unnecessary switching activity, increasing power consumption. Circuits with many XOR functions, such as arithmetic circuits or cyclic redundancy check (CRC) circuits, tend to have many glitches if there are several levels of combinational logic between registers.

Pipelining can reduce design glitches by inserting flipflops into long combinational paths. Flipflops do not allow glitches to propagate through combinational paths. Therefore, a pipelined circuit tends to have less glitching. Pipelining has the additional benefit of generally allowing higher clock speed operations, although it does increase the latency of a circuit (in terms of the number of clock cycles to a first result).

An example where pipelining is applied to break up a long combinational path.

Figure 96. Pipelining Example



Pipelining is very effective for glitch-prone arithmetic systems because it reduces switching activity, resulting in reduced power dissipation in combinational logic. Additionally, pipelining allows higher-speed operation by reducing logic-level numbers between registers. The disadvantage of this technique is that if there are not many glitches in your design, pipelining can increase power consumption by adding unnecessary registers. Pipelining can also increase resource utilization. The benchmark data shows that pipelining can reduce dynamic power consumption by as much as 30% in Cyclone and Stratix devices.

13.4.3 Architectural Optimization

You can use design-level architectural optimization by taking advantage of specific device architecture features. These features include dedicated memory and DSP or multiplier blocks available in FPGA devices to perform memory or arithmetic-related functions. You can use these blocks in place of LUTs to reduce power consumption. For example, you can build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

The Stratix device family allows you to efficiently target small, medium, and large memories with the TriMatrix memory architecture. Each TriMatrix memory block is optimized for a specific function. M512 memory blocks are more power-efficient than the distributed memory structures in some competing FPGAs. The M4K memory blocks are used to implement buffers for a wide variety of applications, including processor code storage, large look-up table implementation, and large memory applications. The M-RAM blocks are useful in applications where a large volume of data must be stored on-chip. Effective utilization of these memory blocks can have a significant impact on power reduction in your design.

The latest Stratix and Cyclone device families have configurable M9K memory blocks that provide various memory functions such as RAM, FIFO buffers, and ROM.

Related Links

[Timing Closure and Optimization](#) on page 201

13.4.4 I/O Power Guidelines

Nonterminated I/O standards such as LVTTTL and LVCMOS have a rail-to-rail output swing. The voltage difference between logic-high and logic-low signals at the output pin is equal to the V_{CCIO} supply voltage. If the capacitive loading at the output pin is known, the dynamic power consumed in the I/O buffer can be calculated.

$$P = 0.5 \times F \times C \times V^2$$

In this equation, F is the output transition frequency and C is the total load capacitance being switched. V is equal to V_{CCIO} supply voltage. Because of the quadratic dependence on V_{CCIO} , lower voltage standards consume significantly less dynamic power.

Transistor-to-transistor logic (TTL) I/O buffers consume very little static power. As a result, the total power consumed by a LVTTTL or LVCMOS output is highly dependent on load and switching frequency.

When using resistively terminated I/O standards like SSTL and HSTL, the output load voltage swings by a small amount around some bias point. The same dynamic power equation is used, where V is the actual load voltage swing. Because this is much smaller than V_{CCIO} , dynamic power is lower than for nonterminated I/O under similar conditions. These resistively terminated I/O standards dissipate significant static (frequency-independent) power, because the I/O buffer is constantly driving current into the resistive termination network. However, the lower dynamic power of these I/O standards means they often have lower total power than LVCMOS or LVTTTL for high-frequency applications. Use the lowest drive strength I/O setting that meets your speed and waveform requirements to minimize I/O power when using resistively terminated standards.



You can save a small amount of static power by connecting unused I/O banks to the lowest possible V_{CCIO} voltage of 1.2 V.

When calculating I/O power, the Power Analyzer uses the default capacitive load set for the I/O standard in the **Capacitive Loading** page of the **Device and Pin Options** dialog box. Any other components defined in the board trace model are not taken into account for the power measurement.

For Cyclone IV GX, Stratix IV, and Stratix V devices, Advanced I/O Timing is always used, which uses the full board trace model.

Related Links

- [Managing Device I/O Pins](#) on page 24
- [Stratix Series FPGA I/O Connectivity](#)
- [I/O Features in Stratix IV Devices](#)
- [I/O Features in Cyclone IV Devices](#)

13.4.5 Dynamically Controlled On-Chip Terminations

Stratix IV and Stratix V FPGAs offer dynamic on-chip termination (OCT). Dynamic OCT enables series termination (RS) and parallel termination (RT) to dynamically turn on/off during the data transfer. This feature is especially useful when Stratix IV and Stratix V FPGAs are used with external memory interfaces, such as interfacing with DDR memories.

Compared to conventional termination, dynamic OCT reduces power consumption significantly as it eliminates the constant DC power consumed by parallel termination when transmitting data. Parallel termination is extremely useful for applications that interface with external memories where I/O standards, such as HSTL and SSTL, are used. Parallel termination supports dynamic OCT, which is useful for bidirectional interfaces.

The following is an example of power saving for a DDR3 interface using on-chip parallel termination.

The static current consumed by parallel OCT is equal to the V_{CCIO} voltage divided by 100 Ω . For DDR3 interfaces that use SSTL-15, the static current is $1.5 \text{ V}/100 \Omega = 15 \text{ mA}$ per pin. Therefore, the static power is $1.5 \text{ V} \times 15 \text{ mA} = 22.5 \text{ mW}$. For an interface with 72 DQ and 18 DQS pins, the static power is $90 \text{ pins} \times 22.5 \text{ mW} = 2.025 \text{ W}$. Dynamic parallel OCT disables parallel termination during write operations, so if writing occurs 50% of the time, the power saved by dynamic parallel OCT is $50\% \times 2.025 \text{ W} = 1.0125 \text{ W}$.

For more information about dynamic OCT in Stratix IV devices, refer to the chapter in the *Stratix IV Device Handbook*.

Related Links

[Stratix IV Device I/O Features](#)
In *Stratix IV Device Handbook*

13.4.6 Power Optimization Advisor

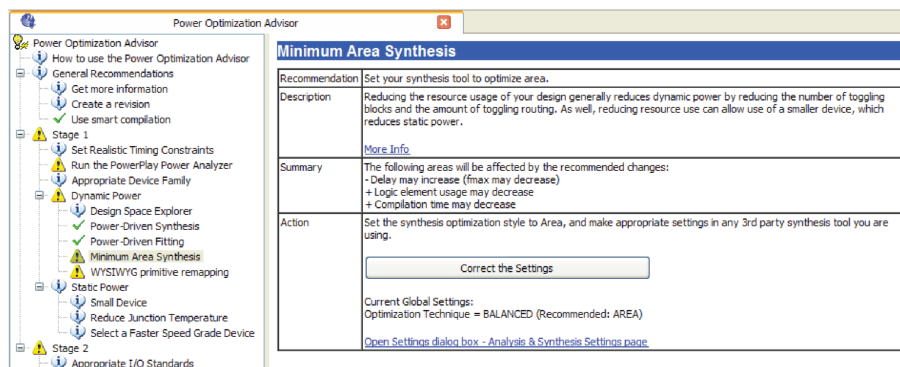
The Intel Quartus Prime software includes the Power Optimization Advisor, which provides specific power optimization advice and recommendations based on the current design project settings and assignments. The advisor covers many of the suggestions listed in this chapter. The following example shows how to reduce your design power with the Power Optimization Advisor.

13.4.6.1 Power Optimization Advisor Example

After compiling your design, run the Power Analyzer to determine your design power and to see where power is dissipated in your design. Based on this information, you can run the Power Optimization Advisor to implement recommendations that can reduce design power.

The Power Optimization Advisor after compiling a design that is not fully optimized for power.

Figure 97. Power Optimization Advisor

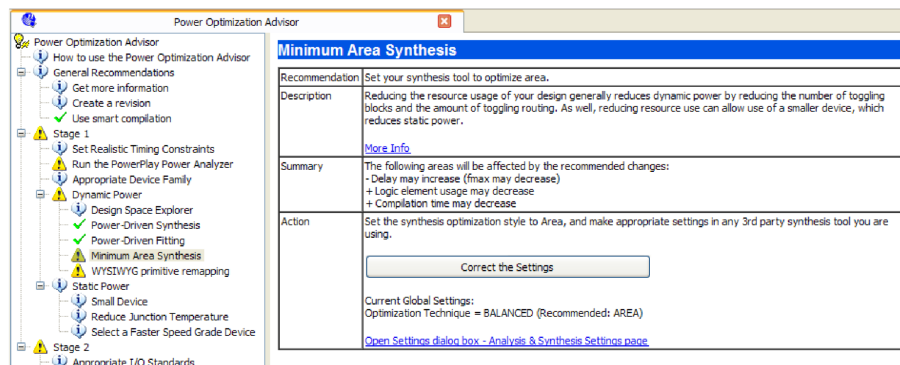


The Power Optimization Advisor shows the recommendations that can reduce power in your design. The recommendations are split into stages to show the order in which you should apply the recommended settings. The first stage shows mostly CAD setting options that are easy to implement and highly effective in reducing design power. An icon indicates whether each recommended setting is made in the current project. The checkmark icons for Stage 1 shows the recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icon shows the general suggestions. Each recommendation includes the description, summary of the effect of the recommendation, and the action required to make the appropriate setting.

There is a link from each recommendation to the appropriate location in the Intel Quartus Prime user interface where you can change the setting. After making the recommended changes, recompile your design. The Power Optimization Advisor indicates with green check marks that the recommendations were implemented successfully. You can use the Power Analyzer to verify your design power results.



Figure 98. Implementation of Power Optimization Advisor Recommendations



The recommendations listed in Stage 2 generally involve design changes, rather than CAD settings changes as in Stage 1. You can use these recommendations to further reduce your design power consumption. Intel recommends that you implement Stage 1 recommendations first, then the Stage 2 recommendations.

13.5 Document Revision History

Table 44. Document Revision History

Date	Version	Changes
2016.10.31	16.1.0	<ul style="list-style-type: none"> Removed statement of support for gate-level timing simulation.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> . <ul style="list-style-type: none"> Updated screenshot for DSE II GUI. Added information about remote hosts for DSE II.
2015.05.04	15.0.0	
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings. Updated DSE II GUI and optimization settings.
2014.06.30	14.0.0	Updated the format.
May 2013	13.0.0	Added a note to "Memory Power Reduction Example" on Qsys and SOPC Builder power savings limitation for on-chip memory block.
June 2012	12.0.0	Removed survey link.
November 2011	10.0.2	Template update.
December 2010	10.0.1	Template update.
July 2010	10.0.0	<ul style="list-style-type: none"> Was chapter 11 in the 9.1.0 release Updated Figures 14-2, 14-3, 14-6, 14-18, 14-19, and 14-20 Updated device support Minor editorial updates

continued...

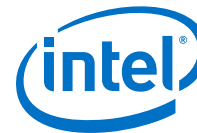


Date	Version	Changes
November 2009	9.1.0	<ul style="list-style-type: none">• Updated Figure 11-1 and associated references• Updated device support• Minor editorial update
March 2009	9.0.0	<ul style="list-style-type: none">• Was chapter 9 in the 8.1.0 release• Updated for the Quartus II software release• Added benchmark results• Removed several sections• Updated Figure 13-1, Figure 13-17, and Figure 13-18
November 2008	8.1.0	<ul style="list-style-type: none">• Changed to 8½" × 11" page size• Changed references to altsyncram to RAM• Minor editorial updates
May 2008	8.0.0	<ul style="list-style-type: none">• Added support for Stratix IV devices• Updated Table 9-1 and 9-9• Updated "Architectural Optimization" on page 9-22• Added "Dynamically-Controlled On-Chip Terminations" on page 9-26• Updated "Referenced Documents" on page 9-29• Updated references

Related Links

[Documentation Archive](#)

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



14 Area Optimization

This chapter describes techniques to reduce resource usage when designing for Intel devices.

14.1 Resource Utilization

Determining device utilization is important regardless of whether your design achieved a successful fit. If your compilation results in a no-fit error, resource utilization information is important for analyzing the fitting problems in your design. If your fitting is successful, review the resource utilization information to determine whether the future addition of extra logic or other design changes might introduce fitting difficulties. Also, review the resource utilization information to determine if it is impacting timing performance.

To determine resource usage, refer to the **Flow Summary** section of the Compilation Report. This section reports resource utilization, including pins, memory bits, digital signal processing (DSP) blocks, and phase-locked loops (PLLs). **Flow Summary** indicates whether your design exceeds the available device resources. More detailed information is available by viewing the reports under **Resource Section** in the **Fitter** section of the Compilation Report.

Flow Summary shows the overall logic utilization. The Fitter can spread logic throughout the device, which may lead to higher overall utilization.

As the device fills up, the Fitter automatically searches for logic functions with common inputs to place in one ALM. The number of packed registers also increases. Therefore, a design that has high overall utilization might still have space for extra logic if the logic and registers can be packed together more tightly.

The reports under the **Resource Section** in the **Fitter** section of the Compilation Report provide more detailed resource information. The Fitter Resource Usage Summary report breaks down the logic utilization information and provides other resource information, including the number of bits in each type of memory block. This panel also contains a summary of the usage of global clocks, PLLs, DSP blocks, and other device-specific resources.

You can also view reports describing some of the optimizations that occurred during compilation. For example, if you use Intel Quartus Prime synthesis, the reports in the Optimization Results folder in the **Analysis & Synthesis** section include information about registers removed during synthesis. Use this report to estimate device resource utilization for a partial design to ensure that registers were not removed due to missing connections with other parts of the design.

If a specific resource usage is reported as less than 100% and a successful fit cannot be achieved, either there are not enough routing resources or some assignments are illegal. In either case, a message appears in the **Processing** tab of the **Messages** window describing the problem.

If the Fitter finishes unsuccessfully and runs much faster than on similar designs, a resource might be over-utilized or there might be an illegal assignment. If the Intel Quartus Prime software seems to run for an excessively long time compared to runs on similar designs, a legal placement or route probably cannot be found. In the Compilation Report, look for errors and warnings that indicate these types of problems.

You can use the Chip Planner to find areas of the device that have routing congestion on specific types of routing resources. If you find areas with very high congestion, analyze the cause of the congestion. Issues such as high fan-out nets not using global resources, an improperly chosen optimization goal (speed versus area), very restrictive floorplan assignments, or the coding style can cause routing congestion. After you identify the cause, modify the source or settings to reduce routing congestion.

Related Links

- [Fitter Resources Reports](#)
In Intel Quartus Prime Help
- [Analyzing and Optimizing the Design Floorplan](#) on page 280

14.2 Optimizing Resource Utilization

The following lists the stages after design analysis:

1. Optimize resource utilization—Ensure that you have already set the basic constraints
2. I/O timing optimization—Optimize I/O timing after you optimize resource utilization and your design fits in the desired target device
3. Register-to-register timing optimization

Related Links

- [Design Optimization Overview](#) on page 184
- [Timing Closure and Optimization](#) on page 201

14.2.1 Using the Resource Optimization Advisor

The Resource Optimization Advisor provides guidance in determining settings that optimize resource usage. To run the Resource Optimization Advisor click **Tools > Advisors > Resource Optimization Advisor**.

The Resource Optimization Advisor provides step-by-step advice about how to optimize resource usage (logic element, memory block, DSP block, I/O, and routing) of your design. Some of the recommendations in these categories might conflict with each other. Intel recommends evaluating the options and choosing the settings that best suit your requirements.

Related Links

[Resource Optimization Advisor Command Tools Menu](#)
In Intel Quartus Prime Help



14.2.2 Resource Utilization Issues Overview

Resource utilization issues can be divided into three categories:

- Issues relating to *I/O pin utilization or placement*, including dedicated I/O blocks such as PLLs or LVDS transceivers.
- Issues relating to *logic utilization or placement*, including logic cells containing registers and LUTs as well as dedicated logic, such as memory blocks and DSP blocks.
- Issues relating to *routing*.

14.2.3 I/O Pin Utilization or Placement

Resolve I/O resource problems with these guidelines.

14.2.3.1 Guideline: Use I/O Assignment Analysis

To help with pin placement, click **Processing > Start > Start I/O Assignment Analysis**. The **Start I/O Assignment Analysis** command allows you to check your I/O assignments early in the design process. You can use this command to check the legality of pin assignments before, during, or after compilation of your design. If design files are available, you can use this command to accomplish more thorough legality checks on your design's I/O pins and surrounding logic. These checks include proper reference voltage pin usage, valid pin location assignments, and acceptable mixed I/O standards.

Common issues with I/O placement relate to the fact that differential standards have specific pin pairings and certain I/O standards might be supported only on certain I/O banks.

If your compilation or I/O assignment analysis results in specific errors relating to I/O pins, follow the recommendations in the error message. Right-click the message in the **Messages** window and click **Help** to open the Intel Quartus Prime Help topic for this message.

14.2.3.2 Guideline: Modify Pin Assignments or Choose a Larger Package

If a design that has pin assignments fails to fit, compile the design without the pin assignments to determine whether a fit is possible for the design in the specified device and package. You can use this approach if an Intel Quartus Prime error message indicates fitting problems due to pin assignments.

If the design fits when all pin assignments are ignored or when several pin assignments are ignored or moved, you might have to modify the pin assignments for the design or select a larger package.

If the design fails to fit because insufficient I/Os pins are available, a successful fit can often be obtained by using a larger device package (which can be the same device density) that has more available user I/O pins.

Related Links

[Managing Device I/O Pins](#) on page 24

14.2.4 Logic Utilization or Placement

Resolve logic resource problems, including logic cells containing registers and LUTs, as well as dedicated logic such as memory blocks and DSP blocks, with these guidelines.

14.2.4.1 Guideline: Optimize Source Code

If your design does not fit because of logic utilization, then evaluate and modify the design at the source. You can often improve logic significantly by making design-specific changes to your source code. This is typically the most effective technique for improving the quality of your results.

If your design does not fit into available logic elements (LEs) or ALMs, but you have unused memory or DSP blocks, check if you have code blocks in your design that describe memory or DSP functions that are not being inferred and placed in dedicated logic. You might be able to modify your source code to allow these functions to be placed into dedicated memory or DSP resources in the target device.

Ensure that your state machines are recognized as state machine logic and optimized appropriately in your synthesis tool. State machines that are recognized are generally optimized better than if the synthesis tool treats them as generic logic. In the Intel Quartus Prime software, you can check for the State Machine report under **Analysis & Synthesis** in the Compilation Report. This report provides details, including the state encoding for each state machine that was recognized during compilation. If your state machine is not being recognized, you might have to change your source code to enable it to be recognized.

Related Links

- [AN 584: Timing Closure Methodology for Advanced FPGA Designs](#)
- [Recommended HDL Coding Styles](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

14.2.4.2 Guideline: Optimize Synthesis for Area, Not Speed

If your design fails to fit because it uses too much logic, resynthesize the design to improve the area utilization. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Particularly when area utilization of the design is a concern, ensure that you do not over-constrain the timing requirements for the design. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.



If resource utilization is an important concern, you can optimize for area instead of speed.

- If you are using Intel Quartus Prime integrated synthesis, click **Assignments** ► **Settings** ► **Compiler Settings** ► **Advanced Settings (Synthesis)** and select **Balanced** or **Area** for the **Optimization Technique**.
- If you want to reduce area for specific modules in your design using the **Area** or **Speed** setting while leaving the default **Optimization Technique** setting at **Balanced**, use the Assignment Editor.
- You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.
- In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization.

Optimizing for area or speed can affect the register-to-register timing performance.

Note:

In the Intel Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.

The Intel Quartus Prime software provides additional attributes and options that can help improve the quality of your synthesis results.

Related Links

- [Intel Quartus Prime Integrated Synthesis](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*
- [Optimization Mode](#)
In Intel Quartus Prime Help

14.2.4.3 Guideline: Restructure Multiplexers

Multiplexers form a large portion of the logic utilization in many FPGA designs. By optimizing your multiplexed logic, you can achieve a more efficient implementation in your Intel device.

Related Links

- [Restructure Multiplexers logic option](#)
For more information about the Restructure Multiplexers option
- [Recommended HDL Coding Styles](#)
For design guidelines to achieve optimal resource utilization for multiplexer designs

14.2.4.4 Guideline: Perform WYSIWYG Primitive Resynthesis with Balanced or Area Setting

The **Perform WYSIWYG Primitive Resynthesis** logic option specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the **Optimization Technique** logic option. The **Perform WYSIWYG Primitive Resynthesis** logic option is useful for resynthesizing some or all of the WYSIWYG primitives in your design for better area or performance. However, WYSIWYG primitive resynthesis can be done only when you use third-party synthesis tools.

Note: The **Balanced** setting typically produces utilization results that are very similar to the **Area** setting with better performance results. The **Area** setting can give better results in some cases. Performing WYSIWYG resynthesis for area in this way typically reduces register-to-register timing performance.

Related Links

[Perform WYSIWYG Primitive Resynthesis logic option](#)

For information about this logic option

14.2.4.5 Guideline: Use Register Packing

The **Auto Packed Registers** option implements the functions of two cells into one logic cell by combining the register of one cell in which only the register is used with the LUT of another cell in which only the LUT is used.

Related Links

[Auto Packed Registers logic option](#)

For more information about the Auto Packed Registers logic option

14.2.4.6 Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or Logic Lock (Standard) assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or Logic Lock (Standard) region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or Logic Lock (Standard) region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and Logic Lock (Standard) assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove Logic Lock (Standard) assignments in the Chip Planner, in the Logic Lock (Standard) Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

Related Links

[Analyzing and Optimizing the Design Floorplan](#) on page 280

14.2.4.7 Guideline: Flatten the Hierarchy During Synthesis

Synthesis tools typically provide the option of preserving hierarchical boundaries, which can be useful for verification or other purposes. However, the Intel Quartus Prime software optimizes across hierarchical boundaries so as to perform the most logic minimization, which can reduce area in a design with no design partitions.

If you are using Intel Quartus Prime incremental compilation, you cannot flatten your design across design partitions. Incremental compilation always preserves the hierarchical boundaries between design partitions, and the synthesis does not flatten it across partitions. Follow Intel's recommendations for design partitioning, such as registering partition boundaries to reduce the effect of cross-boundary optimizations.



14.2.4.8 Guideline: Retarget Memory Blocks

If your design fails to fit because it runs out of device memory resources, your design may require a certain type of memory that the device does not have.

If the memory block was created with a parameter editor, open the parameter editor and edit the RAM block type so it targets a new memory block size.

ROM and RAM memory blocks can also be inferred from your HDL code, and your synthesis software can place large shift registers into memory blocks by inferring the Shift register (RAM-based) IP core. This inference can be turned off in your synthesis tool to cause the memory or shift registers to be placed in logic instead of in memory blocks. Also, for improved timing performance, you can turn this inference off to prevent registers from being moved into RAM.

Depending on your synthesis tool, you can also set the RAM block type for inferred memory blocks. In Intel Quartus Prime synthesis, set the **ramstyle** attribute to the desired memory type for the inferred RAM blocks, or set the option to **logic**, to implement the memory block in standard logic instead of a memory block.

Consider the Resource Utilization by Entity report in the report file and determine whether there is an unusually high register count in any of the modules. Some coding styles can prevent the Intel Quartus Prime software from inferring RAM blocks from the source code because of the blocks' architectural implementation, and force the software to implement the logic in flipflops. As an example, a function such as an asynchronous reset on a register bank might make the register bank incompatible with the RAM blocks in the device architecture, so that the register bank is implemented in flipflops. It is often possible to move a large register bank into RAM by slight modification of associated logic.

14.2.4.9 Guideline: Use Physical Synthesis Options to Reduce Area

The physical synthesis options available at **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)** help you decrease resource usage. When you enable physical synthesis, the Intel Quartus Prime software makes placement-specific changes to the netlist that reduce resource utilization for a specific Intel device.

Note: Physical synthesis increases compilation time. To reduce the impact on compilation time, you can apply physical synthesis options to specific instances.

Related Links

[Advanced Fitter Settings Dialog Box](#)
In Intel Quartus Prime Help

14.2.4.10 Guideline: Retarget or Balance DSP Blocks

A design might not fit because it requires too many DSP blocks. You can implement all DSP block functions with logic cells, so you can retarget some of the DSP blocks to logic to obtain a fit.

If the DSP function was created with the parameter editor, open the parameter editor and edit the function so it targets logic cells instead of DSP blocks. The Intel Quartus Prime software uses the `DEDICATED_MULTIPLIER_CIRCUITRY` IP core parameter to control the implementation.

DSP blocks also can be inferred from your HDL code for multipliers, multiply-adders, and multiply-accumulators. You can turn off this inference in your synthesis tool. When you are using Intel Quartus Prime integrated synthesis, you can disable inference by turning off the **Auto DSP Block Replacement** logic option for your entire project. Click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**. Turn off **Auto DSP Block Replacement**. Alternatively, you can disable the option for a specific block with the Assignment Editor.

The Intel Quartus Prime software also offers the **DSP Block Balancing** logic option, which implements DSP block elements in logic cells or in different DSP block modes. The default **Auto** setting allows DSP block balancing to convert the DSP block slices automatically as appropriate to minimize the area and maximize the speed of the design. You can use other settings for a specific node or entity, or on a project-wide basis, to control how the Intel Quartus Prime software converts DSP functions into logic cells and DSP blocks. Using any value other than **Auto** or **Off** overrides the `DEDICATED_MULTIPLIER_CIRCUITRY` parameter used in IP core variations.

14.2.4.11 Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.

14.2.5 Routing

Resolve routing resource problems with these guidelines.

14.2.5.1 Guideline: Set Auto Packed Registers to Sparse or Sparse Auto

The **Auto Packed Registers** option reduces LE or ALM count in a design. You can set this option by clicking **Assignment > Settings > Compiler Settings > Advanced Settings (Fitter)**.

Related Links

[Auto Packed Registers logic option](#)

14.2.5.2 Guideline: Set Fitter Aggressive Routability Optimizations to Always

The **Fitter Aggressive Routability Optimization** option is useful if your design does not fit due to excessive routing wire utilization.

If there is a significant imbalance between placement and routing time (during the first fitting attempt), it might be because of high wire utilization. Turning on the **Fitter Aggressive Routability Optimizations** option can reduce your compilation time.

On average, this option can save up to 6% wire utilization, but can also reduce performance by up to 4%, depending on the device.

Related Links

[Fitter Aggressive Routability Optimizations logic option](#)

14.2.5.3 Guideline: Increase Router Effort Multiplier

The Router Effort Multiplier controls how quickly the router tries to find a valid solution. The default value is 1.0 and legal values must be greater than 0.



- Numbers higher than 1 help designs that are difficult to route by increasing the routing effort.
- Numbers closer to 0 (for example, 0.1) can reduce router runtime, but usually reduce routing quality slightly.

Experimental evidence shows that a multiplier of 3.0 reduces overall wire usage by approximately 2%. Using a Router Effort Multiplier higher than the default value can benefit designs with complex datapaths with more than five levels of logic. However, congestion in a design is primarily due to placement, and increasing the Router Effort Multiplier does not necessarily reduce congestion.

Note: Any Router Effort Multiplier value greater than 4 only increases by 10% for every additional 1. For example, a value of 10 is actually 4.6.

14.2.5.4 Guideline: Remove Fitter Constraints

A design with conflicting constraints or constraints that are difficult to meet may not fit in the targeted device. For example, a design might fail to fit if the location or Logic Lock (Standard) assignments are too strict and not enough routing resources are available on the device.

To resolve routing congestion caused by restrictive location constraints or Logic Lock (Standard) region assignments, use the **Routing Congestion** task in the Chip Planner to locate routing problems in the floorplan, then remove any internal location or Logic Lock (Standard) region assignments in that area. If your design still does not fit, the design is over-constrained. To correct the problem, remove all location and Logic Lock (Standard) assignments and run successive compilations, incrementally constraining the design before each compilation. You can delete specific location assignments in the Assignment Editor or the Chip Planner. To remove Logic Lock (Standard) assignments in the Chip Planner, in the Logic Lock (Standard) Regions Window, or on the Assignments menu, click **Remove Assignments**. Turn on the assignment categories you want to remove from the design in the **Available assignment categories** list.

Related Links

[Analyzing and Optimizing the Design Floorplan](#) on page 280

14.2.5.5 Guideline: Optimize Synthesis for Area, Not Speed

In some cases, resynthesizing the design to improve the area utilization can also improve the routability of the design. First, ensure that you have set your device and timing constraints correctly in your synthesis tool. Ensure that you do not overconstrain the timing requirements for the design, particularly when the area utilization of the design is a concern. Synthesis tools generally try to meet the specified requirements, which can result in higher device resource usage if the constraints are too aggressive.

If resource utilization is an important concern, you can optimize for area instead of speed.

- If you are using Intel Quartus Prime integrated synthesis, click **Assignments** ► **Settings** ► **Compiler Settings** ► **Advanced Settings (Synthesis)** and select **Balanced** or **Area** for the **Optimization Technique**.
- If you want to reduce area for specific modules in your design using the **Area** or **Speed** setting while leaving the default **Optimization Technique** setting at **Balanced**, use the Assignment Editor.
- You can also use the **Speed Optimization Technique for Clock Domains** logic option to specify that all combinational logic in or between the specified clock domain(s) is optimized for speed.
- In some synthesis tools, not specifying an f_{MAX} requirement can result in less resource utilization.

Optimizing for area or speed can affect the register-to-register timing performance.

Note:

In the Intel Quartus Prime software, the **Balanced** setting typically produces utilization results that are very similar to those produced by the **Area** setting, with better performance results. The **Area** setting can give better results in some cases.

The Intel Quartus Prime software provides additional attributes and options that can help improve the quality of your synthesis results.

Related Links

- [Intel Quartus Prime Integrated Synthesis](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1: Design and Synthesis*
- [Optimization Mode](#)
In Intel Quartus Prime Help

14.2.5.6 Guideline: Optimize Source Code

If your design does not fit because of routing problems and the methods described in the preceding sections do not sufficiently improve the routability of the design, modify the design at the source to achieve the desired results. You can often improve results significantly by making design-specific changes to your source code, such as duplicating logic or changing the connections between blocks that require significant routing resources.

14.2.5.7 Guideline: Use a Larger Device

If a successful fit cannot be achieved because of a shortage of routing resources, you might require a larger device.



14.3 Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime command-line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section either in an instance, or at a global level, or both.

Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <.qsf variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <.qsf variable name> <value> \  
-to <instance name>
```

Note: If the <value> field includes spaces (for example, 'Standard Fit'), you must enclose the value in straight double quotation marks.

Related Links

- [Tcl Scripting](#) on page 81
- [Intel Quartus Prime Standard Edition Settings File Reference Manual](#)
For information about all settings and constraints in the Intel Quartus Prime software.
- [Command Line Scripting](#) on page 67

14.3.1 Initial Compilation Settings

Use the Intel Quartus Prime Settings File (.qsf) variable name in the Tcl assignment to make the setting along with the appropriate value. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 45. Advanced Compilation Settings

Setting Name	.qsf File Variable Name	Values	Type
Placement Effort Multiplier	PLACEMENT_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Effort Multiplier	ROUTER_EFFORT_MULTIPLIER	Any positive, non-zero value	Global
Router Timing Optimization level	ROUTER_TIMING_OPTIMIZATION_LEVEL	NORMAL, MINIMUM, MAXIMUM	Global
Final Placement Optimization	FINAL_PLACEMENT_OPTIMIZATION	ALWAYS, AUTOMATICALLY, NEVER	Global

14.3.2 Resource Utilization Optimization Techniques

This table lists the .qsf file variable name and applicable values for Resource Utilization Optimization settings.



Table 46. Resource Utilization Optimization Settings

Setting Name	.qsf File Variable Name	Values	Type
Auto Packed Registers ⁽²⁾	QII_AUTO_PACKED_REGISTERS	AUTO, OFF, NORMAL, MINIMIZE AREA, MINIMIZE AREA WITH CHAINS, SPARSE, SPARSE AUTO	Global, Instance
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Perform Physical Synthesis for Combinational Logic for Reducing Area (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global, Instance
Perform Physical Synthesis for Mapping Logic to Memory (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA	ON, OFF	Global, Instance
Optimization Technique	<device family name>_OPTIMIZATION_TECHNIQUE	AREA, SPEED, BALANCED	Global, Instance
Speed Optimization Technique for Clock Domains	SYNTH_CRITICAL_CLOCK	ON, OFF	Instance
State Machine Encoding	STATE_MACHINE_PROCESSING	AUTO, ONE-HOT, GRAY, JOHNSON, MINIMAL BITS, ONE-HOT, SEQUENTIAL, USER-ENCODE	Global, Instance
Auto RAM Replacement	AUTO_RAM_RECOGNITION	ON, OFF	Global, Instance
Auto ROM Replacement	AUTO_ROM_RECOGNITION	ON, OFF	Global, Instance
Auto Shift Register Replacement	AUTO_SHIFT_REGISTER_RECOGNITION	ON, OFF	Global, Instance
Auto Block Replacement	AUTO_DSP_RECOGNITION	ON, OFF	Global, Instance
Number of Processors for Parallel Compilation	NUM_PARALLEL_PROCESSORS	Integer between 1 and 16 inclusive, or ALL	Global

(2) Allowed values for this setting depend on the device family that you select.



14.4 Document Revision History

Table 47. Document Revision History

Date	Version	Changes
2017.05.08	17.0.0	<ul style="list-style-type: none">Revised topics: <i>Resolving Resource Utilization Issues, Guideline: Optimize Synthesis for Area, Not Speed</i>
2016.05.02	16.0.0	<ul style="list-style-type: none">Stated limitations about deprecated physical synthesis options.
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
2014.12.15	14.1.0	Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations to Compiler Settings.
June 2014	14.0.0	<ul style="list-style-type: none">Removed Cyclone III and Stratix III devices references.Removed Macrocell-Based CPLDs related information.Updated template.
May 2013	13.0.0	Initial release.

Related Links

[Documentation Archive](#)

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



15 Analyzing and Optimizing the Design Floorplan

As FPGA designs grow larger in density, the ability to analyze the design for performance, routing congestion, and logic placement is critical to meet the design requirements. This chapter discusses how the Chip Planner and Logic Lock (Standard) regions help you improve your design's floorplan.

Design floorplan analysis helps to close timing, and ensure optimal performance in highly complex designs. With analysis capability, the Intel Quartus Prime Chip Planner helps you close timing quickly on your designs. You can use the Chip Planner together with Logic Lock (Standard) regions to compile your designs hierarchically and assist with floorplanning. Additionally, use partitions to preserve placement and routing results from individual compilation runs.

You can perform design analysis, as well as create and optimize the design floorplan with the Chip Planner. To make I/O assignments, use the Pin Planner.

Related Links

- [Managing Device I/O Pins](#) on page 24
- [Intel FPGA Technical Training](#)

15.1 Design Floorplan Analysis in the Chip Planner

The Chip Planner simplifies floorplan analysis by providing visual display of chip resources. With the Chip Planner, you can view post-compilation placement, connections, and routing paths.

The Chip Planner allows you to:

- Make assignment changes, such as creating and deleting resource assignments.
- Perform post-compilation changes such as creating, moving, and deleting logic cells and I/O atoms.
- Perform power and design analyses.
- Implement ECOs.
- Change connections between resources and make post-compilation changes to the properties of logic cells, I/O elements, PLLs, RAMs, and digital signal processing (DSP) blocks.

The Chip Planner showcases:

- Logic Lock (Standard) regions
- Relative resource usage
- Detailed routing information
- Fan-in and fan-out connections between nodes
- Timing paths between registers
- Delay estimates for paths
- Routing congestion information




Related Links

[Engineering Change Orders with the Chip Planner](#) on page 318

15.1.1 Starting the Chip Planner

To start the Chip Planner, select **Tools > Chip Planner**. You can also start the Chip Planner by the following methods:

- Click the Chip Planner icon  on the Intel Quartus Prime software toolbar.
- In the following tools, right-click any chip resource and select **Locate > Locate in Chip Planner**:
 - Design Partition Planner
 - Compilation Report
 - **Logic Lock (Standard) Regions Window**
 - Technology Map Viewer
 - **Project Navigator** window
 - RTL source code
 - Node Finder
 - Simulation Report
 - RTL Viewer
 - Report Timing panel of the Timing Analyzer

15.1.2 Chip Planner GUI Components

15.1.2.1 Chip Planner Toolbar

The Chip Planner toolbar provides powerful tools for visual design analysis. You can access Chip Planner commands either from the **View** or the **Shortcut** menu, or by clicking the icons in the toolbars.

15.1.2.2 Layers Settings and Editing Modes

The Chip Planner allows you to control the display of resources. To determine the operations that you can perform, use the **Editing Mode**.

Layers Settings Pane

With the **Layers Settings** pane, you can manage the graphic elements that the Chip Planner displays.

You open the **Layers Settings** pane by clicking **View > Layers Settings**. The **Layers Settings** pane offers layer presets, which group resources that are often used together. The **Basic**, **Detailed**, and **Floorplan Editing** default presets are useful for general assignment-related activities. You can also create custom presets tailored to your needs. The **Design Partition Planner** preset is optimized for specific activities.

Editing Mode

The Chip Planner's **Editing Mode** determines the operations that you can perform. The **Assignment** editing mode allows you to make assignment changes that are applied by the Fitter during the next place and route operation. The **ECO** editing mode allows you to make post-compilation changes, commonly referred to as engineering change orders (ECOs).

Select the editing mode appropriate for the work that you want to perform, and a preset that displays the resources that you want to view, in a level of detail appropriate for your design.

Related Links

- [Viewing Architecture-Specific Design Information](#) on page 283
- [Layers Settings Dialog Box](#)
In Intel Quartus Prime Help

15.1.2.3 Locate History Window

As you optimize your design floorplan, you might have to locate a path or node in the Chip Planner more than once. The **Locate History** window lists all the nodes and paths you have displayed using a **Locate in Chip Planner** command, providing easy access to the nodes and paths of interest to you.

If you locate a required path from the **Timing Analyzer Report Timing** pane, the **Locate History** window displays the required clock path. If you locate an arrival path from the **Timing Analyzer Report Timing** pane, the **Locate History** window displays the path from the arrival clock to the arrival data. Double-clicking a node or path in the **Locate History** window displays the selected node or path in the Chip Planner.

Related Links

[Engineering Change Orders with the Chip Planner](#) on page 318

15.1.2.4 Chip Planner Floorplan Views

The Chip Planner uses a hierarchical zoom viewer that shows various abstraction levels of the targeted Intel device. As you zoom in, the level of abstraction decreases, revealing more details about your design.

Bird's Eye View

The Bird's Eye View displays a high-level picture of resource usage for the entire chip and provides a fast and efficient way to navigate between areas of interest in the Chip Planner.

The Bird's Eye View is particularly useful when the parts of your design that you want to view are at opposite ends of the chip, and you want to quickly navigate between resource elements without losing your frame of reference.



Properties Window

The **Properties** window displays detailed properties of the objects (such as atoms, paths, Logic Lock (Standard) regions, or routing elements) currently selected in the Chip Planner. To display the **Properties** window, right-click the object and select **View > Properties**.

Related Links

- [Engineering Change Orders with the Chip Planner](#) on page 318
- [Bird's Eye View Window](#)
In Intel Quartus Prime Help

15.1.3 Viewing Architecture-Specific Design Information

The Chip Planner allows you to view architecture-specific information related to your design. By enabling the options in the **Layers Settings** pane, you can view:

- **Device routing resources used by your design**—View how blocks are connected, as well as the signal routing that connects the blocks.
- **LE configuration**—View logic element (LE) configuration in your design. For example, you can view which LE inputs are used; whether the LE utilizes the register, the look-up table (LUT), or both; as well as the signal flow through the LE.
- **ALM configuration**—View ALM configuration in your design. For example, you can view which ALM inputs are used; whether the ALM utilizes the registers, the upper LUT, the lower LUT, or all of them. You can also view the signal flow through the ALM.
- **I/O configuration**—View device I/O resource usage. For example, you can view which components of the I/O resources are used, whether the delay chain settings are enabled, which I/O standards are set, and the signal flow through the I/O.
- **PLL configuration**—View phase-locked loop (PLL) configuration in your design. For example, you can view which control signals of the PLL are used with the settings for your PLL.
- **Timing**—View the delay between the inputs and outputs of FPGA elements. For example, you can analyze the timing of the `DATAB` input to the `COMBOUT` output.

In addition, you can modify the following device properties with the Chip Planner:

- LEs and ALMs
- I/O cells
- PLLs
- Registers in RAM and DSP blocks
- Connections between elements
- Placement of elements

For more information about LEs, ALMs, and other resources of an FPGA device, refer to the relevant device handbook.

Related Links

- [Layers Settings and Editing Modes](#) on page 281

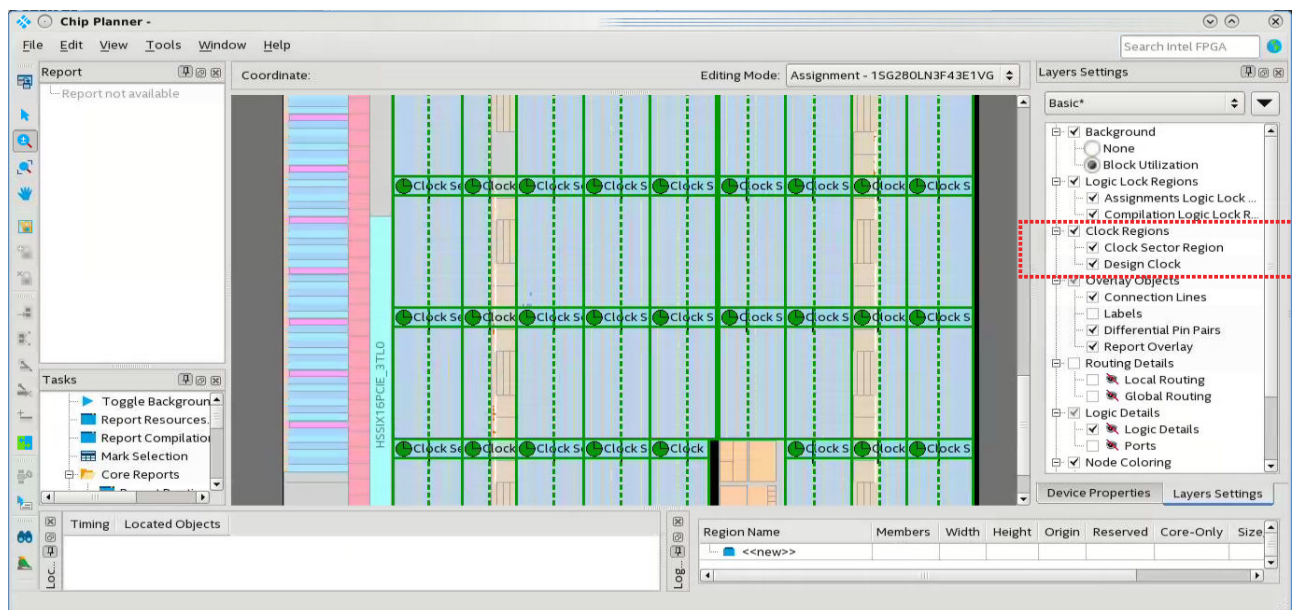
- [Layers Settings Dialog Box](#)
In Intel Quartus Prime Help

15.1.4 Viewing Available Clock Networks in the Device

When you enable a clock region layer in the **Layers Settings** pane, you display the areas of the chip that are driven by global and regional clock networks. When the selected device does not contain a given clock region, the option for that category is unavailable in the dialog box.

This global clock display feature is available for Arria V, Intel Arria 10, Cyclone V, Stratix IV, and Stratix V device families.

Figure 99. Clock Regions



- Depending on the clock layers that you activate in the **Layers Settings** pane, the Chip Planner displays regional and global clock regions in the device, and the connectivity between clock regions, pins, and PLLs.
- Clock regions appear as rectangular overlay boxes with labels indicating the clock type and index. Select a clock network region by clicking the clock region. The clock-shaped icon at the top-left corner indicates that the region represents a clock network region.
- Spine/sector clock regions have a dotted vertical line in the middle. This dotted line indicates where two columns of row clocks meet in a sector clock.
- To change the color in which the Chip Planner displays clock regions, select **Tools > Options > Colors > Clock Regions**.

Related Links

- [Spine Clock Limitations](#) on page 229
- [Layers Settings and Editing Modes](#) on page 281



- [Report Spine Clock Utilization dialog box \(Chip Planner\)](#)
In Intel Quartus Prime Help

15.1.5 Viewing Routing Congestion

The **Report Routing Utilization** task allows you to determine the percentage of routing resources in use following a compilation. This feature can identify zones with lack of routing resources, helping you to make design changes to meet routing congestion design requirements.

To view the routing congestion in the Chip Planner:

1. In the **Tasks** pane, double-click the **Report Routing Utilization** command to launch the **Report Routing Utilization** dialog box.
2. Click **Preview** in the **Report Routing Utilization** dialog box to preview the default congestion display.
3. Change the **Routing Utilization Type** to display congestion for specific resources.
The default display uses dark blue for 0% congestion (blue indicates zero utilization) and red for 100%. You can adjust the slider for **Threshold percentage** to change the congestion threshold level.

The congestion map helps you determine whether you can modify the floorplan, or modify the RTL to reduce routing congestion. Consider:

- The routing congestion map uses the color and shading of logic resources to indicate relative resource utilization; darker shading represents a greater utilization of routing resources. Areas where routing utilization exceeds the threshold value that you specify in the **Report Routing Utilization** dialog box appear in red.
- To identify a lack of routing resources, you must investigate each routing interconnect type separately by selecting each interconnect type in turn in the **Routing Utilization Settings** dialog box.
- The Compiler's messages contain information about average and peak interconnect usage. Peak interconnect usage over 75%, or average interconnect usage over 60%, can indicate difficulties fitting your design. Similarly, peak interconnect usage over 90%, or average interconnect usage over 75%, show increased chances of not getting a valid fit.

Related Links

[Viewing Routing Resources](#) on page 288

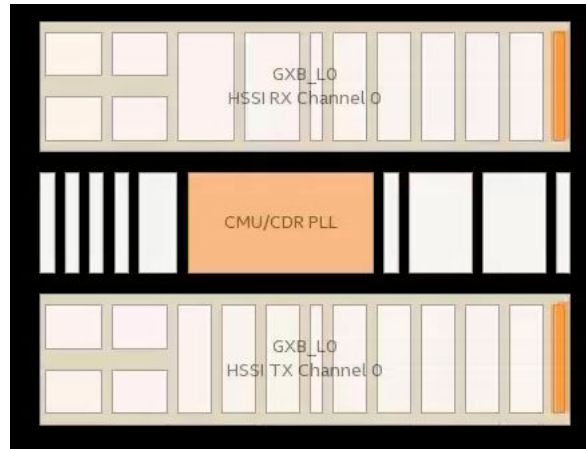
15.1.6 Viewing I/O Banks

To view the I/O bank map of the device in the Chip Planner, double-click **Report All I/O Banks** in the **Tasks** pane.

15.1.7 Viewing High-Speed Serial Interfaces (HSSI)




For selected device families, the Chip Planner displays a detailed block view of the receiver and transmitter channels of the high-speed serial interfaces. To display the HSSI block view, double-click **Report HSSI Block Connectivity** in the **Tasks** pane.

Figure 100. Intel Arria 10 HSSI Channel Blocks



15.1.8 Generating Fan-In and Fan-Out Connections


Displays the atoms that fan-in to or fan-out from a resource.

- To display the fan-in or fan-out connections from a resource you selected, use the **Generate Fan-In Connections** icon  or the **Generate Fan-Out Connections** icon  in the Chip Planner toolbar.
- To remove the connections displayed, use the **Clear Unselected Connections** icon  in the Chip Planner toolbar. Alternatively, use the **View** menu.

15.1.9 Generating Immediate Fan-In and Fan-Out Connections

Displays the immediate fan-in or fan-out connection for the selected atom.


For example, when you view the immediate fan-in for a logic resource, you see the routing resource that drives the logic resource. You can generate immediate fan-ins and fan-outs for all logic resources and routing resources.

- To display the immediate fan-in or fan-out connections, click **View > Generate Immediate Fan-In Connections** or **View > Generate Immediate Fan-Out Connections**.
- To remove the connections displayed, use the **Clear Unselected Connections** icon  in the Chip Planner toolbar.

15.1.10 Exploring Paths in the Chip Planner

Use the Chip Planner to explore paths between logic elements. The following examples use the Chip Planner to traverse paths from the Timing Analysis report.

15.1.10.1 Analyzing Connections for a Path

To determine the elements forming a selected path or connection in the Chip Planner, click the **Expand Connections** icon  in the Chip Planner toolbar.



Related Links

Expand Connections Command (View Menu)
In *Intel Quartus Prime Help*

15.1.10.2 Locate Path from the Timing Analysis Report to the Chip Planner

To locate a path from the Timing Analysis report to the Chip Planner, perform the following steps:

1. Select the path you want to locate in the Timing Analysis report.
2. Right-click the path and point to **Locate Path** ► **Locate in Chip Planner**. The path appears in the **Locate History** window of the Chip Planner.


Figure 101. Path List in the Locate History Window

Timing	Located Objects
-0.790	ram1--port_b_address1FITTER_CREATED_FF -> ram1
-0.758	ram1--port_b_address2FITTER_CREATED_FF -> ram1
-0.753	ram1--port_b_address1FITTER_CREATED_FF -> ram1
-0.725	ram1--port_b_address1FITTER_CREATED_FF -> ram1
-0.723	ram1--port_b_address4FITTER_CREATED_FF -> ram1
-0.710	ram1--port_b_address4FITTER_CREATED_FF -> ram1
-0.707	ram1--port_b_address0FITTER_CREATED_FF -> ram1
-0.678	ram1--port_b_address0FITTER_CREATED_FF -> ram1
-0.677	ram1--port_b_address0FITTER_CREATED_FF -> ram1
-0.670	ram1--port_b_address3FITTER_CREATED_FF -> ram1

Related Links

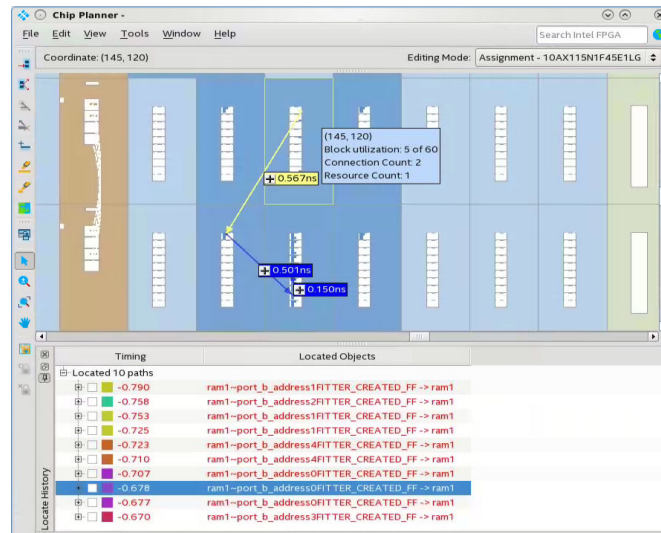
Displaying Path Reports with the Timing Analyzer on page 217

15.1.10.3 Show Delays

With the **Show Delays** feature, you can view timing delays for paths appearing in Timing Analyzer reports. To access this feature, click **View** ► **Show Delays** in the main menu. Alternatively click the Show Delays icon  in the Chip Planner toolbar. To see the partial delays on the selected path, click the "+" sign next to the path delay displayed in the **Locate History** window.

For example, you can view the delay between two logic resources or between a logic resource and a routing resource.

Figure 102. Show Delays Associated in a Timing Analyzer Path



15.1.10.4 Viewing Routing Resources

With the Chip Planner and the **Locate History** window, you can view the routing resources that a path or connection uses. You can also select and display the Arrival Data path and the Arrival Clock path.

Figure 103. Show Physical Routing

In the **Locate History** window, right-click a path and select **Show Physical Routing** to display the physical path. To adjust the display, right-click and select **Zoom to Selection**.

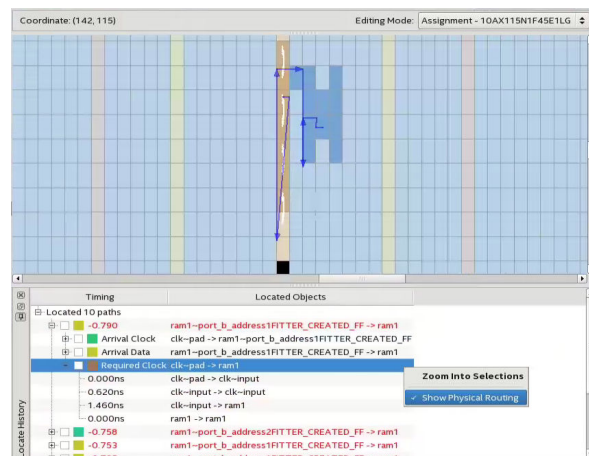
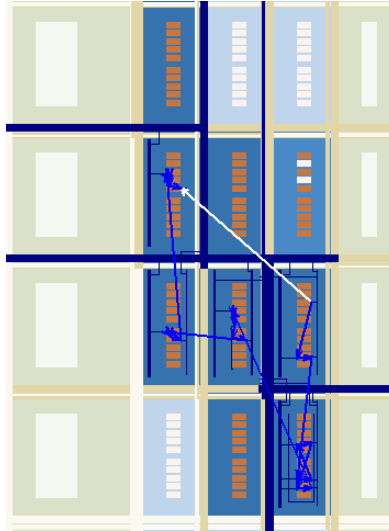


Figure 104. Highlight Routing

To see the rows and columns where the Fitter routed the path, right-click a path and select **Highlight Routing**.



Related Links

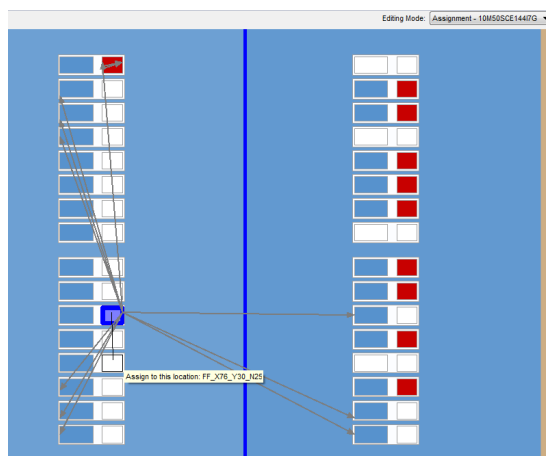
[Viewing Routing Congestion](#) on page 285

15.1.11 Viewing Assignments in the Chip Planner

You can view location assignments in the Chip Planner by using the Assignment editing mode and the **Floorplan Editing** preset in the **Layers Settings** pane.

The Chip Planner displays assigned resources in a predefined color (gray, by default).

Figure 105. Viewing Assignments in the Chip Planner



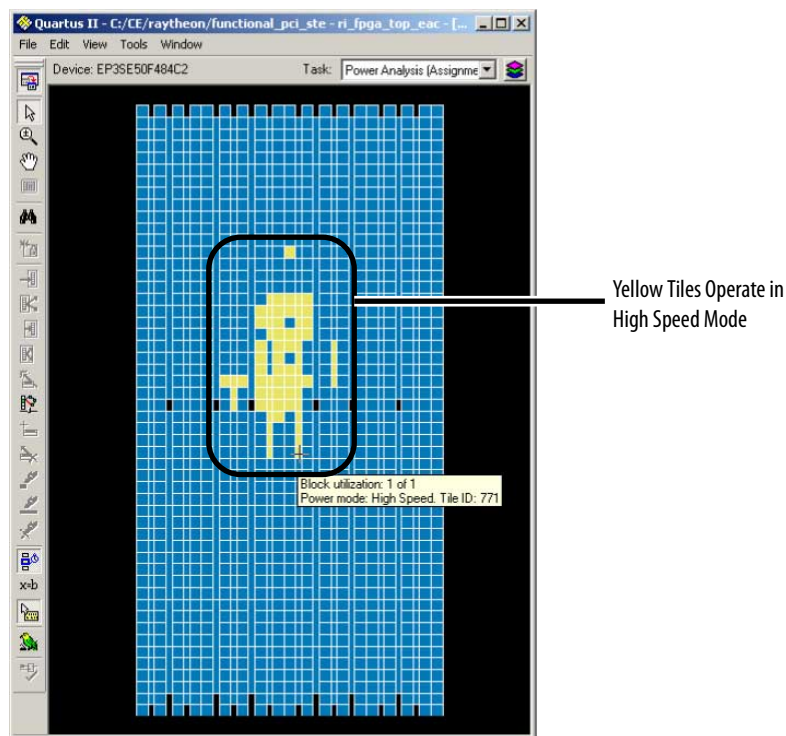
To create or move an assignment, or to make node and pin location assignments to Logic Lock (Standard) regions, drag the selected resource to a new location. The Fitter applies the assignments that you create during the next place-and-route operation.

15.1.12 Viewing High-Speed and Low-Power Tiles in the Chip Planner

Some Intel devices have ALMs that can operate in either high-speed mode or low-power mode. The power mode is set during the fitting process in the Intel Quartus Prime software. These ALMs are grouped together to form larger blocks, called “tiles”.

To view a power map, double-click **Tasks > Core Reports > Report High-Speed/ Low-Power Tiles** after running the Fitter. The Chip Planner displays low-power and high-speed tiles in contrasting colors; yellow tiles operate in a high-speed mode, while blue tiles operate in a low-power mode.

Figure 106. Viewing High-Speed and Low Power Tiles in a Stratix Device



Related Links

[AN 514: Power Optimization in Stratix IV FPGAs](#)

15.1.13 Viewing Design Partition Placement

With the **Report Design Partitions** command, you can view the physical placement of design partitions using the same color map as the Design Partition Planner.

The **Report Design Partitions Advanced** command opens the **Report Design Partitions Advanced** dialog box that allows you to select a partition and generate a report of the pins belonging to the partition. It highlights the selected partition's boundary ports and pins in the Chip Planner, and optionally reports the routing utilization and routing element details.



15.2 Logic Lock (Standard) Regions

Logic Lock (Standard) regions are floorplan location constraints. When you assign instances or nodes to a Logic Lock (Standard) region, you direct the Fitter to place those instances or nodes within the region. A floorplan can contain multiple Logic Lock (Standard) regions.

You can define a Logic Lock (Standard) region by its height, width, and location; Alternatively, you can specify the size or location of a region, or both, or the Intel Quartus Prime software can generate these properties automatically. The Intel Quartus Prime software bases the size and location of a region on the contents of the region and the timing requirements of the module. The Intel Quartus Prime software displays Logic Lock (Standard) regions with colors indicating the percentage of resources available in the region. An orange Logic Lock (Standard) region indicates a nearly full Logic Lock (Standard) region.

The attributes of a Logic Lock regions are:

Table 48. Attributes of Logic Lock (Standard) Regions

Name	Value	Behavior
Size	Auto Fixed	Auto allows the Intel Quartus Prime software to determine the appropriate size of a region given its contents. Fixed regions have a shape and size that you define.
Width	Number of columns	Specifies the width of the Logic Lock (Standard) region.
Height	Number of rows	Specifies the height of the Logic Lock (Standard) region.
State	Floating Locked	Floating allows the Intel Quartus Prime software to determine the location of the region on the device. Floating regions appear with a dashed boundary in the floorplan. Locked allows you to specify the location of the region. Locked regions appear with a solid boundary in the floorplan. A Locked region must have a Fixed size.
Origin	Any Floorplan Location Undetermined	Specifies the location of the Logic Lock (Standard) region on the floorplan. The origin is at the lower left corner of the Logic Lock (Standard) region.
Reserved	Off On	Prevents the Fitter from placing other logic in the region.

Intel Quartus Prime software cannot automatically define the size of a region if the location is **Locked**. Therefore, if you want to specify the exact location of the region, you must also specify the size.

You can use the Design Partition Planner in conjunction with Logic Lock (Standard) regions to create a floorplan for your design.

Related Links

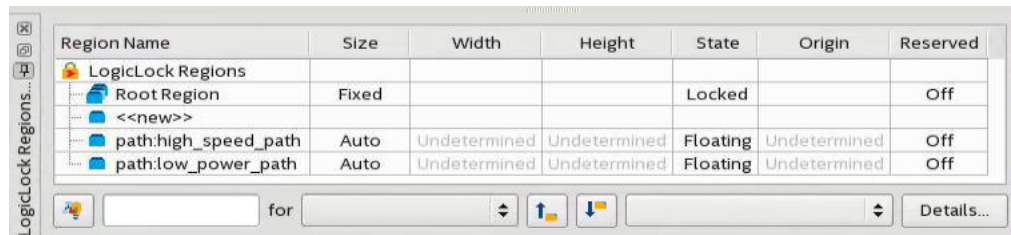
[Defining Routing Regions](#) on page 295

15.2.1 Logic Lock (Standard) Regions Window

The Logic Lock (Standard) Regions Window provides a summary of all Logic Lock (Standard) regions defined in your design. Use the Logic Lock (Standard) Regions Window to create, assign elements, and modify properties of a Logic Lock (Standard) region.

Open the Logic Lock (Standard) Regions Window in the Chip Planner by clicking **View > Logic Lock (Standard) Window**, and in Intel Quartus Prime by clicking **Assignments > Logic Lock (Standard) Window**.

Figure 107. Logic Lock (Standard) Regions Window



Region Name	Size	Width	Height	State	Origin	Reserved
LogicLock Regions						
Root Region	Fixed			Locked		Off
<<new>>						
path:high_speed_path	Auto	Undetermined	Undetermined	Floating	Undetermined	Off
path:low_power_path	Auto	Undetermined	Undetermined	Floating	Undetermined	Off

The Logic Lock (Standard) Regions Window also has a recommendations toolbar; select a Logic Lock (Standard) region from the drop-down list in the recommendations toolbar to display the relevant suggestions to optimize that Logic Lock (Standard) region.

The Intel Quartus Prime software automatically creates a Logic Lock (Standard) region that encompasses the entire device. This default region is labeled `Root_Region`, and is locked and fixed.

You can customize the Logic Lock (Standard) Regions Window by dragging and dropping the columns to change their order; you can also show and hide optional columns by right-clicking any column heading and then selecting the appropriate columns in the shortcut menu.

Logic Lock (Standard) Regions Properties Dialog Box

Use the **Logic Lock (Standard) Regions Properties** dialog box to view and modify detailed information about your Logic Lock (Standard) region, such as which entities and nodes are assigned to your region, and which resources are required.

To open the **Logic Lock (Standard) Regions Properties** dialog box, right-click the region and select **Logic Lock (Standard) Regions Properties....**

15.2.2 Creating Logic Lock (Standard) Regions

You can create Logic Lock (Standard) Regions using several methods:

Creating Logic Lock (Standard) Regions with the Project Navigator:

1. Perform either a full compilation or analysis and elaboration on the design.
2. If the Project Navigator is not already open, click **View > Utility Windows > Project Navigator**. The Project Navigator displays the hierarchy of the design.
3. With the design hierarchy fully expanded, right-click any design entity, and click **Create New Logic Lock (Standard) Region**.
4. Assign the entity to the new region.



The new region has the same name as the entity.

Creating Logic Lock (Standard) Regions with the Logic Lock (Standard) Regions Window:

1. Click **Assignments** > **Logic Lock (Standard) Regions Window**.
2. In the **Logic Lock (Standard) Regions** window, click <<new>>.

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape doesn't matter.

Creating Logic Lock (Standard) Regions with the Chip Planner:

1. Click **View** > **Logic Lock (Standard) Regions** > **Create Logic Lock (Standard) Region**
2. Click and drag on the Chip Planner floorplan to create a region of your preferred location and size.

After you create the region, you can define the region shape and then assign a single entity to the region. The order that you assign the entity or define the shape doesn't matter.

Related Links

- [Placing Device Resources into Logic Lock \(Standard\) Regions](#) on page 296
- [Node Finder Command](#)
In *Intel Quartus Prime Help*
- [Creating Logic Lock \(Standard\) Assignments with Tcl commands](#)
- [Creating or Modifying Logic Lock \(Standard\) Regions](#) on page 302

15.2.2.1 Considerations on Using Auto Sized Regions

If you use **Auto** Sized Logic Lock (Standard) regions, take into account:



- **Auto/Floating** regions cannot be reserved.
- Verify that your Logic Lock (Standard) region is not empty. If you do not assign any instance to the region, the Fitter reduces the size to 0 by 0, making the region invalid.
- The region may or may not be associated with a partition. When you combine partitions with **Auto** Sized Logic Lock (Standard) regions, you get flexibility to solve your particular fitting challenges. However, every constraint that you add reduces the solutions available, and too many constraints can result in the Fitter not finding a solution. Some cases are:
 - If a partition is preserved at synthesis or not preserved, the Logic Lock (Standard) region confines the logic to a specific area, allowing the Fitter to optimize the logic within the partition, and optimize the placement within the Logic Lock (Standard) region.
 - If a partition is preserved at placement, routed, or final; a Logic Lock (Standard) region is not an effective placement boundary, because the location of the partition's logic is fixed.
 - However, if the Logic Lock (Standard) region is reserved, the Fitter avoids placing other logic in the area, which can help you reduce resource congestion.
- Once the outcome of the Logic Lock (Standard) region meets your specification, you can:
 - Convert the Logic Lock (Standard) region to **Fixed** Size.
 - Leave the Logic Lock (Standard) region with **Auto** Sized attribute and use the region as a "keep together" type of function.
 - If the Logic Lock (Standard) region is also a partition, you can preserve the place and route through the partition and remove the Logic Lock (Standard) region entirely.

15.2.2.2 Customizing the Shape of Logic Lock Regions

To create custom shaped Logic Lock regions, you can perform logic operations. Non-rectangular Logic Lock (Standard) regions can help you exclude certain resources, or place parts of your design around specific device resources to improve performance.

Attention: There is no undo feature for the Logic Lock (Standard) shapes for 17.1.

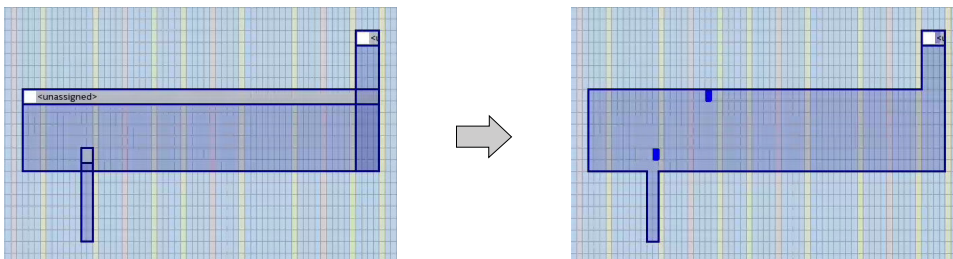
15.2.2.2.1 Merging Logic Lock (Standard) Regions

To merge two or more Logic Lock (Standard) regions, perform the following steps:

1. Ensure that no more than one of the regions that you intend to merge has logic assignments.
2. Arrange the regions into the locations where you want the resultant region.
3. Select all the individual regions that you want to merge by clicking each of them while pressing the Shift key.
4. Right-click the title bar of any of the selected Logic Lock (Standard) regions and select **Logic Lock (Standard) Regions > Merge Logic Lock (Standard) Region**. The individual regions that you select merge to create a single new region.

Note: By default, the new Logic Lock (Standard) region has the same name as the component region containing the greatest number of resources; however, you can rename the new region. In the **Logic Lock (Standard) Regions Window**, the new region is shown as having a **Custom Shape**.

Figure 108. Using the Merge Logic Lock (Standard) Region command



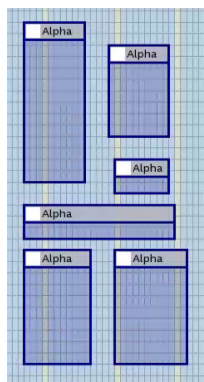
Related Links

[Creating Logic Lock \(Standard\) Regions on page 292](#)

15.2.2.3 Noncontiguous Logic Lock (Standard) Regions

You can create disjointed regions by using the Logic Lock (Standard) region manipulation tools. Noncontiguous regions act as a single Logic Lock (Standard) region for all Logic Lock (Standard) region attributes.

Figure 109. Noncontiguous Logic Lock (Standard) Region



Related Links

[Merging Logic Lock \(Standard\) Regions on page 294](#)

15.2.3 Defining Routing Regions

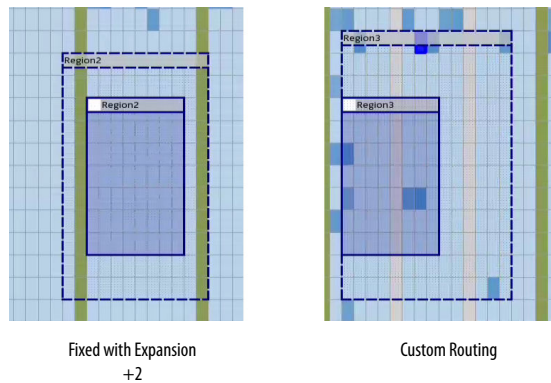
A routing region is an element of a Logic Lock region that specifies the routing area. A routing region must encompass the existing Logic Lock placement region. Routing regions cannot be set as reserved. To define the routing region, double-click the **Routing Region** cell in the **Logic Lock (Standard) Regions** window, and select an option from the drop-down menu.

Valid routing region options are:

Table 49. Routing Region Options

Option	Description
Unconstrained (default)	Allows the fitter to use any available routes on the device.
Whole Chip	Same as Unconstrained, but writes the constraint in the Intel Quartus Prime settings file (.qsf).
Fixed with Expansion	Follows the outline of the placement region. The routing region scales by a number of rows/cols larger than the placement region.
Custom	Allows you to make a custom shape routing region around the Logic Lock region. When you select the Custom option, the placement and routing regions move independently in the Chip Planner. In this case, move the placement and routing regions by selecting both using the Shift key.

Figure 110. Routing Regions



15.2.4 Placing Logic Lock (Standard) Regions

A fixed region must contain all resources required by the design block assigned to the region. Although the Intel Quartus Prime software can automatically place and size Logic Lock (Standard) regions to meet resource and timing requirements, you can manually place and size regions to meet your design requirements.

If you manually place or size a Logic Lock (Standard) region:

- Logic Lock (Standard) regions with pin assignments must be placed on the periphery of the device, adjacent to the pins. You must also include the I/O block within the Logic Lock (Standard) Region.
- Floating Logic Lock (Standard) regions can overlap with their ancestors or descendants, but not with other floating Logic Lock (Standard) regions.

15.2.5 Placing Device Resources into Logic Lock (Standard) Regions

You can assign an entity in the design to only one Logic Lock (Standard) region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a sub region without reserving the resources in the sub region.



If a Logic Lock (Standard) region boundary includes part of a device resource, the Intel Quartus Prime software allocates the entire resource to that Logic Lock (Standard) region. When the Intel Quartus Prime software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the Logic Lock (Standard) region.

To add an instance using the **Logic Lock Region** window, right-click the region and select **Logic Lock Properties ► Add**. Alternatively, in the Intel Quartus Prime software you can drag entities from the Hierarchy viewer into a Logic Lock (Standard) region's name field in the Logic Lock (Standard) Regions Window.

15.2.5.1 Empty Logic Lock Regions

Intel Quartus Prime allows you to have Logic Lock regions with no members. Empty regions are a tool to manage space in the FPGA for future logic. This technique only works when you set the regions to **Reserved**

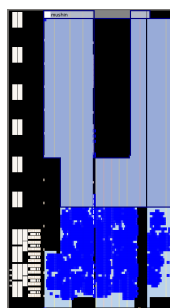
Some reasons to use empty Logic Lock regions are:

- Preliminary floorplanning.
- Complex incremental builds.
- Team based design and interconnect logic.
- Confining logic placements.

Since Logic Lock regions do not reserve any routing resources, the Fitter may use the area for routing purposes.

Use the **Core Only** attribute for empty Logic Lock regions. When you include periphery resources in empty regions, you restrict the periphery component placement, which can result in a no fit design. After you name the empty region, you can perform the same manipulations as with any populated Logic Lock Region.

Figure 111. Logic Placed Outside of an Empty region



The figure shows an empty Logic Lock region and the logic around it. However, some IOs, HSSIIO, and PLLs are in the empty region. This placement happens because the output port connects to the IO, and the IO is always part of the root_partition (top-level partition).



15.2.5.2 Pin Assignment

A Logic Lock (Standard) region incorporates all device resources within its boundaries, including memory and pins. The Intel Quartus Prime Standard Edition software automatically includes pins when you assign an instance to a region. You can manually exclude pins with a **Core Only** assignment.

Note: Pin assignments to Logic Lock (Standard) regions are effective only in fixed and locked regions. Pin assignments to floating regions do not influence the placement of the region.

You can assign an entity in the design to only one Logic Lock (Standard) region, but the entity can inherit regions by hierarchy. This hierarchy allows a reserved region to have a subregion without reserving the resources in the subregion.

When the Intel Quartus Prime software places a floating auto-sized region, it places the region in an area that meets the requirements of the contents of the Logic Lock (Standard) region.

15.2.5.3 Reserved Logic Lock (Standard) Regions

The **Reserved** attribute instructs the Fitter to only place the entities and nodes that you specifically assigned to the Logic Lock (Standard) region in the Logic Lock (Standard) region.

The Intel Quartus Prime software honors all entity and node assignments to Logic Lock (Standard) regions. Occasionally entities and nodes do not occupy an entire region, which leaves some of the region's resources unoccupied.

To increase the region's resource utilization and performance, Intel Quartus Prime software by default fills the unoccupied resources with other nodes and entities that have not been assigned to another region. To prevent this behavior, turn on **Reserved** on the **Logic Lock (Standard) Region Properties** ► **General** tab.

15.2.5.4 Excluded Resources

The Excluded Resources feature allows you to easily exclude specific device resources such as DSP blocks or M4K memory blocks from a Logic Lock (Standard) region.

For example, you can assign a specific entity to a Logic Lock (Standard) region but allow the DSP blocks of that entity to be placed anywhere on the device. Use the Excluded Resources feature on a per-Logic Lock (Standard) region member basis.

To exclude certain device resources from an entity, in the **Logic Lock (Standard) Region Properties** dialog box, highlight the entity in the **Design Element** column, and click **Edit**. In the **Edit Node** dialog box, under **Excluded Element Types**, click the **Browse** button. In the **Excluded Resources Element Types** dialog box, you can select the device resources you want to exclude from the entity. When you have selected the resources to exclude, the **Excluded Resources** column is updated in the **Logic Lock (Standard) Region Properties** dialog box to reflect the excluded resources.



Note: The Excluded Resources feature prevents certain resource types from being included in a region, but it does not prevent the resources from being placed inside the region unless you set the region's **Reserved** property to **On**. To indicate to the Fitter that certain resources are not required inside a Logic Lock (Standard) region, define a resource filter.

15.2.5.5 Logic Lock (Standard) Assignment Precedence

You can encounter conflicts during the assignment of entities and nodes to Logic Lock (Standard) regions. For example, an entire top-level entity might be assigned to one region and a node within this top-level entity assigned to another region.

To resolve conflicting assignments, the Intel Quartus Prime software maintains an order of precedence for Logic Lock (Standard) assignments. The following order of precedence, from highest to lowest, applies:

1. Exact node-level assignments
2. Path-based and wildcard assignments
3. Hierarchical assignments

Note: To open the **Priority** dialog box, select **Logic Lock (Standard) Regions Properties > General > Priority**. You can change the priority of path-based and wildcard assignments with the **Up** and **Down** buttons in the **Priority** dialog box. To prioritize assignments between regions, you must select multiple Logic Lock (Standard) regions and then open the **Priority** dialog box from the **Logic Lock (Standard) Regions Properties** dialog box.

15.2.5.6 Virtual Pins

A virtual pin is an I/O element that the Compiler temporarily maps to a logic element, and not to a pin during compilation. The software implements virtual pins as LUTs. To assign a Virtual Pin, use the Assignment Editor. You can create virtual pins by assigning the **Virtual Pin** logic option to an I/O element.

When you apply the **Virtual Pin** assignment to an input pin, the pin no longer appears as an FPGA pin; the Compiler fixes the virtual pin to GND in the design. The virtual pin is not a floating node.

Use virtual pins only for I/O elements in lower-level design entities that become nodes after you import the entity to the top-level design; for example, when compiling a partial design.

Note: The **Virtual Pin** logic option must be assigned to an input or output pin. If you assign this option to a bidirectional pin, tri-state pin, or registered I/O element, Analysis & Synthesis ignores the assignment. If you assign this option to a tri-state pin, the Fitter inserts an I/O buffer to account for the tri-state logic; therefore, the pin cannot be a virtual pin. You can use multiplexer logic instead of a tri-state pin if you want to continue to use the assigned pin as a virtual pin. Do not use tri-state logic except for signals that connect directly to device I/O pins.

In the top-level design, you connect these virtual pins to an internal node of another module. By making assignments to virtual pins, you can place those pins in the same location or region on the device as that of the corresponding internal nodes in the top-level module. You can use the **Virtual Pin** option when compiling a Logic Lock

(Standard) module with more pins than the target device allows. The **Virtual Pin** option can enable timing analysis of a design module that more closely matches the performance of the module after you integrate it into the top-level design.

To display all assigned virtual pins in the design with the Node Finder, you can set **Filter Type** to **Pins: Virtual**. To access the Node Finder from the Assignment Editor, double-click the **To** field; when the arrow appears on the right side of the field, click and select **Node Finder**.

Related Links

- [Assigning Virtual Pins with a Tcl command](#) on page 304
- [Managing Device I/O Pins](#) on page 24
- [Node Finder Command \(View Menu\)](#)
In Intel Quartus Prime Help

15.2.6 Hierarchical (Parent and Child) Logic Lock (Standard) Regions

To further constrain module locations, you can define a hierarchy for a group of regions by declaring parent and child regions.

The Intel Quartus Prime software places a child region completely within the boundaries of its parent region; a child region must be placed entirely within the boundary of its parent. Additionally, parent and child regions allow you to further improve the performance of a module by constraining nodes in the critical path of a module.

To make one Logic Lock (Standard) region a child of another Logic Lock (Standard) region, in the Logic Lock (Standard) Regions window, select the new child region and dragging the new child region into its new parent region.

Note: The Logic Lock (Standard) region hierarchy does not have to be the same as the design hierarchy.

You can create both auto-sized and fixed-sized Logic Lock (Standard) regions within a parent Logic Lock (Standard) region; however, the parent of a fixed-sized child region must also be fixed-sized. The location of a locked parent region is locked relative to the device; the location of a locked child region is locked relative to its parent region. If you change the parent's location, the locked child's origin changes, but maintains the same placement relative to the origin of its parent. The location of a floating child region can float within its parent. Complex region hierarchies might result in some LABs not being used, effectively increasing the resource utilization in the device. Do not create more levels of hierarchy than you need.

15.2.7 Additional Intel Quartus Prime Logic Lock (Standard) Design Features

To complement the **Logic Lock (Standard) Regions Window**, the Intel Quartus Prime software has additional features to help you design with Logic Lock (Standard) regions.



15.2.7.1 Analysis and Synthesis Resource Utilization by Entity

The Compilation Report contains an **Analysis and Synthesis Resource Utilization by Entity** section, which reports resource usage statistics, including entity-level information. You can use this feature to verify that any Logic Lock (Standard) region you manually create contains enough resources to accommodate all the entities you assign to it.

15.2.7.2 Intel Quartus Prime Revisions Feature

When you evaluate different Logic Lock (Standard) regions in your design, you might want to experiment with different configurations to achieve your desired results. The Intel Quartus Prime Revisions feature allows you to organize the same project with different settings until you find an optimum configuration.

To use the Revisions feature, choose **Project > Revisions**. You can create a revision from the current design or any previously created revisions. Each revision can have an associated description. You can use revisions to organize the placement constraints created for your Logic Lock (Standard) regions.

15.3 Using Logic Lock (Standard) Regions in the Chip Planner

You can easily create Logic Lock (Standard) regions in the Chip Planner and assign resources to them.

15.3.1 Viewing Connections Between Logic Lock (Standard) Regions in the Chip Planner

You can view and edit Logic Lock (Standard) regions using the Chip Planner. To view and edit Logic Lock (Standard) regions, use **Floorplan Editing** in the **Layers Settings** window, or any layers setting mode that has the **User-assigned Logic Lock (Standard) regions** setting enabled.

The Chip Planner shows the connections between Logic Lock (Standard) regions. By default, you can view each connection as an individual line. You can choose to display connections between two Logic Lock (Standard) regions as a single bundled connection rather than as individual connection lines. To use this option, open the Chip Planner and on the View menu, click **Inter-region Bundles**.

Related Links

[Inter-region Bundles Dialog Box](#)

For more information about the Inter-region Bundles dialog box, refer to Intel Quartus Prime Help.

15.3.2 Using Logic Lock (Standard) Regions with the Design Partition Planner

You can optimize timing in a design by placing entities that share significant logical connectivity close to each other on the device.

By default, the Fitter usually places closely connected entities in the same area of the device; however, you can use Logic Lock (Standard) regions, together with the Design Partition Planner and the Chip Planner, to help ensure that logically connected entities retain optimal placement from one compilation to the next.



You can view the logical connectivity between entities with the Design Partition Planner, and the physical placement of those entities with the Chip Planner. In the Design Partition Planner, you can identify entities that are highly interconnected, and place those entities in a partition. In the Chip Planner, you can create Logic Lock (Standard) regions and assign each partition to a Logic Lock (Standard) region, thereby preserving the placement of the entities.

15.4 Scripting Support

You can run procedures and specify the settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt.

Related Links

- [Tcl Scripting](#) on page 81
- [API Functions for Tcl](#)
In Intel Quartus Prime Help
- [Intel Quartus Prime Standard Edition Settings File Reference Manual](#)

15.4.1 Initializing and Uninitializing a Logic Lock (Standard) Region

You must initialize the Logic Lock (Standard) data structures before creating or modifying any Logic Lock (Standard) regions and before executing any of the Tcl commands listed below.

Use the following Tcl command to initialize the Logic Lock (Standard) data structures:

```
initialize_logiclock
```

Use the following Tcl command to uninitialize the Logic Lock (Standard) data structures before closing your project:

```
uninitialize_logiclock
```

15.4.2 Creating or Modifying Logic Lock (Standard) Regions

Use the following Tcl command to create or modify a Logic Lock (Standard) region:

```
set_logiclock -auto_size true -floating true -region <my_region-name>
```

Note:

The command in the above example sets the size of the region to auto and the state to floating.

If you specify a region name that does not exist in the design, the command creates the region with the specified properties. If you specify the name of an existing region, the command changes all properties you specify and leaves unspecified properties unchanged.

Related Links

[Creating Logic Lock \(Standard\) Regions](#) on page 292



15.4.3 Obtaining Logic Lock (Standard) Region Properties

Use the following Tcl command to obtain Logic Lock (Standard) region properties. This example returns the height of the region named `my_region`:

```
get_logiclock -region my_region -height
```

15.4.4 Assigning Logic Lock (Standard) Region Content

Use the following Tcl commands to assign or change nodes and entities in a Logic Lock (Standard) region. This example assigns all nodes with names matching `fifo*` to the region named `my_region`.

```
set_logiclock_contents -region my_region -to fifo*
```

You can also make path-based assignments with the following Tcl command:

```
set_logiclock_contents -region my_region -from fifo -to ram*
```

15.4.5 Save a Node-Level Netlist for the Entire Design into a Persistent Source File

Make the following assignments to cause the Intel Quartus Prime Fitter to save a node-level netlist for the entire design into a `.vqm` file:

```
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT ON  
set_global_assignment-name LOGICLOCK_INCREMENTAL_COMPILE_FILE <file name>
```

Any path specified in the file name is relative to the project directory. For example, specifying `atom_netlists/top.vqm` places `top.vqm` in the **atom_netlists** subdirectory of your project directory.

A `.vqm` file is saved in the directory specified at the completion of a full compilation.

Note: The saving of a node-level netlist to a persistent source file is not supported for designs targeting newer devices such as MAX V, Stratix IV, or Stratix V.

15.4.6 Setting Logic Lock (Standard) Assignment Priority

Use the following Tcl code to set the priority for a Logic Lock (Standard) region's members. This example reverses the priorities of the Logic Lock (Standard) region in your design.

```
set reverse [list]  
for each member [get_logiclock_member_priority] {  
    set reverse [insert $reverse 0 $member]  
}  
set_logiclock_member_priority $reverse
```

15.4.7 Assigning Virtual Pins with a Tcl command

Use the following Tcl command to turn on the virtual pin setting for a pin called `my_pin`:

```
set_instance_assignment -name VIRTUAL_PIN ON -to my_pin
```

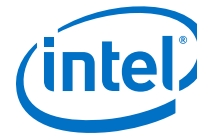
Related Links

- [Virtual Pins](#) on page 299
- [Managing Device I/O Pins](#) on page 24
- [Node Finder Command \(View Menu\)](#)
In Intel Quartus Prime Help

15.5 Document Revision History

Table 50. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> • Changed instances of <i>LogicLock</i> to <i>Logic Lock (Standard)</i>.
2017.05.08	17.0.0	<ul style="list-style-type: none"> • Chapter reorganization and content update. • Added figures: Clock Regions, Creating a Hole in a LogicLock Region, Noncontiguous LogicLock Region, Routing Regions, Logic Placed Outside of an Empty Region. • Moved topic: <i>Viewing Critical Paths</i> to <i>Timing Closure and Optimization</i> chapter and renamed to <i>Critical Paths</i>. • Renamed topic: <i>Creating Non-Rectangular LogicLock Plus Regions</i> to <i>Merging LogicLock Plus Regions</i>. • Renamed topic: <i>Chip Planner Overview</i> to <i>Design Floorplan Analysis in the Chip Planner</i>. • Renamed chapter from <i>Analyzing and Optimizing the Design Floorplan with the Chip Planner</i> to <i>Analyzing and Optimizing the Design Floorplan</i>.
2015.11.02	15.1.0	<ul style="list-style-type: none"> • Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
2015.05.04	15.0.0	Added information about color coding of LogicLock regions.
2014.12.15	14.1.0	Updated description of Virtual Pins assignment to clarify that assigned input is not available.
June 2014	14.0.0	Updated format
November 2013	13.1.0	Removed HardCopy device information.
May 2013	13.0.0	Updated "Viewing Routing Congestion" section Updated references to Quartus UI controls for the Chip Planner
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	<ul style="list-style-type: none"> • Updated for the 11.0 release. • Edited "LogicLock Regions" • Updated "Viewing Routing Congestion" • Updated "Locate History" • Updated Figures 15-4, 15-9, 15-10, and 15-13 • Added Figure 15-6
<i>continued...</i>		



Date	Version	Changes
December 2010	10.1.0	<ul style="list-style-type: none"> Updated for the 10.1 release.
July 2010	10.0.0	<ul style="list-style-type: none"> Updated device support information Removed references to Timing Closure Floorplan; removed "Design Analysis Using the Timing Closure Floorplan" section Added links to online Help topics Added "Using LogicLock Regions with the Design Partition Planner" section Updated "Viewing Critical Paths" section Updated several graphics Updated format of Document revision History table
November 2009	9.1.0	<ul style="list-style-type: none"> Updated supported device information throughout Removed deprecated sections related to the Timing Closure Floorplan for older device families. (For information on using the Timing Closure Floorplan with older device families, refer to previous versions of the Quartus Prime Handbook, available in the Documentation Archive.) Updated "Creating Nonrectangular LogicLock Regions" section Added "Selected Elements Window" section Updated table 12-1
May 2008	8.0.0	<ul style="list-style-type: none"> Updated the following sections: <ul style="list-style-type: none"> "Chip Planner Tasks and Layers" "LogicLock Regions" "Back-Annotating LogicLock Regions" "LogicLock Regions in the Timing Closure Floorplan" Added the following sections: <ul style="list-style-type: none"> "Reserve LogicLock Region" "Creating Nonrectangular LogicLock Regions" "Viewing Available Clock Networks in the Device" Updated Table 10-1 Removed the following sections: <ul style="list-style-type: none"> Reserve LogicLock Region Design Analysis Using the Timing Closure Floorplan

Related Links

Documentation Archive

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



16 Netlist Optimizations and Physical Synthesis

The Intel Quartus Prime software offers netlist and physical synthesis optimizations that improve performance of your design. Click to enable physical synthesis options during fitting. This chapter also provides guidelines for applying netlist and physical synthesis options, and for preserving compilation results through back-annotation.

Table 51. Netlist Optimization and Physical Synthesis Options

Options	Location/Description
Enable physical synthesis options.	Assignments > Settings > Compiler Settings > Advanced Settings (Fitter). Physical synthesis optimizations apply at different stages of the compilation flow, either during synthesis, fitting, or both.
Enable netlist optimization options.	Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis). Netlist optimizations operate with the atom netlist of your design, which describes a design in terms of specific primitives. An atom netlist file can be an Electronic Design Interchange Format (.edf) file or a Verilog Quartus Mapping (.vqm) file generated by a third-party synthesis tool. Intel Quartus Prime synthesis generates and internally uses the atom netlist internally.

Note: Because the node names for primitives in the design can change when you use physical synthesis optimizations, you should evaluate whether your design depends on fixed node names. If you use a verification flow that might require fixed node names, such as the Signal Tap Logic Analyzer, formal verification, or the Logic Lock (Standard) based optimization flow (for legacy devices), disable physical synthesis options.

16.1 Physical Synthesis Optimizations

The Intel Quartus Prime Fitter places and routes the logic cells to ensure critical portions of logic are close together and use the fastest possible routing resources. However, routing delays are often a significant part of the typical critical path delay. Physical synthesis optimizations take into consideration placement information, routing delays, and timing information to determine the optimal placement. The Fitter then focuses timing-driven optimizations at those critical parts of the design. The tight integration of the synthesis and fitting processes is known as physical synthesis.

To enable or disable physical synthesis options, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**.

The following sections describe the physical synthesis optimizations available in the Intel Quartus Prime software, and how they can help improve performance and fitting for the selected device.

Note: To disable global physical synthesis optimizations for specific elements of your design, assign the **Netlist Optimizations** logic option to **Never Allow** to the specific nodes or entities.



Some physical synthesis options affect only registered logic, while others affect only combinational logic. Select options based on whether you want to keep the registers intact. For example, if your verification flow involves formal verification, you might want to keep the registers intact.

Related Links

[Compiler Settings Page \(Settings Dialog Box\)](#)

16.1.1 Enabling Physical Synthesis Optimization

Physical synthesis optimization improves circuit performance by performing combinational and sequential optimization and register duplication.

To enable physical synthesis options, follow these steps:

1. Click **Assignments** > **Settings** > **Compiler Settings**.
2. To enable physical synthesis, click **Advanced Settings (Fitter)**, and then enable **Perform Physical Synthesis for Combinational Logic for Performance** and **Perform Physical Synthesis for Combinational Logic for Fitting**.
3. View physical synthesis results in the **Netlist Optimizations** report.

16.1.2 Physical Synthesis Options

The Intel Quartus Prime software provides physical synthesis optimization options to improve fitting results. To access these options, click **Assignments** > **Settings** > **Compiler Settings** > **Advanced Settings (Fitter)**.

Table 52. Physical Synthesis Options

Option	Description
Perform asynchronous signal pipelining (no Intel Arria 10 support)	Automatically inserts pipeline stages for asynchronous clear and asynchronous load signals during fitting to increase circuit performance. This option is useful for asynchronous signals that are failing recovery and removal timing because they feed registers using a high-speed clock. You can use this option if asynchronous control signal recovery and removal times are not achieving requirements. This option adds registers and potential latency to nets driving the asynchronous clear or asynchronous load ports of registers. The additional register delays can change the behavior of the signal in the design; therefore, you should use this option only if additional latency on the reset signals does not violate any design requirements. This option also prevents the promotion of signals to global routing resources.
Perform Register Duplication for Performance (no Intel Arria 10 support)	Duplicates registers based on Fitter placement information to reduce the delay of one path without degrading the delay of another. You can also duplicate combinational logic when you enable this option. The Fitter can place the new logic cell closer to critical logic without affecting the other fan-out paths of the original logic cell. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow .
Perform Register Retiming for Performance (no Arria 10 support)	Enables the movement of registers across combinational logic, allowing the Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths.
Perform Physical synthesis for combinational logic for Performance (no Intel Arria 10 support)	Performs physical synthesis optimizations on combinational logic during synthesis and fitting to increase circuit performance. Swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. Also allows the duplication of LUTs to enable further optimizations on the critical path.
Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support)	Reduces delay along critical paths. This option swaps the look-up table (LUT) ports within LEs so that the critical path has fewer layers through which to travel. The option also allows the duplication of LUTs to enable further optimizations on the critical path.

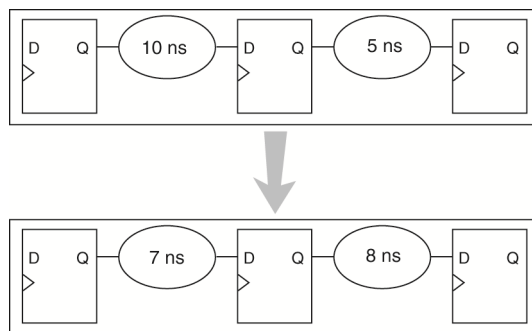
continued...

Option	Description
	The option causes registers that do not have a Power-Up Level logic option setting to power up with a don't care logic level (x). When the Power-Up Don't Care option is turned on, the Compiler determines when it is beneficial to change the power-up level of a register to minimize the area of the design. A power-up state of zero is maintained unless there is an immediate area advantage. The registers contained in the affected logic cells are not modified. Inputs into memory blocks, DSP blocks, and I/O elements (IOEs) are not swapped. This setting does not apply to logic cells that are part of a chain, drive global signals, are constrained to a single LAB, or the Netlist Optimizations option set to Never Allow .
Perform WYSIWYG Primitive Resynthesis	Specifies whether to perform WYSIWYG primitive resynthesis during synthesis. This option uses the setting specified in the Optimization Technique logic option.
Physical Synthesis Effort Level (no Intel Arria 10 support)	Specifies the amount of effort, in terms of compile time, physical synthesis should use. Compared to the Default setting, a setting of Extra uses extra compile time to try to gain extra circuit performance. Conversely, a setting of Fast uses less compile time but may reduce the performance gain that physical synthesis is able to achieve.
Netlist Optimizations	You can use the Assignment Editor to apply the Netlist Optimizations logic option. Use this option to disable physical synthesis optimizations for parts of your design.
Allow Register Duplication	Allows the Compiler to duplicate registers to improve design performance. When you enable this option, the Compiler copies registers and moves some fan-out to this new node. This optimization improves routability and can reduce the total routing wire in nets with many fan-outs. If you disable this option, this disables optimizations that retime registers. This setting affects Analysis & Synthesis and the Fitter.
Allow Register Merging	Allows the Compiler to remove registers that are identical to other registers in the design. When you enable this option, in cases where two registers generate the same logic, the Compiler deletes one register, and the remaining registers fan-out to the deleted register's destinations. This option is useful if you wish to prevent the Compiler from removing intentional use of duplicate registers. If you disable register merging, the Compiler disables optimizations that retime registers. This setting affects Analysis & Synthesis and the Fitter.

16.1.3 Perform Register Retiming for Performance

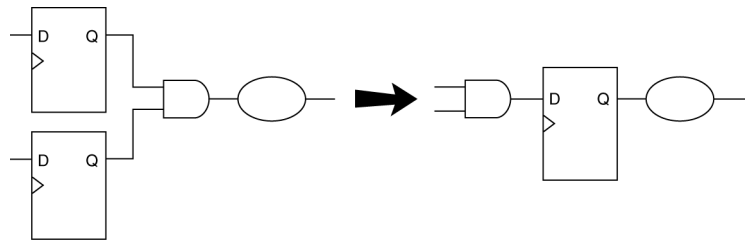
The **Perform Register Retiming for Performance** option enables the movement of registers across combinational logic, allowing the Intel Quartus Prime software to trade off the delay between timing-critical paths and non-critical paths. Register retiming can be done during Intel Quartus Prime integrated synthesis or during the Fitter stages of design compilation.

Figure 112. Reducing Critical Delay by Moving the Register Relative to Combinational Logic



Retiming can create multiple registers at the input of a combinational block from a register at the output of a combinational block. In this case, the new registers have the same clock and clock enable. The asynchronous control signals and power-up level are derived from previous registers to provide equivalent functionality. Retiming can also combine multiple registers at the input of a combinational block to a single register.

Figure 113. Combining Registers with Register Retiming



To move registers across combinational logic to balance timing, click **Assignments > Settings > Compiler Settings > Advanced Settings (Fitter)**. Specify your preferred option under **Optimize for performance (physical synthesis)** and **Effort level**.

16.1.4 Preventing Register Movement During Retiming

If you want to prevent register movement during register retiming, you can set the **Netlist Optimizations** logic option to **Never Allow**. You can apply this option to either individual registers or entities in the design using the Assignment Editor.

In digital circuits, synchronization registers are instantiated on cross clock domain paths to reduce the possibility of metastability. The Intel Quartus Prime software detects such synchronization registers and does not move them, even if register retiming is turned on.

The following sets of registers are not moved during register retiming:

- Both registers in a direct connection from input pin-to-register-to-register if both registers have the same clock and the first register does not fan-out to anywhere else. These registers are considered synchronization registers.
- Both registers in a direct connection from register-to-register if both registers have the same clock, the first register does not fan out to anywhere else, and the first register is fed by another register in a different clock domain (directly or through combinational logic). These registers are considered synchronization registers.

The Intel Quartus Prime software does not perform register retiming on logic cells that have the following properties:

- Are part of a cascade chain
- Contain registers that drive asynchronous control signals on another register
- Contain registers that drive the clock of another register
- Contain registers that drive a register in another clock domain
- Contain registers that are driven by a register in another clock domain

Note: The Intel Quartus Prime software does not usually retime registers across different clock domains; however, if you use the Classic Timing Analyzer and specify a global f_{MAX} requirement, the Intel Quartus Prime software interprets all clocks as related. Consequently, the Intel Quartus Prime software might try to retime register-to-register paths associated with different clocks.

To avoid this circumstance, provide individual f_{MAX} requirements to each clock when using Classic Timing Analysis. When you constrain each clock individually, the Intel Quartus Prime software assumes no relationship between different clock domains and considers each clock domain to be asynchronous to other clock domains; hence no register-to-register paths crossing clock domains are retimed.

When you use the Timing Analyzer, register-to-register paths across clock domains are never retimed, because the Timing Analyzer treats all clock domains as asynchronous to each other unless they are intentionally grouped.

- Contain registers that are constrained to a single LAB location
- Contain registers that are connected to SERDES
- Are considered virtual I/O pins
- Registers that have the **Netlist Optimizations** logic option set to **Never Allow**

The Intel Quartus Prime software assumes that a synchronization register chain consists of two registers. If your design has synchronization register chains with more than two registers, you must indicate the number of registers in your synchronization chains so that they are not affected by register retiming. To do this, perform the following steps:

1. Click **Assignments > Settings > Compiler Settings > Advanced Settings (Synthesis)**.
2. Modify the **Synchronization Register Chain Length** setting to match the synchronization register length used in your design. If you set a value of 1 for the **Synchronization Register Chain Length**, it means that any registers connected to the first register in a register-to-register connection can be moved during retiming. A value of $n > 1$ means that any registers in a sequence of length 1, 2, ... n are not moved during register retiming.

If you want to consider logic cells that meet any of these conditions for physical synthesis, you can override these rules by setting the **Netlist Optimizations** logic option to **Always Allow** on a given set of registers.

Related Links

[Analyzing and Optimizing the Design Floorplan](#) on page 280

16.2 Applying Netlist Optimizations

The improvement in performance when using netlist optimizations is design dependent. If you have restructured your design to balance critical path delays, netlist optimizations might yield minimal improvement in performance.

You may have to experiment with available options to see which combination of settings works best for a particular design. Refer to the messages in the compilation report to see the magnitude of improvement with each option, and to help you decide whether you should turn on a given option or specific effort level.



Turning on more netlist optimization options can result in more changes to the node names in the design; bear this in mind if you are using a verification flow, such as the Signal Tap Logic Analyzer or formal verification that requires fixed or known node names.

Applying all the physical synthesis options at the **Extra** effort level generally produces the best results for those options, but adds significantly to the compilation time. You can also use the **Physical synthesis effort level** options to decrease the compilation time. The WYSIWYG primitive resynthesis option does not add much compilation time relative to the overall design compilation time.

To find the best results, you can use the Intel Quartus Prime Design Space Explorer II (DSE) to apply various sets of netlist optimization options.

Related Links

- [Design Space Explorer II](#) on page 249
- [Optimizing with Design Space Explorer II](#)
In *Intel Quartus Prime Standard Edition Handbook Volume 1*

16.2.1 WYSIWYG Primitive Resynthesis

If you use a third-party tool to synthesize your design, use the **Perform WYSIWYG primitive resynthesis** option to apply optimizations to the synthesized netlist.

The **Perform WYSIWYG primitive resynthesis** option directs the Intel Quartus Prime software to un-map the logic elements (LEs) in an atom netlist to logic gates, and then re-map the gates back to Intel-specific primitives. Third-party synthesis tools generate either an `.edf` or `.vqm` atom netlist file using Intel-specific primitives. When you turn on the **Perform WYSIWYG primitive resynthesis** option, the Intel Quartus Prime software uses device-specific techniques during the re-mapping process. This feature re-maps the design using the **Optimization Technique** specified for your project (**Speed**, **Area**, or **Balanced**).

The **Perform WYSIWYG primitive resynthesis** option unmaps and remaps only logic cells, also referred to as LCELL or LE primitives, and regular I/O primitives (which may contain registers). Double data rate (DDR) I/O primitives, memory primitives, digital signal processing (DSP) primitives, and logic cells in carry/cascade chains are not remapped. This process does not process logic specified in an encrypted `.vqm` file or an `.edf` file, such as third-party intellectual property (IP).

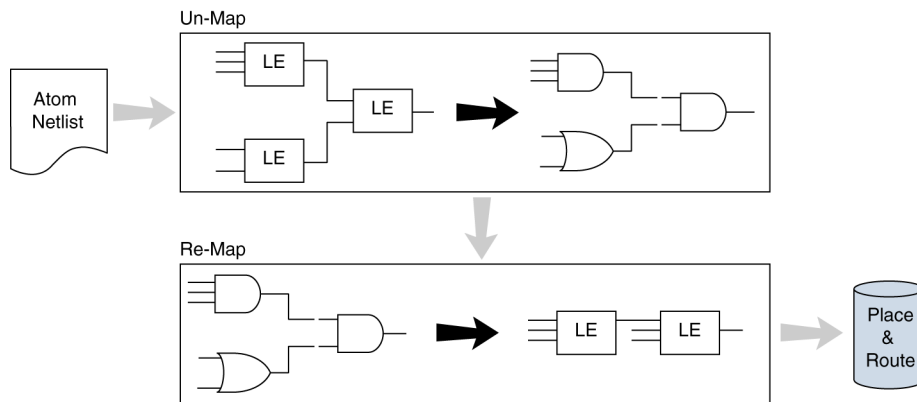
The **Perform WYSIWYG primitive resynthesis** option can change node names in the `.vqm` file or `.edf` file from your third-party synthesis tool, because the primitives in the atom netlist are broken apart and then re-mapped by the Intel Quartus Prime software. The re-mapping process removes duplicate registers. Registers that are not removed retain the same name after re-mapping.

Any nodes or entities that have the **Netlist Optimizations** logic option set to **Never Allow** are not affected during WYSIWYG primitive resynthesis. You can use the Assignment Editor to apply the **Netlist Optimizations** logic option. This option disables WYSIWYG resynthesis for parts of your design.

Note: Primitive node names are specified during synthesis. When netlist optimizations are applied, node names might change because primitives are created and removed. HDL attributes applied to preserve logic in third-party synthesis tools cannot be maintained because those attributes are not written into the atom netlist, which the Intel Quartus Prime software reads.

If you use the Intel Quartus Prime software to synthesize your design, you can use the **Preserve Register (preserve)** and **Keep Combinational Logic (keep)** attributes to maintain certain nodes in the design.

Figure 114. Intel Quartus Prime Flow for WYSIWYG Primitive Resynthesis



16.2.2 Saving a Node-Level Netlist

For non-Intel Arria 10 designs, you can preserve a node-level netlist in Verilog Quartus Mapping File (.vqm) format. You might need to preserve nodes if you use the Logic Lock (Standard) flow to back-annotate placement, import one design into another, or both. For all device families that support incremental compilation, you can use this feature to preserve compilation results.

Note: This feature does not support Intel Arria 10 devices.

Use the **Export version-compatible database** option to save synthesis results as an atom-based netlist in .vqm file format. By default, the Intel Quartus Prime software places the .vqm in the **atom_netlists** directory under the current project directory.

If you use the physical synthesis optimizations and want to lock down the location of all LEs and other device resources in the design with the **Back-Annotate Assignments** command, a .vqm file netlist is required. The .vqm file preserves the changes that you made to your original netlist. Because the physical synthesis optimizations depend on the placement of the nodes in the design, back-annotating the placement changes the results from physical synthesis. Changing the results means that node names are different, and your back-annotated locations are no longer valid.

You should not use an Intel Quartus Prime-generated .vqm file or back-annotated location assignments with physical synthesis optimizations unless you have finalized the design. Making any changes to the design invalidates your physical synthesis



results and back-annotated location assignments. If you require changes later, use the new source HDL code as your input files, and remove the back-annotated assignments corresponding to the Intel Quartus Prime-generated `.vqm` file.

To back-annotate logic locations for a design that was compiled with physical synthesis optimizations, first create a `.vqm` file. When recompiling the design with the hard logic location assignments, use the new `.vqm` file as the input source file and turn off the physical synthesis optimizations for the new compilation.

If you are importing a `.vqm` file and back-annotated locations into another project that has any **Netlist Optimizations** turned on, you must apply the **Never Allow** constraint to make sure node names don't change; otherwise, the back-annotated location or Logic Lock (Standard) assignments are invalid.

To preserve the nodes from Intel Quartus Prime physical synthesis optimization options for devices that do not support incremental compilation, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Compilation Process Settings**. The **Compilation Process Settings** page appears.
3. Turn on **Export version-compatible database**. This setting is not available for some devices.
4. Click **OK**.

16.3 Viewing Synthesis and Netlist Optimization Reports

Physical synthesis optimizations performed during synthesis write results to the synthesis report. To access this report, perform the following steps:

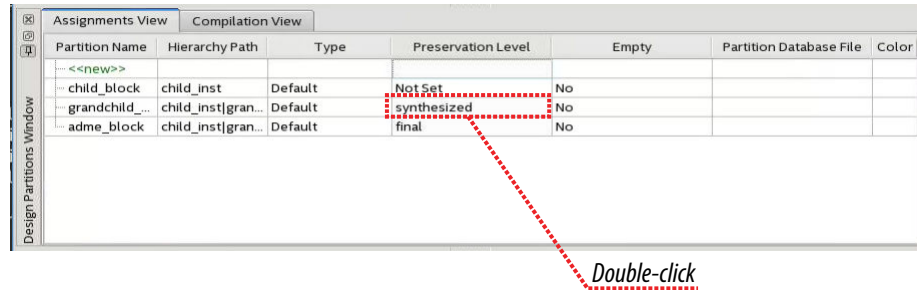
1. On the Processing menu, click **Compilation Report**.
2. In the **Compilation Report** list, open the **Analysis & Synthesis** folder to view synthesis results.
3. In the **Compilation Report** list, open the **Fitter** folder to view the **Netlist Optimizations** table.

16.4 Isolating a Partition Netlist

To make changes to one partition without modifying the rest of the design:

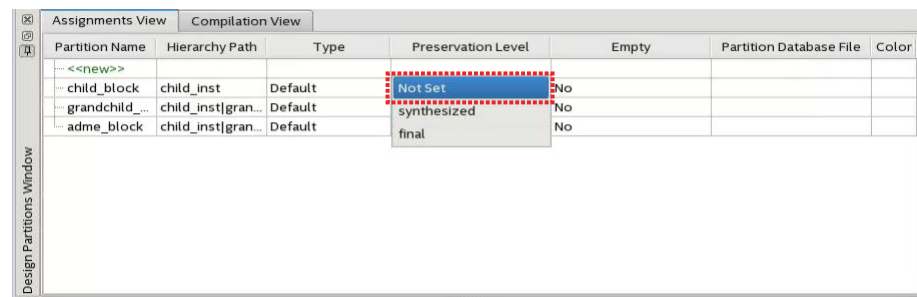
1. Click **Assignments** ► **Design Partitions Window**.
2. In the Design Partitions window, double-click the **Preservation Level** cell for the partition with the worst slack value.

Figure 115. Preservation Level in Design Partitions Window



3. Choose **Not Set**.

Figure 116. Setting a Partition's Preservation Level to Not Set



4. For every other partition in the Design Partitions window, set the preservation level to **Empty**.

After making changes in the selected partition, set the partition's preservation level to **final**. This action fixes the partition's location, and preserves the gains achieved as you continue to optimize other partitions.

Related Links

- [Using Partitions to Achieve Timing Closure](#) on page 215
- [Viewing Design Connectivity and Hierarchy](#)
- [Using Block-Based Compilation](#) on page 195

16.5 Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Intel Quartus Prime Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```

You can specify many of the options described in this section on either an instance or global level, or both.



Use the following Tcl command to make a global assignment:

```
set_global_assignment -name <QSF variable name> <value>
```

Use the following Tcl command to make an instance assignment:

```
set_instance_assignment -name <QSF variable name> <value> \  
-to <instance name>
```

Related Links

- [Tcl Scripting](#) on page 81
- [Command Line Scripting](#) on page 67
- [API Functions for Tcl](#)
In *Intel Quartus Prime Help*
- [Intel Quartus Prime Standard Edition Settings File Reference Manual](#)
For information about all settings and constraints in the Intel Quartus Prime software.

16.5.1 Synthesis Netlist Optimizations

The project `.qsf` file preserves the settings that you specify in the GUI. Alternatively, you can edit the `.qsf` directly. The `.qsf` file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.

Table 53. Synthesis Netlist Optimizations and Associated Settings

Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Perform WYSIWYG Primitive Resynthesis	ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP	ON, OFF	Global, Instance
Optimization Mode	OPTIMIZATION_MODE	BALANCEDHIGH PERFORMANCE EFFOR AGGRESSIVE PERFORMANCE	Global, Instance
Power-Up Don't Care	ALLOW_POWER_UP_DONT_CARE	ON, OFF	Global
Save a node-level netlist into a persistent source file	LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT	ON, OFF	Global
	LOGICLOCK_INCREMENTAL_COMPILE_FILE	<file name>	
Allow Netlist Optimizations	ADV_NETLIST_OPT_ALLOWED	"ALWAYS ALLOW", DEFAULT, "NEVER ALLOW"	Instance

16.5.2 Physical Synthesis Optimizations

The project `.qsf` file preserves the settings that you specify in the GUI. Alternatively, you can edit the `.qsf` directly. The `.qsf` file supports the following synthesis netlist optimization commands. The **Type** column indicates whether the setting is supported as a global setting, an instance setting, or both.



Table 54. Physical Synthesis Optimizations and Associated Settings

Setting Name	Intel Quartus Prime Settings File Variable Name	Values	Type
Perform Physical Synthesis for Combinational Logic for Performance (no Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC	ON, OFF	Global
Perform Physical Synthesis for Combinational Logic for Fitting (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA	ON, OFF	Global
Advanced Physical Synthesis	ADVANCED_PHYSICAL_SYNTHESIS	ON, OFF	Global
Automatic Asynchronous Signal Pipelining	PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING	ON, OFF	Global
Perform Register Duplication for Performance (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION	ON, OFF	Global
Perform Register Retiming for Performance (no Intel Arria 10 support)	PHYSICAL_SYNTHESIS_REGISTER_RETIMING	ON, OFF	Global
Power-Up Don't Care	ALLOW_POWER_UP_DONT_CARE	ON, OFF	Global, Instance
Power-Up Level	POWER_UP_LEVEL	HIGH, LOW	Instance
Allow Netlist Optimizations	ADV_NETLIST_OPT_ALLOWED	"ALWAYS ALLOW", DEFAULT, "NEVER ALLOW"	Instance
Save a node-level netlist into a persistent source file (no Intel Arria 10 support)	LOGICLOCK_INCREMENTAL_COMPILE_ASSIGNMENT	ON, OFF	Global
	LOGICLOCK_INCREMENTAL_COMPILE_FILE	<file name>	

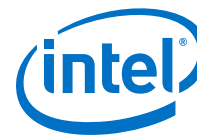
16.5.3 Back-Annotating Assignments

You can use the `logiclock_back_annotate` Tcl command to back-annotate resources in your design. This command can back-annotate resources in Logic Lock (Standard) regions, and resources in designs without Logic Lock (Standard) regions.

The following Tcl command back-annotates all registers in your design:

```
logiclock_back_annotate -resource_filter "REGISTER"
```

The `logiclock_back_annotate` command is in the `backannotate` package.



16.6 Document Revision History

Table 55. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> Added topic: Isolating a Partition Netlist.
2016.10.31	16.1.0	<ul style="list-style-type: none"> Updated physical synthesis options and procedure.
2016.05.02	16.0.0	<ul style="list-style-type: none"> Stated limitations about deprecated physical synthesis options.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
2014.12.15	14.1.0	<ul style="list-style-type: none"> Updated location of Fitter Settings, Analysis & Synthesis Settings, and Physical Synthesis Optimizations Settings to Compiler Settings. Updated DSE II content.
June 2014	14.0.0	Updated format.
November 2013	13.1.0	Removed HardCopy device information.
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	<ul style="list-style-type: none"> Added links to Intel Quartus Prime Help in several sections. Removed Referenced Documents section. Reformatted Document Revision History
November 2009	9.1.0	<ul style="list-style-type: none"> Added information to "Physical Synthesis for Registers—Register Retiming" Added information to "Applying Netlist Optimization Options" Made minor editorial updates
March 2009	9.0.0	<ul style="list-style-type: none"> Was chapter 11 in the 8.1.0 release. Updated the "Physical Synthesis for Registers—Register Retiming" and "Physical Synthesis Options for Fitting" Updated "Performing Physical Synthesis Optimizations" Deleted Gate-Level Register Retiming section. Updated the referenced documents
November 2008	8.1.0	Changed to 8½" × 11" page size. No change to content.
May 2008	8.0.0	<ul style="list-style-type: none"> Updated "Physical Synthesis Optimizations for Performance on page 11-9" Added Physical Synthesis Options for Fitting on page 11-16

Related Links

Documentation Archive

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



17 Engineering Change Orders with the Chip Planner

Programmable logic can accommodate changes to a system specification late in the design cycle. In a typical engineering project development cycle, the specification of the programmable logic portion is likely to change after engineering begins or while integrating all system elements. Last-minute design changes, commonly referred to as engineering change orders (ECOs), are small targeted changes to the functionality of a design after the design has been fully compiled.

The Chip Planner supports ECOs by allowing quick and efficient changes to your logic late in the design cycle. The Chip Planner provides a visual display of your post-place-and-route design mapped to the device architecture of your chosen FPGA and allows you to create, move, and delete logic cells and I/O atoms.

Note:

In addition to making ECOs, the Chip Planner allows you to perform detailed analysis on routing congestion, relative resource usage, logic placement, Logic Lock (Standard) regions, fan-ins and fan-outs, paths between registers, and delay estimates for paths.

ECOs directly apply to atoms in the target device. As such, performing an ECO relies on your understanding of the device architecture of the target device.

Related Links

- [Analyzing and Optimizing the Design Floorplan](#) on page 280
For more information about using the Chip Planner for design analysis
- [Literature](#)
For more information about the architecture of your device

17.1 Engineering Change Orders

In the context of an FPGA design, you can apply an ECO directly to a physical resource on the device to modify its behavior. ECOs are typically made during the verification stage of a design cycle. When a small change is required on a design (such as modifying a PLL for a different clock frequency or routing a signal out to a pin for analysis) recompilation of the entire design can be time consuming, especially for larger designs.

Because several iterations of small design changes can occur during the verification cycle, recompilation times can quickly add up. Furthermore, a full recompilation due to a small design change can result in the loss of previous design optimizations. Making ECOs, instead of performing a full recompilation on your design, limits the change only to the affected portions of logic.

17.1.1 Performance Preservation

You can preserve the results of previous design optimizations when you make changes to an existing design with one of the following methods:

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2008
Registered



- Incremental compilation
- Rapid recompile
- ECOs

Choose the method to modify your design based on the scope of the change. The methods above are arranged from the larger scale change to the smallest targeted change to a compiled design.

The incremental compilation feature allows you to preserve compilation results at an RTL component or module level. After the initial compilation of your design, you can assign modules in your design hierarchy to partitions. Upon subsequent compilations, incremental compilation recompiles changed partitions based on the chosen preservation levels.

The rapid recompilation feature leverages results from the latest post-fit netlist to determine the changes required to honor modifications you have made to the source code. If you run a rapid recompilation, the Compiler refits only changed portion of the netlist.

ECOs provide a finer granularity of control compared to the incremental compilation and the rapid recompilation feature. All modifications are performed directly on the architectural elements of the device. You should use ECOs for targeted changes to the post-fit netlist.

Note: In the Intel Quartus Prime software versions 10.0 and later, the software does not preserve ECO modifications to the netlist when you recompile a design with the incremental compilation feature turned on. You can reapply ECO changes made during a previous compilation with the Change Manager.

Related Links

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design](#)

17.1.2 Compilation Time

In the traditional programmable logic design flow, a small change in the design requires a complete recompilation of the design. A complete recompilation of the design consists of synthesis and place-and-route. Making small changes to the design to reach the final implementation on a board can be a long process. Because the Chip Planner works only on the post-place-and-route database, you can implement your design changes in minutes without performing a full compilation.

17.1.3 Verification

After you make a design change, you can verify the impact on your design. To verify that your changes do not violate timing requirements, perform static timing analysis with the Intel Quartus Prime Timing Analyzer after you check and save your netlist changes in the Chip Planner.

Additionally, you can perform a gate-level or timing simulation of the ECO-modified design with the post-place-and-route netlist generated by the Intel Quartus Prime software.

Related Links

[Intel Quartus Prime Timing Analyzer](#)



17.1.4 Change Modification Record

All ECOs made with the Chip Planner are logged in the Change Manager to track all changes. With the Change Manager, you can easily revert to the original post-fit netlist or you can pick and choose which ECOs to apply.

Additionally, the Intel Quartus Prime software provides support for multiple compilation revisions of the same project. You can use ECOs made with the Chip Planner in conjunction with revision support to compare several different ECO changes and revert back to previous project revisions when required.

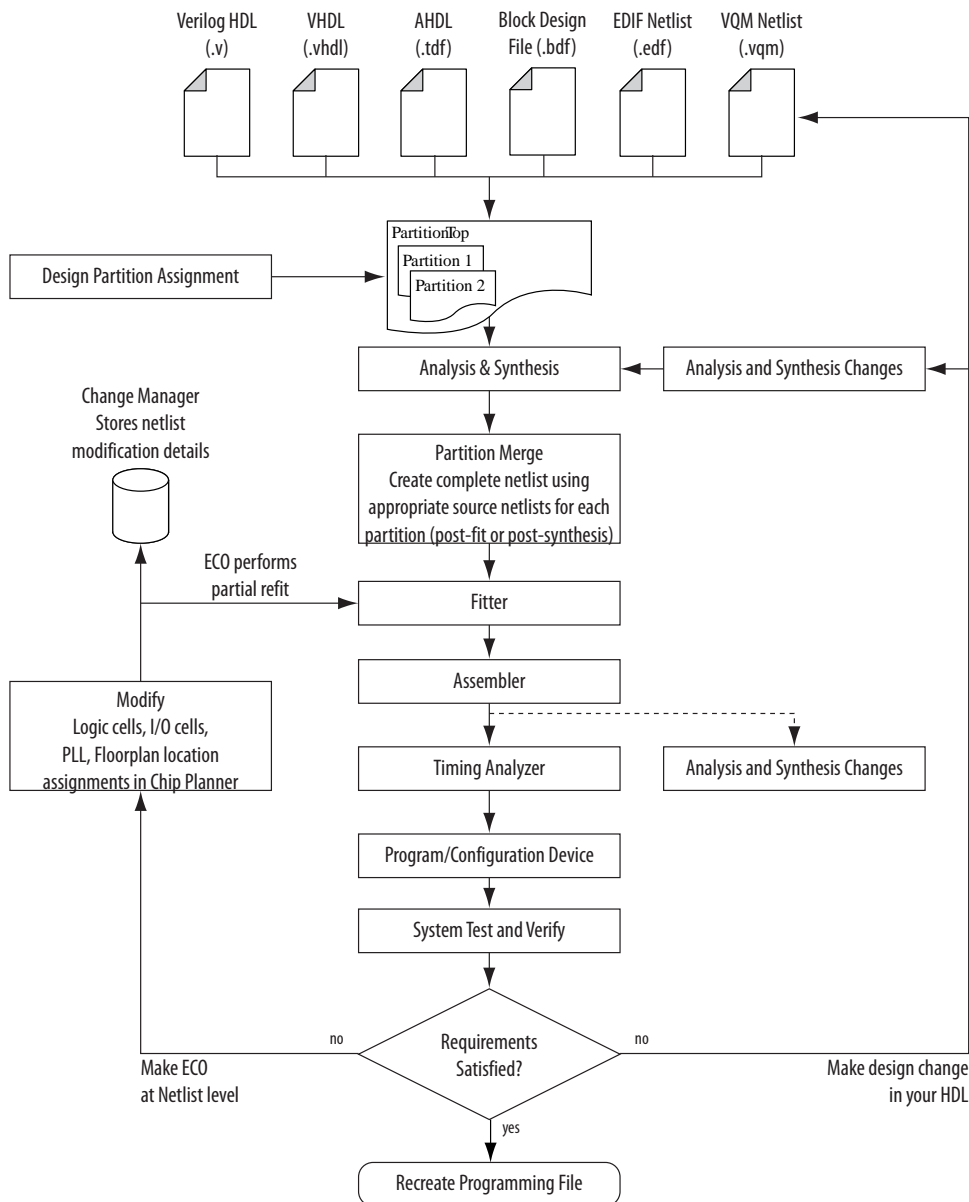
17.2 ECO Design Flow

For iterative verification cycles, implementing small design changes at the netlist level can be faster than making an RTL code change. As such, making ECO changes are especially helpful when you debug the design on silicon and require a fast turnaround time to generate a programming file for debugging the design.

The figure shows the design flow for making ECOs.



Figure 117. Design Flow to Support ECOs



A typical ECO application occurs when you uncover a problem on the board and isolate the problem to the appropriate nodes or I/O cells on the device. You must be able to correct the functionality quickly and generate a new programming file. By making small changes with the Chip Planner, you can modify the post-place-and-route netlist directly without having to perform synthesis and logic mapping, thus decreasing the turnaround time for programming file generation during the verification cycle. If the



change corrects the problem, no modification of the HDL source code is necessary. You can use the Chip Planner to perform the following ECO-related changes to your design:

- Document the changes made with the Change Manager
- Easily recreate the steps taken to produce design changes
- Generate EDA simulation netlists for design verification

Note: For more complex changes that require HDL source code modifications, the incremental compilation feature can help reduce recompilation time.

17.3 The Chip Planner Overview

The Chip Planner provides a visual display of device resources. It shows the arrangement and usage of the resource atoms in the device architecture that you are targeting. Resource atoms are the building blocks for your device, such as ALMs, LEs, PLLs, DSP blocks, memory blocks, or I/O elements.

The Chip Planner also provides an integrated platform for design analysis and for making ECOs to your design after place-and-route. The toolset consists of the Chip Planner (providing a device floorplan view of your mapped design) and two integrated subtools—the Resource Property Editor and the Change Manager.

For analysis, the Chip Planner can show logic placement, Logic Lock (Standard) regions, relative resource usage, detailed routing information, routing congestion, fan-ins and fan-outs, paths between registers, and delay estimates for paths. Additionally, the Chip Planner allows you to create location constraints or resource assignment changes, such as moving or deleting logic cells or I/O atoms with the device floorplan. For ECO changes, the Chip Planner enables you to create, move, or delete logic cells in the post-place-and-route netlist for fast programming file generation. Additionally, you can open the Resource Property Editor from the Chip Planner to edit the properties of resource atoms or to edit the connections between resource atoms. All changes to resource atoms and connections are logged automatically with the Change Manager.

17.3.1 Opening the Chip Planner

To open the Chip Planner, on the Tools menu, click **Chip Planner**. Alternatively, click the **Chip Planner** icon on the Intel Quartus Prime software toolbar.

Optionally, you can open the Chip Planner by cross-probing from the shortcut menu in the following tools:

- Design Partition Planner
- Compilation Report
- Logic Lock (Standard) Regions window
- Technology Map Viewer
- Project Navigator window
- RTL source code
- Node Finder



- Simulation Report
- RTL Viewer
- Report Timing panel of the Timing Analyzer

17.3.2 The Chip Planner Tasks and Layers

The Chip Planner allows you to set up tasks to quickly implement ECO changes or manipulate assignments for the floorplan of the device. Each task consists of an editing mode and a set of customized layer settings.

Related Links

- [Performing ECOs in the Resource Property Editor](#) on page 323
- [Analyzing and Optimizing the Design Floorplan](#) on page 280

17.4 Performing ECOs with the Chip Planner (Floorplan View)

You can manipulate resource atoms in the Chip Planner when you select the ECO editing mode.

The following ECO changes can be made with the Chip Planner Floorplan view:

- Create atoms
- Delete atoms
- Move existing atoms

Note: To configure the properties of atoms, such as managing the connections between different LEs/ALMs, use the Resource Property Editor.

To select the ECO editing mode in the Chip Planner, in the **Editing Mode** list at the top of the Chip Planner, select the ECO editing mode.

Related Links

[Performing ECOs in the Resource Property Editor](#) on page 323

17.4.1 Creating, Deleting, and Moving Atoms

You can use the Chip Planner to create, delete, and move atoms in the post-compilation design.

17.4.2 Check and Save Netlist Changes

After making all the ECOs, you can run the Fitter to incorporate the changes by clicking the **Check and Save Netlist Changes** icon in the Chip Planner toolbar. The Fitter compiles the ECO changes, performs design rule checks on the design, and generates a programming file.

17.5 Performing ECOs in the Resource Property Editor

You can view and edit the following resources with the Resource Property Editor.

17.5.1 Logic Elements

An Altera® LE contains a four-input LUT, which is a function generator that can implement any function of four variables. In addition, each LE contains a register fed by the output of the LUT or by an independent function generated in another LE.

You can use the Resource Property Editor to view and edit any LE in the FPGA. To open the Resource Property Editor for an LE, on the Project menu, point to **Locate**, and then click **Locate in Resource Property Editor** in one of the following views:

- RTL Viewer
- Technology Map Viewer
- Node Finder
- Chip Planner

For more information about LE architecture for a particular device family, refer to the device family handbook or data sheet.

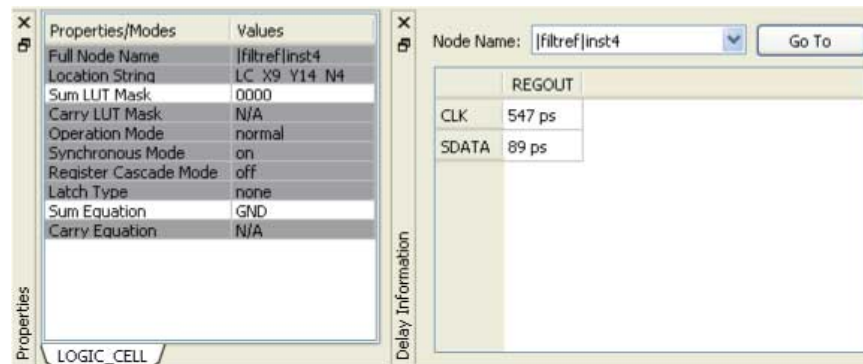
You can use the Resource Property Editor to change the following LE properties:

- Data input to the LUT
- LUT mask or LUT

17.5.1.1 Logic Element Properties

To view logic element properties, on the View menu, click **View Properties**.

Figure 118. LE Properties in the Resource Property Editor



17.5.1.2 Modes of Operation

LUTs in an LE can operate in either normal or arithmetic mode.

When an LE is configured in normal mode, the LUT in the LE can implement a function of four inputs.

When the LE is configured in arithmetic mode, the LUT in the LE is divided into two 3-input LUTs. The first LUT generates the signal that drives the output of the LUT, while the second LUT generates the carry-out signal. The carry-out signal can drive only a carry-in signal of another LE.

For more information about LE modes of operation, refer to volume 1 of the appropriate device handbook.



17.5.1.3 Sum and Carry Equations

You can change the logic function implemented by the LUT by changing the sum and carry equations. When the LE is configured in normal mode, you can change only the sum equation. When the LE is configured in arithmetic mode, you can change both the sum and the carry equations.

The LUT mask is the hexadecimal representation of the LUT equation output. When you change the LUT equation, the Intel Quartus Prime software automatically changes the LUT mask. Conversely, when you change the LUT mask, the Intel Quartus Prime software automatically computes the LUT equation.

17.5.1.4 sload and sclr Signals

Each LE register contains a synchronous load (`sload`) signal and a synchronous clear (`sclr`) signal. You can invert either the `sload` or `sclr` signal feeding into the LE.

If the design uses the `sload` signal in an LE, the signal and its inversion state must be the same for all other LEs in the same LAB. For example, if two LEs in a LAB have the `sload` signal connected, both LEs must have the `sload` signal set to the same value. This is also true for the `sclr` signal.

17.5.1.5 Register Cascade Mode

When register cascade mode is enabled, the cascade-in port feeds the input to the register. The register cascade mode is used most often when the design implements shift registers.

You can change the register cascade mode by connecting (or disconnecting) the cascade in the port. However, if you create this port, you must ensure that the source port LE is directly above the destination LE.

17.5.1.6 Cell Delay Table

The cell delay table describes the propagation delay from all inputs to all outputs for the selected LE.

17.5.1.7 Logic Element Connections

To view the connections that feed in and out of an LE, on the View menu, click **View Port Connections**.

Figure 119. View LE Connections in the Connectivity Window

Input Port Name	Signal Name	Inverted	Output Port Name	Signal Name
DATAA	<Disconnected>	False	COUT	<Disconnected>
DATAB	<Disconnected>	False	COMBOUT	<Disconnected>
SDATA	filter state_m:inst1 filter.tap4	False	REGOUT	filter inst4
DATAD	<Disconnected>	False		
CIN	<Disconnected>	False		
INVERTA	<Disconnected>	False		
REGCASCIN	<Disconnected>	False		
SLOAD	VCC	False		
SCLR	<Disconnected>	False		
IACLR	VCC	False		
ALOAD	<Disconnected>	False		
CLK	filter clk	False		
ENA	<Disconnected>	False		

17.5.1.8 Deleting a Logic Element

To delete an LE, follow these steps:

1. Right-click the desired LE in the Chip Planner, point to **Locate**, and click **Locate in Resource Property Editor**.
2. You must remove all fan-out connections from an LE prior to deletion. To delete fan-out connections, right-click each connected output signal, point to **Remove**, and click **Fanouts**. Select all of the fan-out signals in the **Remove Fan-outs** dialog box and click **OK**.
3. To delete an atom after all fan-out connections are removed, right-click the atom in the Chip Planner and click **Delete Atom**.

17.5.2 Adaptive Logic Modules

Each ALM contains LUT-based resources that can be divided between two adaptive LUTs (ALUTs).

With up to eight inputs to the two ALUTs, each ALM can implement various combinations of two functions. This adaptability allows the ALM to be completely backward-compatible with four-input LUT architectures. One ALM can implement any function with up to six inputs and certain seven-input functions. In addition to the ALUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. The ALM can efficiently implement various arithmetic functions and shift registers with these dedicated resources.

You can implement the following types of functions in a single ALM:

- Two independent 4-input functions
- An independent 5-input function and an independent 3-input function
- A 5-input function and a 4-input function, if they share one input
- Two 5-input functions, if they share two inputs
- An independent 6-input function
- Two 6-input functions, if they share four inputs and share the same functions
- Certain 7-input functions

You can use the Resource Property Editor to change the following ALM properties:

- Data input to the LUT
- LUT mask or LUT equation

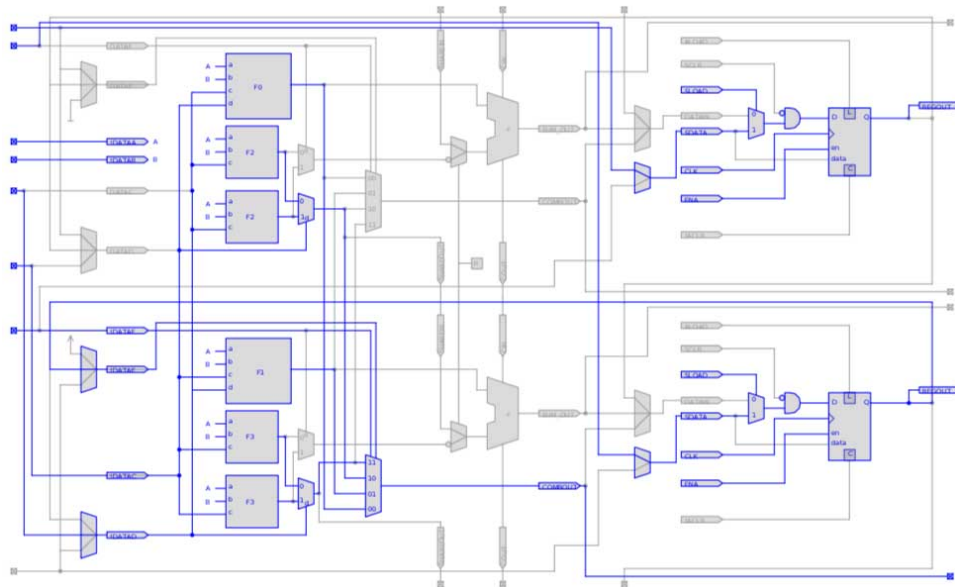
17.5.2.1 Adaptive Logic Module Schematic

You can view and edit any ALM atom with the Resource Property Editor by right-clicking the ALM in the RTL Viewer, the Node Finder, or the Chip Planner, and clicking **Locate in Resource Property Editor**.

For a detailed description of the ALM, refer to the device handbooks of devices based on an ALM architecture.

By default, the Intel Quartus Prime software displays the used resources in blue and the unused in gray. For the figure, the used resources are in blue and the unused resources are in gray.

Figure 120. Adaptive Logic Module



17.5.2.2 Adaptive Logic Module Properties

The properties that you can display for the ALM include an equations table that shows the name and location of each of the two combinational nodes and two register nodes in the ALM, the individual LUT equations for each of the combinational nodes, and the combout, sumout, carryout, and shareout equations for each combinational node.

17.5.2.3 Adaptive Logic Module Connections

Click **View > View Connectivity** to view the input and output connections for the ALM.

17.5.3 FPGA I/O Elements

Altera FPGAs that have high-performance I/O elements, including up to six registers, are equipped with support for a number of I/O standards that allow you to run your design at peak speeds. Use the Resource Property Editor to view, change connectivity, and edit the properties of the I/O elements. Use the Chip Planner (Floorplan view) to change placement, delete, and create new I/O elements.

For a detailed description of the device I/O elements, refer to the applicable device handbook.

You can change the following I/O properties:

- Delay chain
- Bus hold
- Weak pull up
- Slow slew rate

- I/O standard
- Current strength
- Extend OE disable
- PCI I/O
- Register reset mode
- Register synchronous reset mode
- Register power up
- Register mode

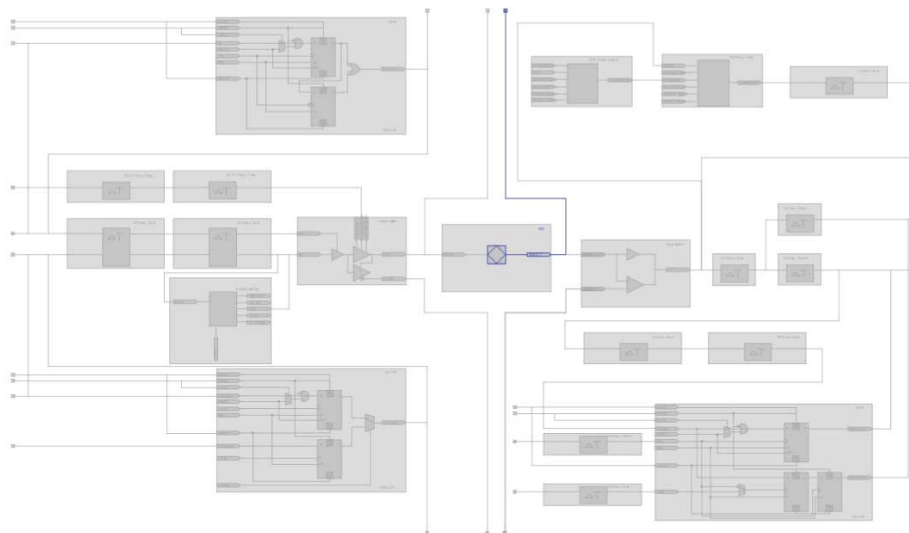
17.5.3.1 Stratix V I/O Elements

The I/O elements in Stratix® V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional single data rate (SDR) or double data rate (DDR) transfer.

I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal to the output buffer. These registers allow faster source-synchronous register-to-register transfers and resynchronization. The input path consists of the DDR input registers, alignment and synchronization registers, and half data rate blocks; you can bypass each block in the input path. The input path uses the deskew delay to adjust the input register clock delay across process, voltage, and temperature (PVT) variations.

By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 121. Stratix V Device I/O Element Structure



Related Links

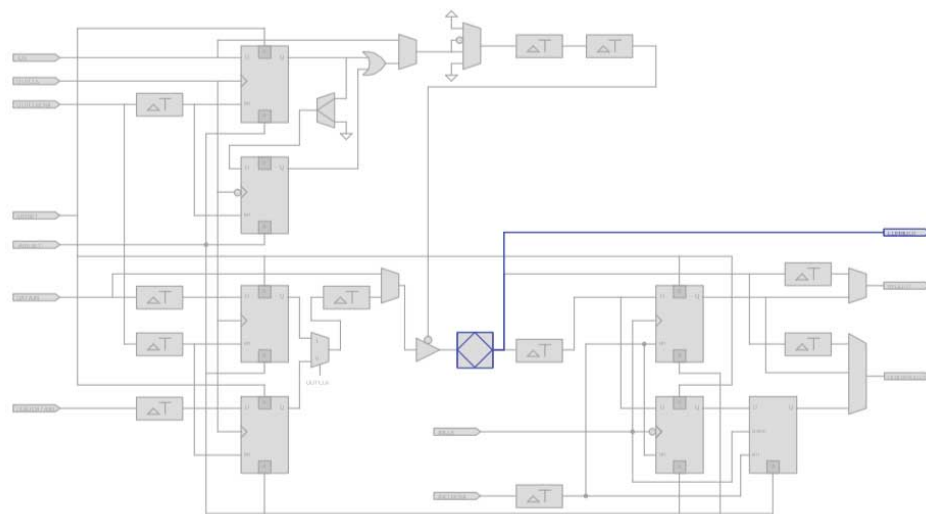
[Stratix V Device Handbook](#)

17.5.3.2 Stratix IV I/O Elements

The I/O elements in Stratix IV devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional SDR or DDR transfer.

The I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal for the output buffer. Each path consists of a set of delay elements that allow you to fine-tune the timing characteristics of each path for skew management. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 122. Stratix IV I/O Element and Structure



Related Links

Literature

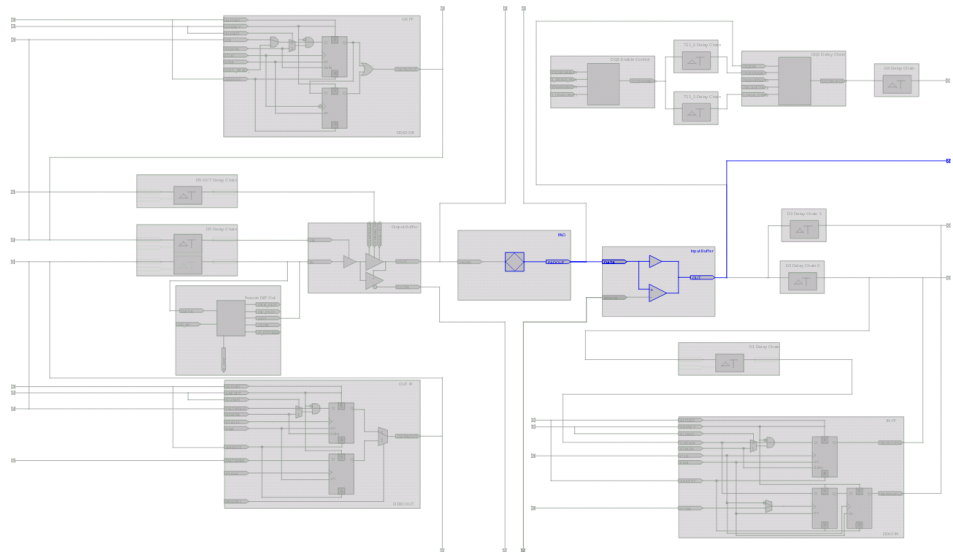
For more information about I/O elements in Stratix IV devices

17.5.3.3 Arria V I/O Elements

The I/O elements in Arria[®] V devices contain a bidirectional I/O buffer and I/O registers to support a complete embedded bidirectional SDR or DDR transfer.

The I/O registers are composed of the input path for handling data from the pin to the core, the output path for handling data from the core to the pin, and the output enable path for handling the output enable signal for the output buffer. Each path consists of a set of delay elements that allow you to fine-tune the timing characteristics of each path for skew management. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 123. Arria V Device I/O Element and Structure



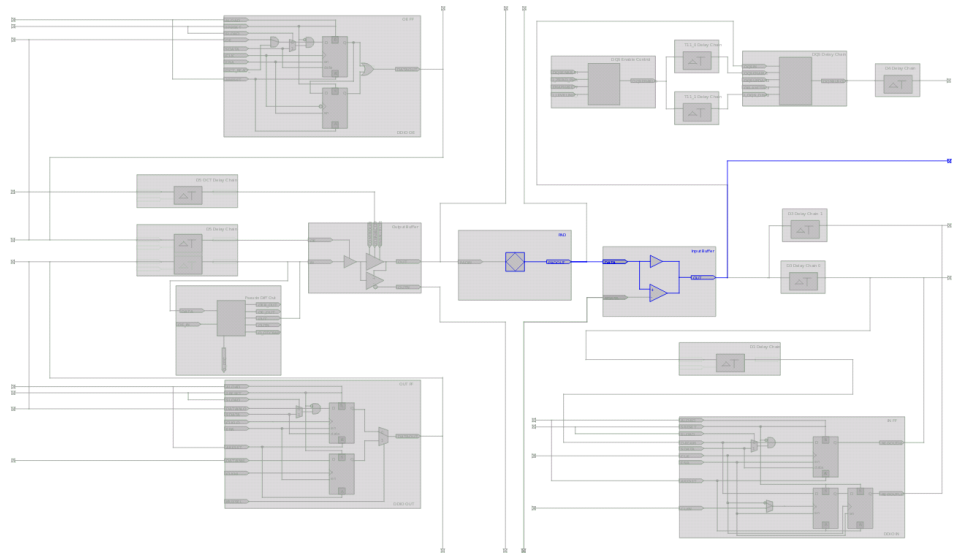
17.5.3.4 Cyclone V I/O Elements

The I/O elements in Cyclone V devices contain a bidirectional I/O buffer and registers for complete embedded bidirectional single data rate transfer. The I/O element contains three input registers, two output registers, and two output-enable registers. The two output registers and two output-enable registers are utilized for double-data rate (DDR) applications.

You can use the input registers for fast setup times and the output registers for fast clock-to-output times. Additionally, you can use the output-enable (OE) registers for fast clock-to-output enable timing. You can use I/O elements for input, output, or bidirectional data paths. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.



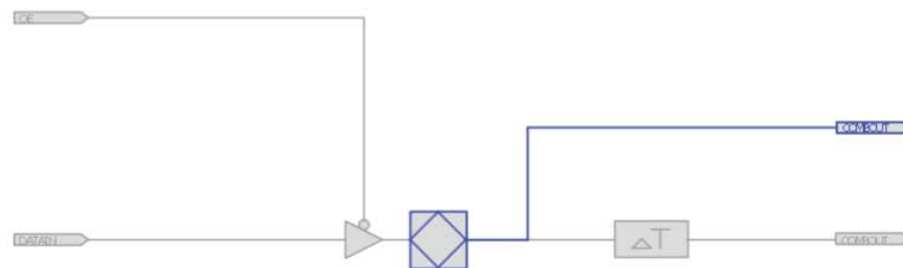
Figure 124. Cyclone V Device I/O Elements and Structure



17.5.3.5 MAX V I/O Elements

The I/O elements in MAX[®] V devices contain a bidirectional I/O buffer. You can drive registers from adjacent LABs to or from the bidirectional I/O buffer of the I/O element. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

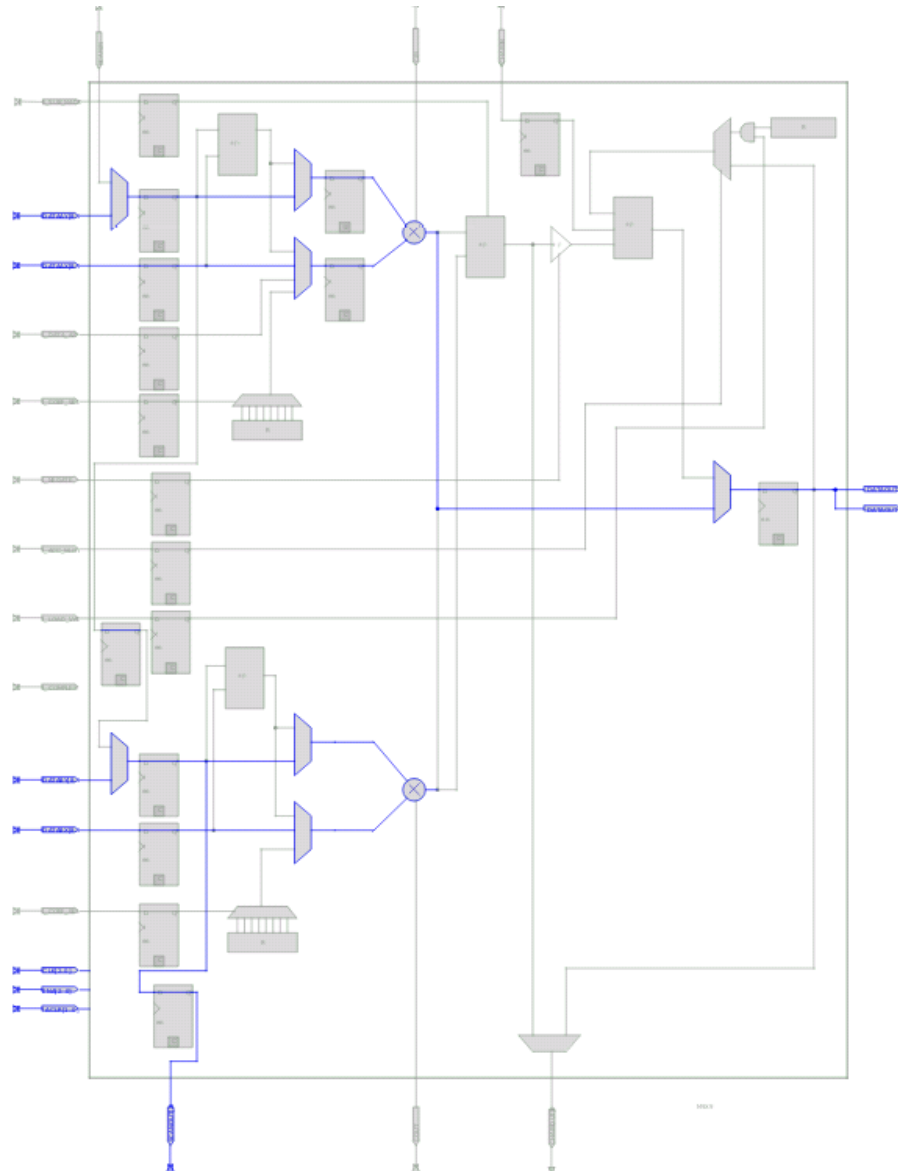
Figure 125. MAX V Device I/O Elements and Structure



17.5.4 FPGA RAM Blocks

With the Resource Property Editor, you can view the architecture of different RAM blocks in the device, modify the input and output registers to and from the RAM blocks, and modify the connectivity of the input and output ports. By default, the Intel Quartus Prime software displays the used resources in blue and the unused resources in gray.

Figure 126. M9K RAM View in a Stratix V Device



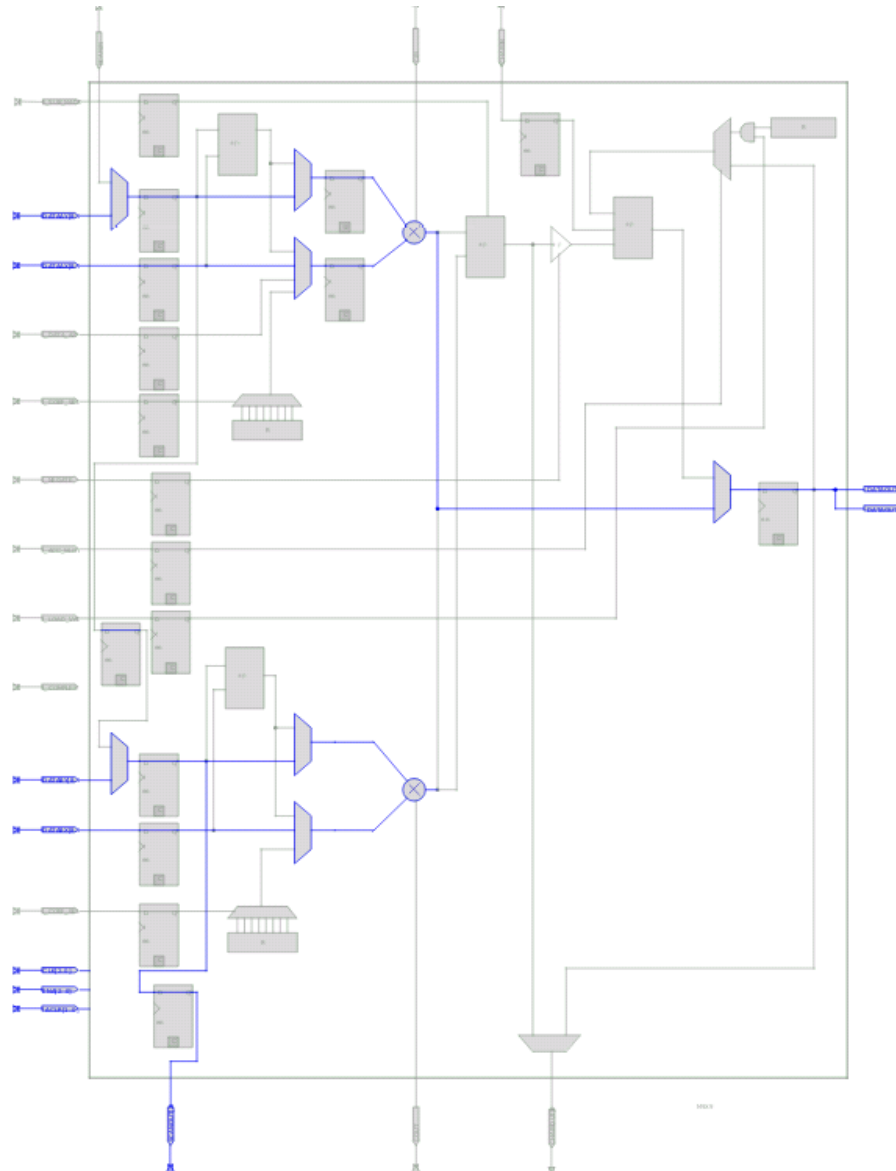
17.5.5 FPGA DSP Blocks

Dedicated hardware DSP circuit blocks in Altera devices provide performance benefits for the critical DSP functions in your design.

The Resource Property Editor allows you to view the architecture of DSP blocks in the Resource Property Editor for the Cyclone and Stratix series of devices. The Resource Property Editor also allows you to modify the signal connections to and from the DSP blocks and modify the input and output registers to and from the DSP blocks. By default, the [Intel Quartus Prime](#) software displays the used resources in blue and the unused resources in gray.



Figure 127. DSP Block View in a Stratix V Device



17.6 Change Manager

The Change Manager maintains a record of every change you perform with the Chip Planner, the Resource Property Editor, the Signal Probe feature, or a Tcl script. Each row of data in the Change Manager represents one ECO.

The Change Manager allows you to apply changes, roll back changes, delete changes, and export change records to a Text File (**.txt**), a Comma-Separated Value File (**.csv**), or a Tcl Script File (**.tcl**). The Change Manager tracks dependencies between changes, so that when you apply, roll back, or delete a change, any prerequisite or dependent changes are also applied, rolled back, or deleted.

17.6.1 Complex Changes in the Change Manager

Certain changes in the Change Manager (including creating or deleting atoms and changing connectivity) can appear to be self-contained, but are actually composed of multiple actions. The Change Manager marks such complex changes with a plus icon in the **Index** column.

You can click the plus icon to expand the change record and show all the component actions performed as part of that complex change.

Related Links

[Example of Managing Changes With the Change Manager](#)

17.6.2 Managing Signal Probe Signals

The Signal Probe pins that you create from the **Signal Probe Pins** dialog box are recorded in the Change Manager. After you have made a Signal Probe assignment, you can use the Change Manager to quickly disable Signal Probe assignments by selecting **Revert to Last Saved Netlist** on the shortcut menu in the Change Manager.

Related Links

[Quick Design Debugging Using Signal Probe](#)

17.6.3 Exporting Changes

You can export changes to a **.txt**, a **.csv**, or a **.tcl**. Tcl scripts allow you to reapply changes that were deleted during compilation.

Related Links

[Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design](#)

17.7 Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. The Tcl commands for controlling the Chip Planner are located in the `chip_planner` package of the `quartus_cdb` executable.

Related Links

- [About Intel Quartus Prime Scripting](#)
- [Tcl Scripting](#) on page 81
- [Intel Quartus Prime Settings File Manual](#)
- [Command Line Scripting](#) on page 67

17.8 Common ECO Applications

You can use an ECO to make a post-compilation change to your design.



To help build your system quickly, you can use Chip Planner functions to perform the following activities:

- Adjust the drive strength of an I/O with the Chip Planner
- Modify the PLL properties with the Resource Property Editor, see "[Modify the PLL Properties With the Chip Planner](#)"
- Modify the connectivity between new resource atoms with the Chip Planner and Resource Property Editor

Related Links

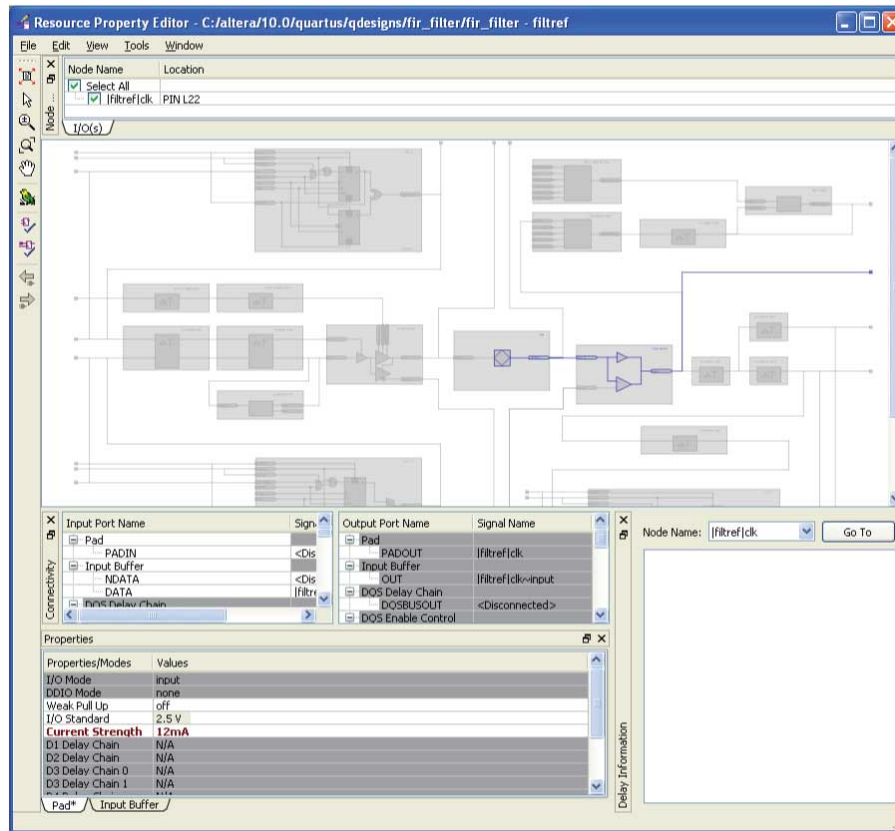
[Modify the PLL Properties With the Chip Planner](#) on page 336

17.8.1 Adjust the Drive Strength of an I/O with the Chip Planner

To adjust the drive strength of an I/O, follow these steps to incorporate the ECO changes into the netlist of the design.

1. In the **Editing Mode** list at the top of the Chip Planner, select the ECO editing mode.
2. Locate the I/O in the **Resource Property Editor**.
3. In the **Resource Property Editor**, point to the **Current Strength** option in the **Properties** pane and double-click the value to enable the drop-down list.
4. Change the value for the **Current Strength** option.
5. Right-click the ECO change in the Change Manager and click **Check & Save All Netlist Changes** to apply the ECO change.

Figure 128. I/O in the Resource Property Editor



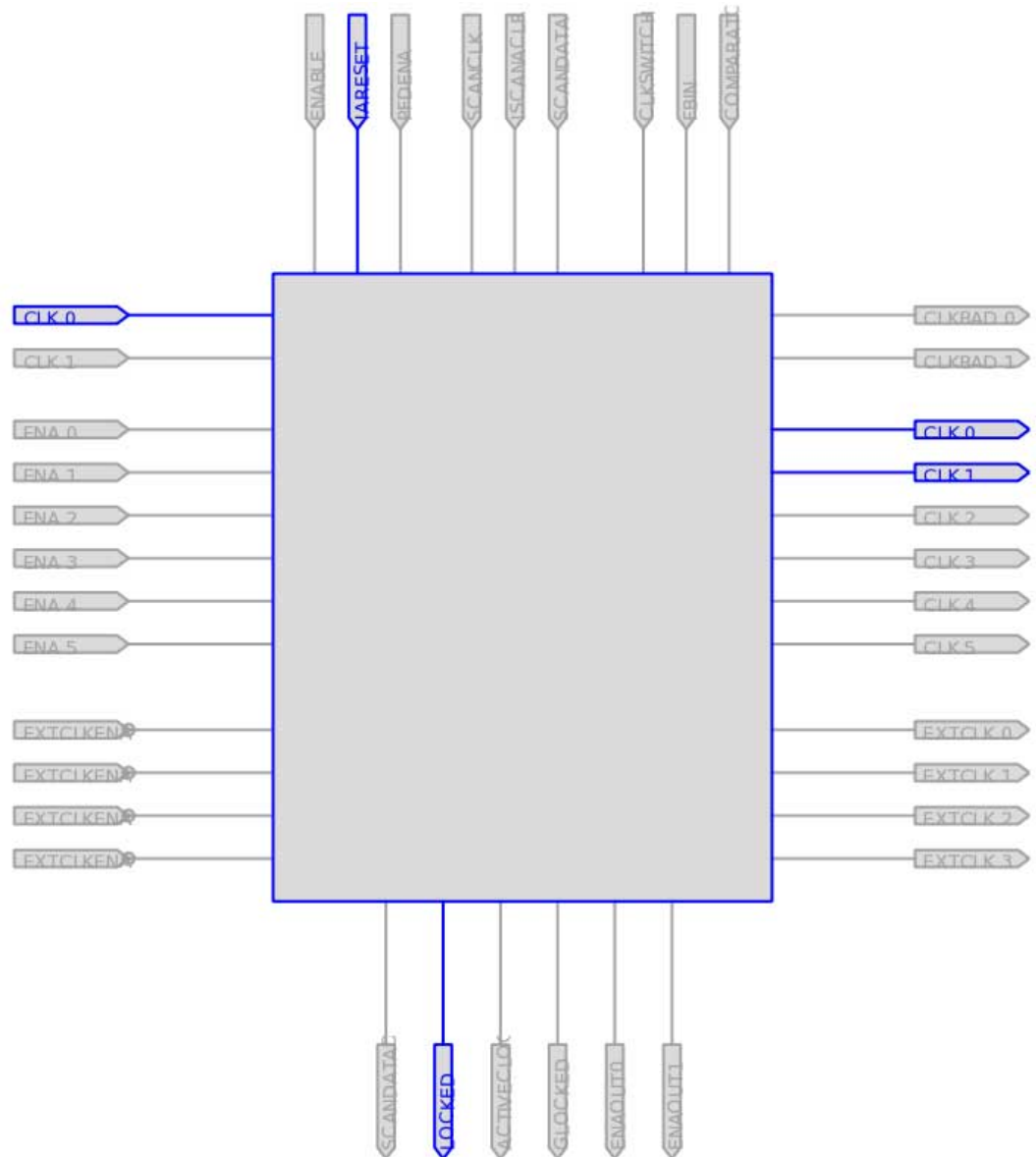
Note: You can change the pin locations of input or output ports with the ECO flow. You can drag and move the signal from an existing pin location to a new location while in the Post Compilation Editing (ECO) task in the Chip Planner. You can then click **Check & Save All Netlist Changes** to compile the ECO.

17.8.2 Modify the PLL Properties With the Chip Planner

You use PLLs to modify and generate clock signals to meet design requirements. Additionally, you can use PLLs to distribute clock signals to different devices in a design, reducing clock skew between devices, improving I/O timing, and generating internal clock signals.

The Resource Property Editor allows you to view and modify PLL properties to meet your design requirements.

Figure 129. PLL View in the Resource Property Editor of a Stratix Device



17.8.3 PLL Properties

The Resource Property Editor allows you to modify PLL options, such as phase shift, output clock frequency, and duty cycle.

You can also change the following PLL properties with the Resource Property Editor:

- Input frequency
- M V_{CO} tap
- M initial
- M value



- N value
- M counter delay
- N counter delay
- M2 value
- N2 value
- SS counter
- Charge pump current
- Loop filter resistance
- Loop filter capacitance
- Counter delay
- Counter high
- Counter low
- Counter mode
- Counter initial
- V_{CO} tap

You can also view post-compilation PLL properties in the Compilation Report. To do so, in the Compilation Report, select **Fitter** and then select **Resource Section**.

17.8.3.1 Adjusting the Duty Cycle

Use the equation to adjust the duty cycle of individual output clocks.

$$\text{High \%} = \frac{\text{Counter High}}{(\text{Counter High} + \text{Counter Low})}$$

17.8.3.2 Adjusting the Phase Shift

Use the equation to adjust the phase shift of an output clock of a PLL.

$$\text{Phase Shift} = (\text{Period } V_{CO} \times 0.125 \times \text{Tap } V_{CO}) + (\text{Initial } V_{CO} \times \text{Period } V_{CO})$$

For normal mode, Tap V_{CO}, Initial V_{CO}, and Period V_{CO} are governed by the following settings:

$$\text{Tap } V_{CO} = \text{Counter Delay} - M \text{ Tap } V_{CO}$$

$$\text{Initial } V_{CO} = \text{Counter Delay} - M \text{ Initial}$$

$$\text{Period } V_{CO} = \text{In Clock Period} \times N \div M$$

For external feedback mode, Tap V_{CO}, Initial V_{CO}, and Period V_{CO} are governed by the following settings:

$$\text{Tap } V_{CO} = \text{Counter Delay} - M \text{ Tap } V_{CO}$$

$$\text{Initial } V_{CO} = \text{Counter Delay} - M \text{ Initial}$$

$$\text{Period } V_{CO} = \underline{\text{In Clock Period} \times N}$$



(M+ Counter High+Counter Low)

Related Links

[Stratix Device Handbook](#)

17.8.3.3 Adjusting the Output Clock Frequency

Use the equation to adjust the PLL output clock in normal mode.

$$\text{Output Clock Frequency} = \text{Input Frequency} \cdot \frac{\text{M Value}}{\text{N Value} + \text{Counter High} + \text{Counter Low}}$$

Use the equation to adjust the PLL output clock in external feedback mode.

$$\text{OUTCLK} = \frac{\text{M Value} + \text{External Feedback Counter High} + \text{External Feedback Counter Low}}{\text{N Value} + \text{Counter High} + \text{Counter Low}}$$

17.8.3.4 Adjusting the Spread Spectrum

Use the equation to adjust the spread spectrum for your PLL.

$$\% \text{ Spread} = \frac{M_2 N_1}{M_1 N_2}$$

17.8.4 Modify the Connectivity between Resource Atoms

The Chip Planner and Resource Property Editor allow you to create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist. These features are useful for small changes when you are debugging a design, such as manually inserting pipeline registers into a combinational path that fails timing, or routing a signal to a spare I/O pin for analysis.

Use the following procedure to create a new register in a Cyclone V device and route register output to a spare I/O pin. This example illustrates how to create a new resource atom and modify the connections between resource atoms.

To create new resource atoms and manipulate the existing connection between resource atoms in the post-fit netlist, follow these steps:

1. Create a new register in the Chip Planner.
2. Locate the atom in the Resource Property Editor.
3. To assign a clock signal to the register, right-click the clock input port for the register, point to **Edit connection**, and click **Other**. Use the Node Finder to assign a clock signal from your design.
4. To tie the SLOAD input port to V_{CC} , right-click the clock input port for the register, point to **Edit connection**, and click **VCC**.
5. Assign a data signal from your design to the SDATA port.
6. In the Connectivity window, under the output port names, copy the port name of the register.
7. In the Chip Planner, locate a free I/O resource and create an output buffer.
8. Locate the new I/O atom in the Resource Property Editor.



9. On the input port to the output buffer, right-click, point to **Edit connection**, and click **Other**.
10. In the **Edit Connection** dialog box, type the output port name of the register you have created.
11. Run the ECO Fitter to apply the changes by clicking **Check and Save Netlist Changes**.

Note: A successful ECO connection is subject to the available routing resources. You can view the relative routing utilization by selecting **Routing Utilization** as the Background Color Map in the **Layers Settings** dialog box of the Chip Planner. Also, you can view individual routing channel utilization from local, row, and column interconnects with the tooltips created when you position your mouse pointer over the appropriate resource. Refer to the device data sheet for more information about the architecture of the routing interconnects of your device.

17.9 Post ECO Steps

After you make an ECO change with the Chip Planner, you must perform static timing analysis of your design with the Timing Analyzer to ensure that your changes did not adversely affect the timing performance of your design.

For example, when you turn on one of the delay chain settings for a specific pin, you change the I/O timing. Therefore, to ensure that the design still meets all timing requirements, you should perform static timing analysis.

Related Links

[Intel Quartus Prime Timing Analyzer](#)

For more information about performing a static timing analysis of your design

17.10 Document Revision History

Table 56. Document Revision History

Date	Version	Changes
2015.11.02	15.1.0	Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i> .
June 2014	14.0.0	<ul style="list-style-type: none"> • Updated formatting. • Removed references to Stratix, Stratix II, Stratix III, Arria GX, Arria II GX, Cyclone, Cyclone II, Cyclone III, and MAX II devices. • Added MAX V, Cyclone V, Arria V I/O elements
June 2012	12.0.0	Removed survey link.
November 2011	10.1.1	Template update.
December 2010	10.1.0	<ul style="list-style-type: none"> • Updated chapter to new template • Removed "The Chip Planner FloorPlan Views" section • Combined "Creating Atoms", "Deleting Atoms", and "Moving Atoms" sections, and linked to Help. • Added Stratix V I/O elements in "FPGA I/O Elements".
July 2010	10.0.0	<ul style="list-style-type: none"> • Added information to page 17-1. • Added information to "Engineering Change Orders" on page 17-2. • Changed heading from "Performance" to "Performance Preservation" on page 7-2. • Updated information in "Performance Preservation" on page 17-2.

continued...



Date	Version	Changes
		<ul style="list-style-type: none"> • Changed heading from "Documentation" to "Change Modification Record" on page 17-3. • Changed heading from "Resource Property Editor" to "Performing ECOs in the Resource Property Editor" on page 17-15. • Removed "Using Incremental Compilation in the ECO Flow" section. Preservation support for ECOs with the incremental compilation flow has been removed in the Intel Quartus Prime software version 10.0. • Removed "Referenced Documents" section.
November 2009	9.1.0	<ul style="list-style-type: none"> • Updated device support list • Made minor editorial updates
March 2009	9.0.0	<ul style="list-style-type: none"> • Updated Figure 17-17. • Made minor editorial updates. • Chapter 15 was previously Chapter 13 in the 8.1.0 release.
November 2008	8.1.0	<ul style="list-style-type: none"> • Corrected preservation attributes for ECOs in the section "Using Incremental Compilation in the ECO Flow" on page15-32. • Minor editorial updates. • Changed to 8½" x 11" page size.
May 2008	8.0.0	<ul style="list-style-type: none"> • Updated device support list • Modified description for ECO support for block RAMs and DSP blocks • Corrected Stratix PLL ECO example • Added an application example to show modifying the connectivity between resource atoms

Related Links

[Documentation Archive](#)

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