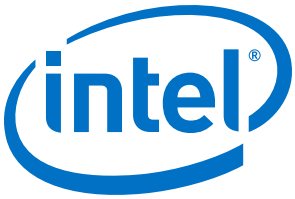


Intel® Quartus® Prime Standard Edition User Guide

Getting Started

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

Subscribe
Send Feedback

UG-20173 | 2018.09.24
Latest document on the web: PDF | HTML

Contents

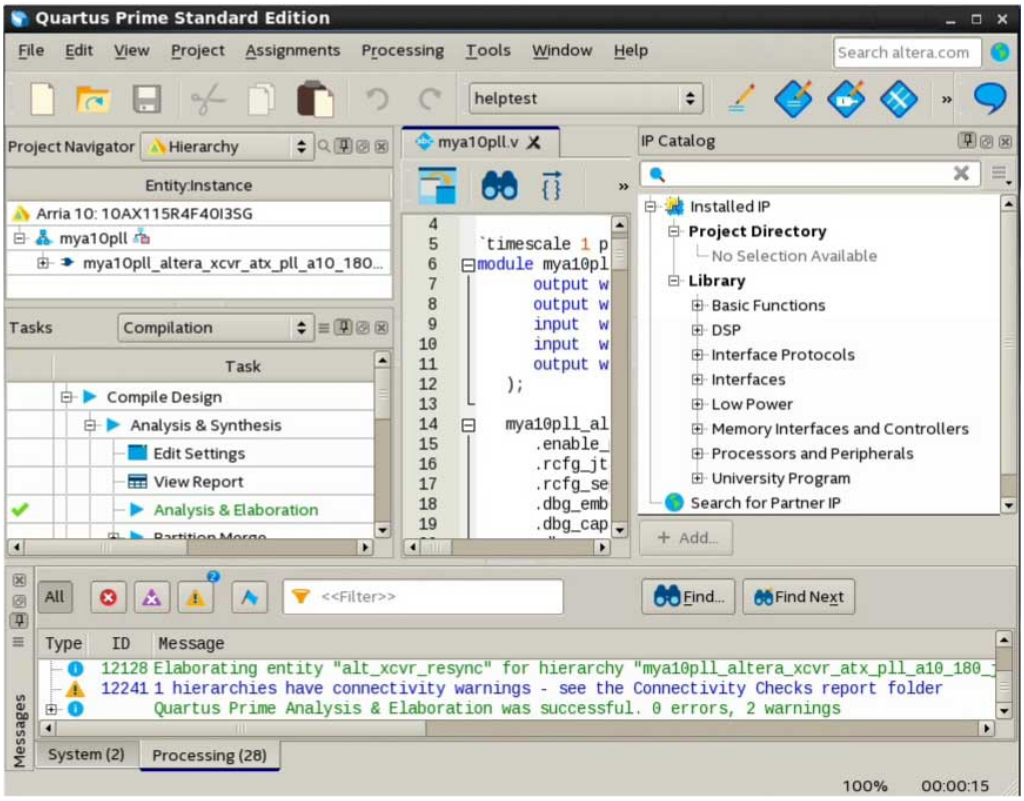
1. Introduction to Intel® Quartus® Prime Standard Edition

This user guide describes basic concepts and operation of the Intel® Quartus® Prime Standard Edition design software, including GUI and project structure basics, initial design planning, use of Intel FPGA IP, and migration to Intel Quartus Prime Pro Edition. The Intel Quartus Prime Standard Edition software provides a complete design environment for the following device families:

- Intel Arria® 10, Arria V, and Arria II
- Intel Cyclone® 10 LP, Cyclone IV, and Cyclone V
- MAX® series

The Intel Quartus Prime software GUI supports easy design entry, fast design processing, straightforward device programming, and integration with other industry-standard EDA tools. The user interface makes it easy for you to focus on your design—not on the design tool. The modular Compiler streamlines the FPGA development process, and ensures the highest performance for the least effort.

Figure 1. Intel Quartus Prime Standard Edition Software GUI



The Intel Quartus Prime Standard Edition software offers a full range of features at each phase of the design flow to shorten your design cycle and achieve the highest performance:

- Easy Project Setup—quickly create a new project, add design files, and specify the target Intel device with the New Project Wizard. Create different revisions of your project to compare results with different settings. Save the current state of your project and project files as a single, compressed file. Refer to Managing Intel Quartus Prime Projects on page 9 for more information.
- Design Planning Tools— plan for initial I/O pin layout, power consumption, and area utilization in the Early Power Estimator, the Power Analyzer Tool, and the Pin Planner. Refer to Design Planning on page 36 for more information.
- Design Constraint Entry—specify timing, placement, and other constraints with the **Settings** dialog box, Assignment Editor, Pin Planner, and Timing Analyzer. Visualize and modify logic placement within a view of the device floorplan in the Chip Planner and Timing Closure Floorplan. Refer to Intel Quartus Prime Standard Edition User Guide: Design Constraints for more information.
- Integrated Synthesis—provides efficient synthesis support for VHDL (1987, 1993, 2008), Verilog HDL (1995, 2001), and SystemVerilog (2005) design entry languages. Refer to Intel Quartus Prime Standard Edition User Guide: Compiler for more information.
- Incremental Compilation—preserve the results and performance for unchanged logic in your design as you make changes elsewhere, facilitating top-down or bottom-up team-based design methodologies. Refer to Intel Quartus Prime Standard Edition User Guide: Compiler for more information.
- Optimizing Results—Design Space Explorer automatically determines the best combination of settings for your design. Design Assistant validates your project against predetermined design rules for gated clocks, reset signals, asynchronous design practices, and signal race conditions. Refer to Intel Quartus Prime Standard Edition User Guide: Design Optimization for more information.
- Design Debugging—The Signal Tap Logic Analyzer captures and displays real-time signal behavior in an FPGA design, allowing to examine the behavior of internal signals during normal device operation without the need for extra I/O pins or external lab equipment. The Transceiver Toolkit provides real-time control, monitoring, and debugging of the transceiver links running on your board. Refer to Intel Quartus Prime Standard Edition User Guide: Debug Tools for more information.
- System and IP Integration—define and generate a complete system in much less time than using traditional, manual integration methods with Platform Designer (Standard). Refer to Introduction to Intel FPGA IP Cores on page 52 and Intel Quartus Prime Standard Edition User Guide: Platform Designer for more information.
- Third-party EDA Tool Support—integrate with supported versions of third-party EDA synthesis, simulation, and board-level timing analysis tools. Refer to Third-Party Simulation and Third-Party Synthesis user guides for more information.

The Intel Quartus Prime Pro Edition software expands on these capabilities of the Intel Quartus Prime Standard Edition, and provides unique features that support the latest Intel FPGAs. Select the Intel Quartus Prime software edition that provides the device support and features you require, as Selecting an Intel Quartus Prime Software Edition on page 7 describes.

1.1. Selecting an Intel Quartus Prime Software Edition

Depending on your device support and software feature requirements, you can choose either the Intel Quartus Prime Pro Edition or Intel Quartus Prime Standard Edition software for your design. Consider the requirements and timeline of your project in determining whether to select the Intel Quartus Prime Standard Edition or Intel Quartus Prime Pro Edition software:

- Select the Intel Quartus Prime Pro Edition if you are beginning a new Intel Arria 10, Intel Cyclone 10 GX, or Intel Stratix® 10 design, or to take advantage of the unique features of Intel Quartus Prime Pro Edition.

Figure 2. Intel Quartus Prime Feature Support Matrix

Software Features	Intel Quartus® Prime Standard Edition	Intel Quartus Prime Pro Edition
New Hybrid Placer & Global Router	✔	✔
New Timing Analyzer	✔	✔
New Physical Synthesis	✔	✔
Platform Designer (formerly Qsys)	✔	✔
Intel Stratix® 10, Intel Cyclone® 10 GX		✔
New Synthesis Engine		✔
Partial Reconfiguration		✔
Block-Based (Hierarchical) Design Flows		✔
OpenCL support		✔
Incremental Fitter Optimization		✔
Interface Planner (formerly BluePrint)		✔

- Select the Intel Quartus Prime Standard Edition software if your design must target Arria V, Arria, Intel Cyclone 10 LP, Cyclone IV, Cyclone V, or MAX series devices, and you do not want to migrate you design to a device that Intel Quartus Prime Pro Edition supports.
- Intel Quartus Prime Pro Edition software does not support the following Intel Quartus Prime Standard Edition features:
  - I/O Timing Analysis
  - NativeLink third party tool integration (other third-party tool integration available)
  - Video and Image Processing Suite IP Cores
  - Talkback features
  - Various register merging and duplication settings
  - Saving a node-level netlist as `.vqm`
  - Compare project revisions

Note: Intel replaces the following Altera tool names in the Intel Quartus Prime software:

Table 1. Intel Quartus Prime Tool Name Updates

Altera Name	Intel Name
Qsys	Platform Designer
TimeQuest	Timing Analyzer
EyeQ	Eye Viewer
JNEye	Advanced Link Analyzer

Related Information

Migrating to Intel Quartus Prime Pro Edition on page 73

1.2. Introduction to Intel Quartus Prime Standard Edition Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release for Intel Quartus Prime Standard Edition.

Related Information

Documentation Archive

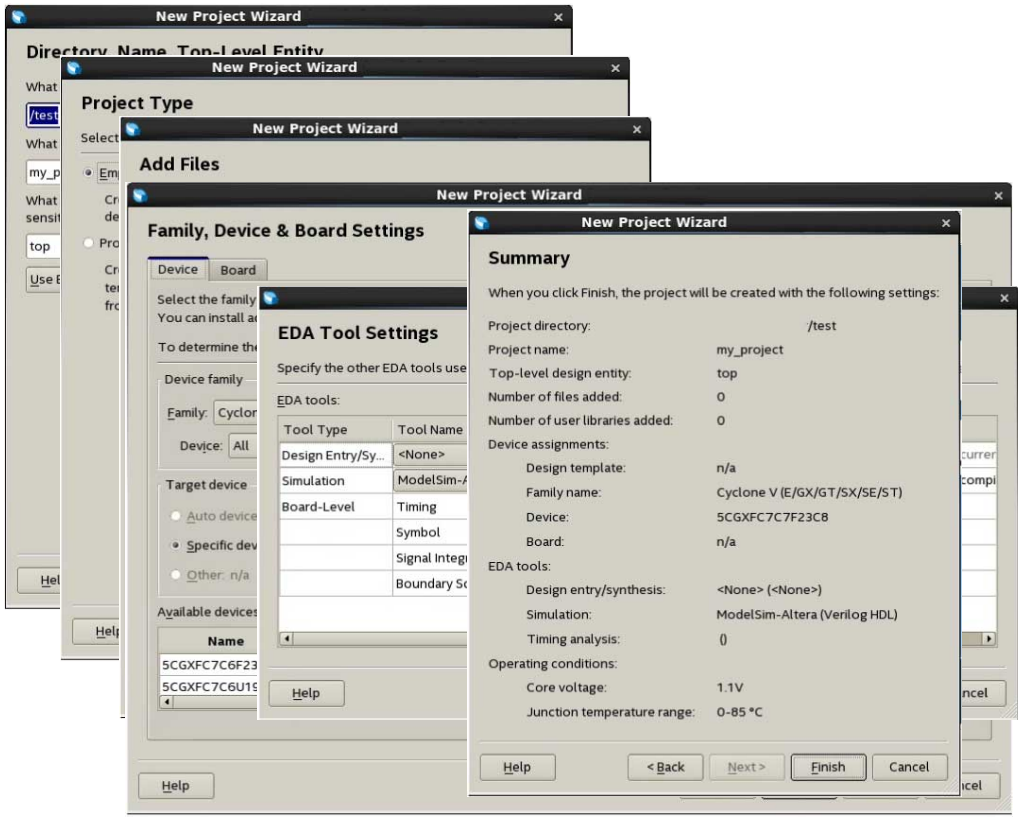
For previous versions of the *Intel Quartus Prime Handbook*, search the documentation archives.

2. Managing Intel Quartus Prime Projects

The Intel Quartus Prime software organizes and manages the elements of your design within a *project*. The project encapsulates information about your design files, hierarchy, libraries, constraints, and project settings. This chapter describes the basics of working with Intel Quartus Prime software projects, including initial project setup, viewing project information, adding design files and constraints, and exporting compilation results.

Click **File > New Project Wizard** to quickly setup and open a new project.

Figure 3. New Project Wizard



After you create or open a project, the GUI displays integrated information and controls for the open project.

2.1. Viewing Basic Project Information

You can view basic information about your project in the **Tasks** pane, Project Navigator, Report panel, and **Messages** window.

Project Tasks Pane

The **Tasks** pane (**View ➤ Tasks**) provides one-click launch of common project tasks, such as creating design files, specifying project settings, running compilation, debug and timing closure, and device programming.

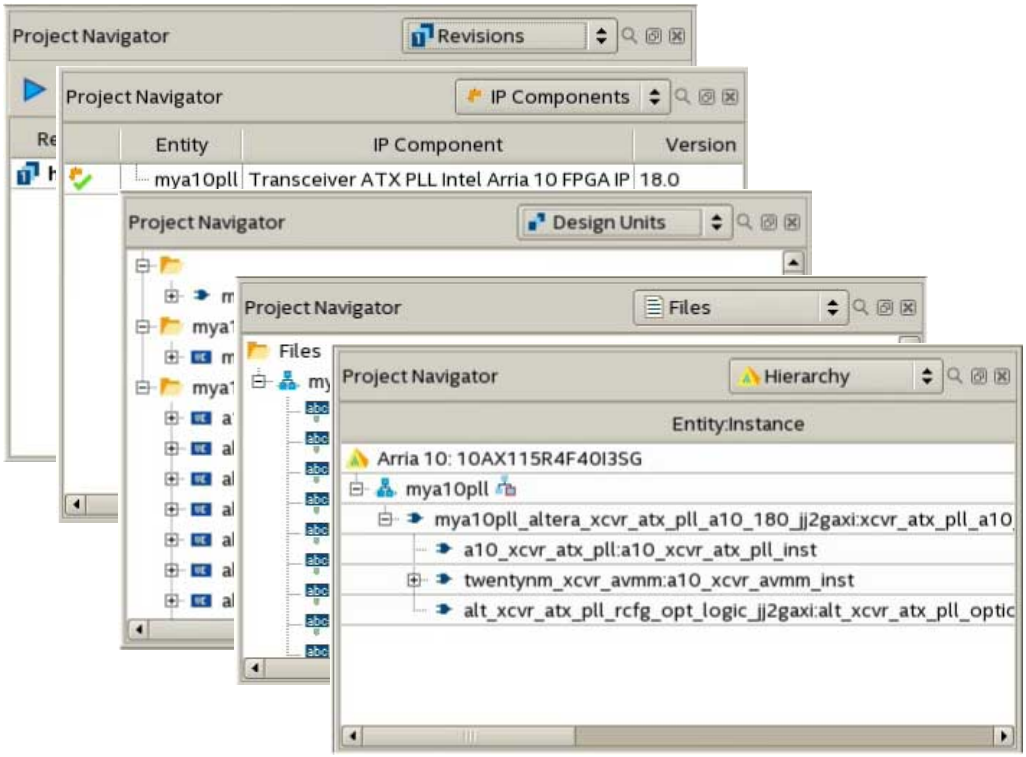
The Project Navigator

The **Project Navigator** (**View ➤ Utility Windows ➤ Project Navigator**) displays information about the elements of your project, such as the design files, IP components, and your project hierarchy (after elaboration). You can right-click items in the **Project Navigator** to locate or perform actions on the elements of your project. The Project Navigator organizes information on the **Files**, **Hierarchy**, **Design Units**, and **IP Components** tabs.

Table 2. Project Navigator Tabs

Project Navigator Tab	Description
Files	Lists all design files in the current project. Right-click design files in this tab to run these commands: • **Open** the file • **Remove** the file from project • View file **Properties** • **Create AHDL Include Files for Current File** • **Create Symbol Files for Current File** • **Create Verilog Instantiation Template Files for Current File** • **Create VHDL Component Declaration Files for Current File**
Hierarchy	Provides a visual representation of the project hierarchy, specific resource usage information, and device and device family information. Right-click items in the hierarchy to **Locate**, **Set as Top-Level Entity**, or define Logic Lock regions or design partitions.
Design Units	Displays the design units in the project. Right-click a design unit to **Locate in Design File**.
IP Components	Displays the design files that make up the IP instantiated in the project, including Intel FPGA IP, Platform Designer (Standard) components, and third-party IP. Click **Launch IP Upgrade Tool** from this tab to upgrade outdated IP components. Right-click any IP component to **Edit in Parameter Editor**.
Revisions	Displays the current project revisions.

Figure 4. Project Navigator Hierarchy, Files, Design Units, IP Components, and Revisions Tabs



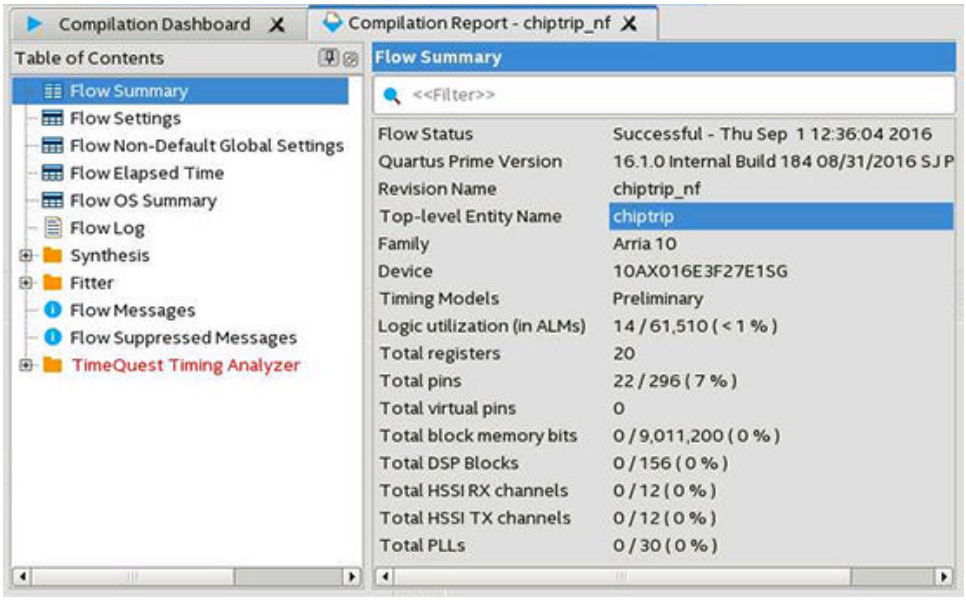
2.1.1. Viewing Project Reports

The Compilation Report panel updates dynamically to display detailed reports during project processing.

To access Compilation Reports, click (**Processing ➤ Compilation Report**):

Analyze the detailed information in these reports to determine correct implementation. Right-click report data to locate and edit the source in project files.

Figure 5. Compilation Report



Related Information

List of Compilation Reports

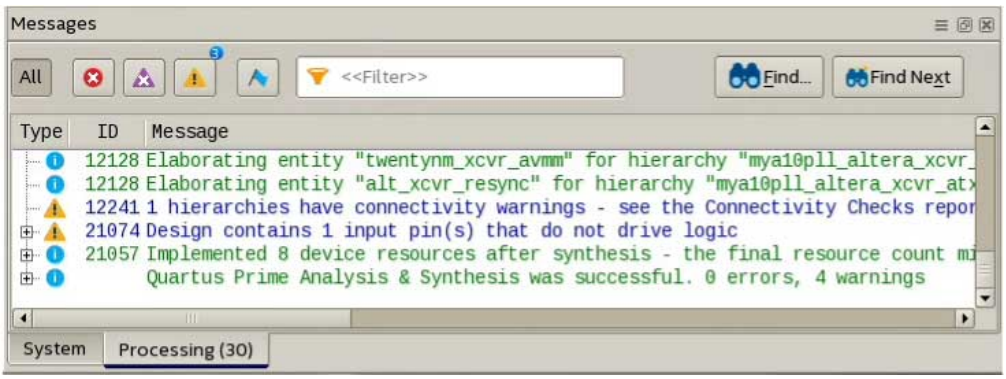
2.1.2. Viewing Project Messages

The Messages window (**View ➤ Utility Windows ➤ Messages**) displays information, warning, and error messages about Intel Quartus Prime processes. Right-click messages to locate the source or get message help.

- **Processing** tab—displays messages from the most recent process
- **System** tab—displays messages unrelated to design processing
- **Search**—locates specific messages

Messages are written to `stdout` when you use command-line executables.

Figure 6. Messages Window



To suppress display of unimportant messages, right-click one or more messages and click on of the following:

- **Suppress Message**—suppresses all messages matching exact text
- **Suppress Messages with Matching ID**—suppresses all messages matching the message ID number, ignoring variables
- **Suppress Messages with Matching Keyword**—suppresses all messages matching keyword or hierarchy

Figure 7. Message Suppression by Message ID Number

- You cannot suppress error or Intel legal agreement messages.
- Suppressing a message also suppresses any submessages.
- Message suppression is project revision-specific. Derivative project revisions inherit any suppression.
- You cannot edit messages or suppression rules during compilation.

2.2. Intel Quartus Prime Project Contents

The Intel Quartus Prime software organizes your design work within a project. You can create and compare multiple revisions of your project, to experiment with settings that achieve your design goals. When you create a new project in the GUI, the Intel Quartus Prime software automatically creates an Intel Quartus Prime Project File (`.qpf`) for that project. The `.qpf` references the Intel Quartus Prime Settings File (`.qsf`). The `.qsf` lists the project's design, constraint, and IP files, and stores project-wide and entity-specific settings that you specify in the GUI. You do not need to edit the text-based `.qpf` or `.qsf` files directly. The Intel Quartus Prime software creates and updates these files automatically as you make changes in the GUI.

Table 3. Intel Quartus Prime Project Files

File Type	Contains	To Edit	Format
Project file	Project and revision name	**File ➤ New Project Wizard**	Intel Quartus Prime Project File (`.qpf`)
Settings file	Lists design files, entity settings, target device, synthesis directives, placement constraints	**Assignments ➤ Settings**	Intel Quartus Prime Settings File (`.qsf`)
Quartus database	Project compilation results	**Project ➤ Export Database**	`export_db` directory
Partition database	Partition compilation results	**Project ➤ Export Design Partition**	Exported Partition File (`.qxp`)
Timing constraints	Clock properties, exceptions, setup/hold	**Tools ➤ Timing Analyzer**	Synopsys Design Constraints File (`.sdc`)
Logic design files	RTL and other design source files	**File ➤ New**	All supported HDL files
Programming files	Device programming image and information	**Tools ➤ Programmer**	SRAM Object File (`.sof`) Programmer Object File (`.pof`)
IP core files	IP core variation parameterization	**Tools ➤ IP Catalog**	Intel Quartus Prime IP File (`.qip`)
Platform Designer system files	System definition	**Tools ➤ Platform Designer**	Platform Designer System File (`.qsys`)
EDA tool files	Scripts for third-party EDA tools	**Assignments ➤ Settings ➤ EDA Tool Settings**	Verilog Output File (`.vo`) VHDL Output File (`.vho`) Verilog Quartus Mapping File (`.vqm`)
Archive files	Complete project as single compressed file	**Project ➤ Archive Project**	Intel Quartus Prime Archive File (`.qar`)

2.2.1. Project File Best Practices

The Intel Quartus Prime software provides various options for specifying project settings and constraints. The following best practices help ensure automated management and portability of your project files.

- Avoid manually editing Intel Quartus Prime data files, such as the Intel Quartus Prime Project File (`.qpf`), Intel Quartus Prime Settings File (`.qsf`), Quartus IP File (`.ip`), or Platform Designer (Standard) System File (`.qsys`). Syntax errors in these files cause errors during compilation. For example, the software may ignore improperly formatted settings and assignments.
- Do not compile multiple projects into the same directory. Instead, use a separate directory for each project.
- By default, the Intel Quartus Prime software saves all project output files, such as Text-Format Report Files (`.rpt`), in the project directory. If you want to change the location of output files, instead of manually moving project output files, click **Assignments ➤ Settings ➤ Compilation Process Settings**, and specify the **Save project output files in specified directory** option.

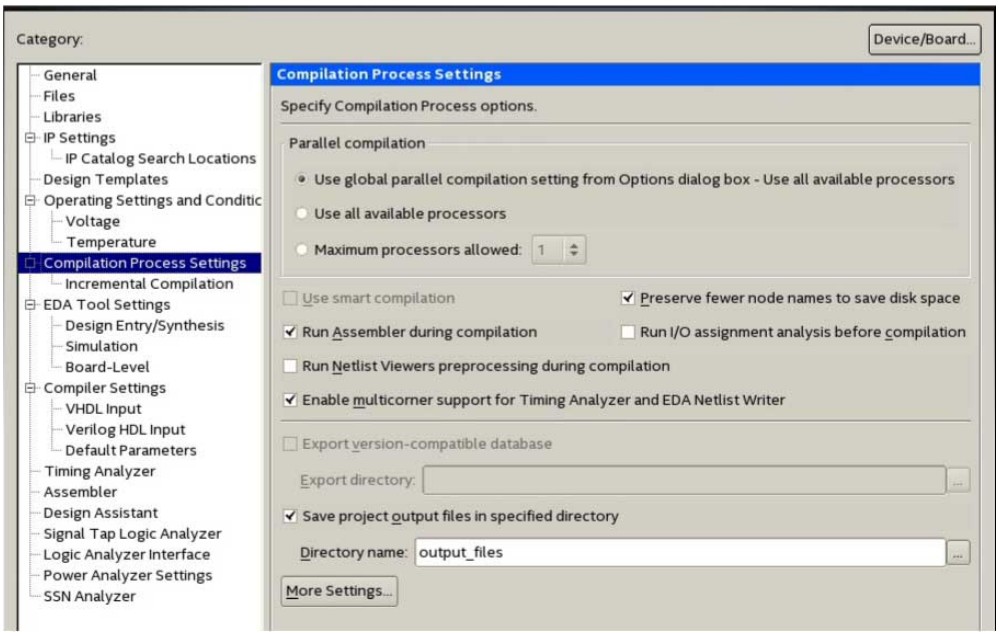
2.3. Managing Project Settings

The New Project Wizard guides you to make initial project settings when you setup a new project. You can modify these and other global project settings in the **Settings** and **Device** dialog boxes, respectively. The `.qsf` stores the settings for each project revision. The optimization of these project settings helps the Compiler to generate programming files that meet or exceed your specifications.

Global Project Settings

To access global project settings, click **Assignments ➤ Settings**, or click **Settings** on the **Tasks** pane.

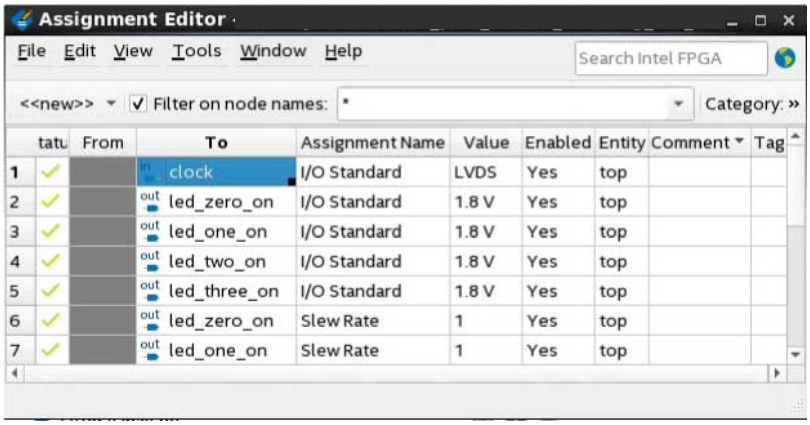
Figure 8. Settings Dialog Box for Global Project Settings



The **Settings** dialog box provides access to settings that control project design files, synthesis, Fitter, and timing constraints, operating conditions, EDA tool file generation, programming file generation, and other project-level settings.

Additionally, the Assignment Editor (**Assignments ➤ Assignment Editor**) provides a spreadsheet-like interface for specifying instance-specific settings and constraints.

Figure 9. Assignment Editor

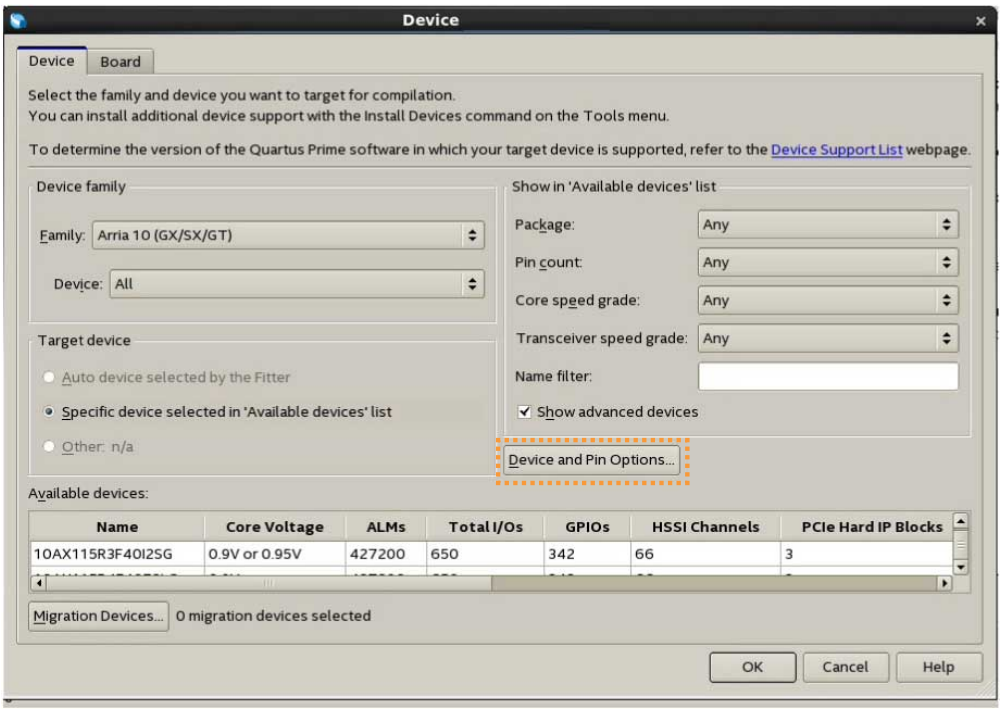


2.3.1. Specifying the Target Device or Board

Specify the target Intel device or board for your project in the **Device** dialog box. Click the **Device and Pin Options** button in the dialog to specify preferences for the device configuration scheme, programming file generation, I/O timing, voltage, and other options.

1. Open a project in the Intel Quartus Prime software.
2. Click **Assignments ➤ Device**.

Figure 10. Device Dialog Box



3. Specify either a target Intel FPGA board or device for your project. When you specify a board, the Intel Quartus Prime software generates the appropriate pin assignments script for that board automatically.
   - To specify an Intel FPGA board or development kit for your project:
     a. Click the **Board** tab.
     b. Select the target device **Family** and a supported **Development Kit**. Click **Yes** if prompted to remove existing **Location** and **I/O Standard** pin assignments. The Intel Quartus Prime software creates the kit's baseline design and stores the design in `<current_project_dir>/devkits/<design_name>`. To retain all your existing pin assignments, click **No**.
     c. Select the desired development kit and click **OK**.
   - To specify a device family for your project:
     a. On the **Device** tab, select the **Family** and **Device** name. The list of **Available devices** reflects your selections.
     b. To further refine your selection, specify options for the **Package**, **Pin count**, **Core speed grade**, **Name filter**, and **Show advanced devices** filters.
     c. From the **Available devices**, select your target device **Name** and click **OK**.

2.3.2. Optimizing Project Settings

Optimize project settings to meet your design goals. The Intel Quartus Prime Design Space Explorer II iteratively compiles your project with various setting combinations to find the optimal settings for your goals. Alternatively, you can create a project revision or project copy to manually compare various project settings and design combinations.

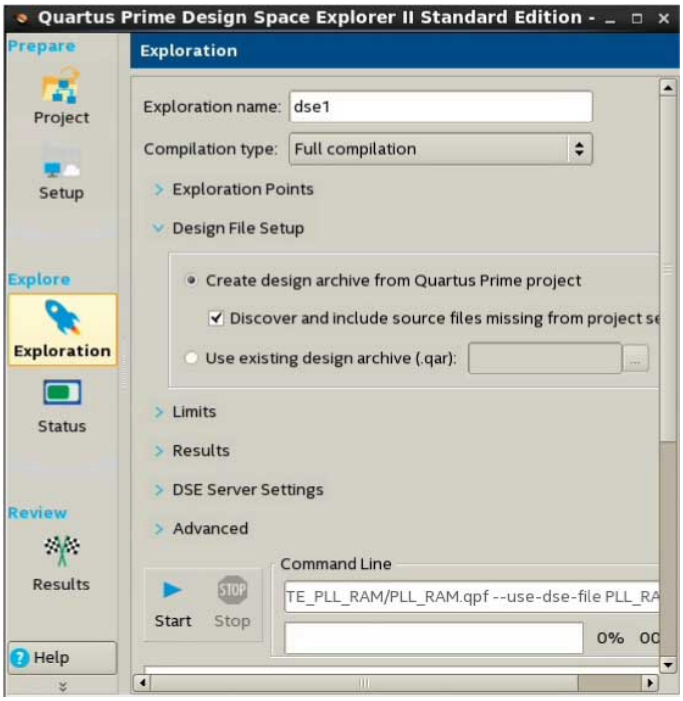
The Intel Quartus Prime software includes several advisors to help you optimize your design and reduce compilation time. The advisors listed in the **Tools ➤ Advisors** menu can provide recommendations based on your project settings and design constraints.

2.3.2.1. Optimize Settings with Design Space Explorer II

Use Design Space Explorer II (**Tools > Launch Design Space Explorer II**) to find optimal project settings for resource, performance, or power optimization goals. Design Space Explorer II (DSE II) processes your design using various setting and constraint combinations, and reports the best settings for your design.

DSE II attempts multiple seeds to identify one meeting your requirements. DSE II can run different compilations on multiple computers in parallel to streamline timing closure.

Figure 11. Design Space Explorer II



2.3.2.2. Optimize Settings with Project Revisions

You can save multiple, named project revisions within your Intel Quartus Prime project (**Project > Revisions**). Each project revision captures a unique set of project settings and constraints, while using the same set of logic design files.

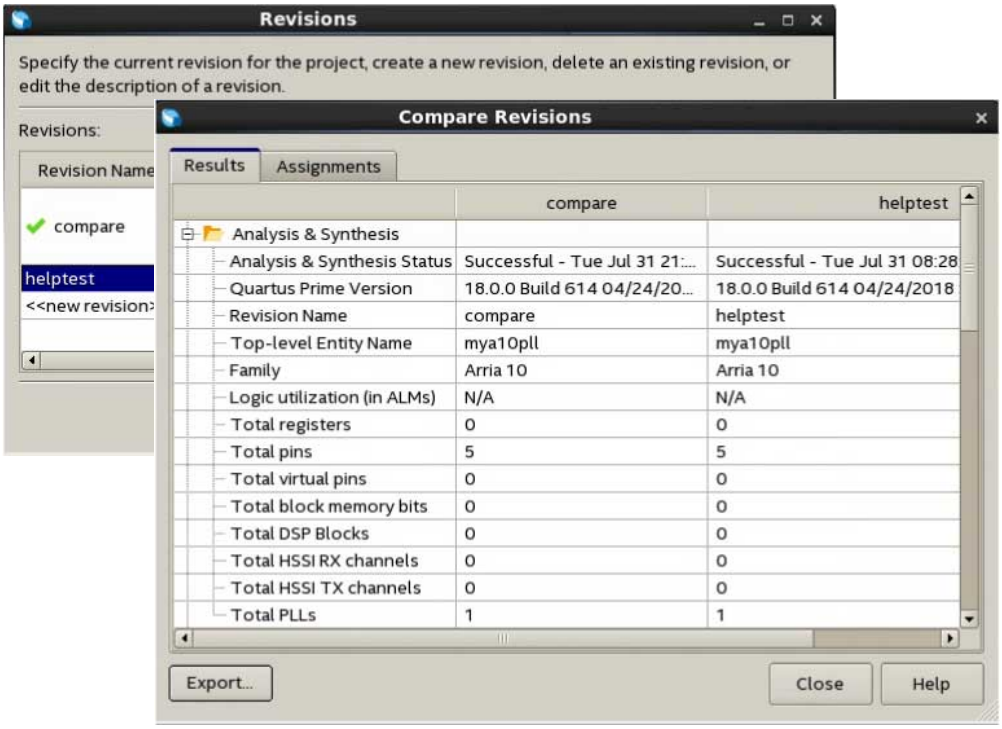
Use revisions to experiment with different settings while preserving the original. Optimize different revisions for separate applications:

- Create a unique revision to optimize a design for different criteria, such as by area in one revision and by f_{MAX} in another revision.
- When you create a new revision the default Intel Quartus Prime settings initially apply.
- Create a revision of a revision to experiment with settings and constraints. The child revision includes all the assignments and settings of the parent revision.

You create, delete, and edit revisions in the **Revisions** dialog box. Each time you create a new project revision, the Intel Quartus Prime software creates a new `.qsf` using the revision name.

To compare each revision's synthesis, fitting, and timing analysis results side-by-side, click **Project > Revisions** and then click **Compare**. In addition to viewing the compilation **Results** of each revision, you can also compare the **Assignments** for each revision. This comparison reveals how different optimization options affect your design.

Figure 12. Comparing Project Revisions



Related Information

Project Revision Commands on page 31

2.3.2.3. Back-Annotating Compiler Assignments

The Compiler maps the elements of your design to specific device resources during fitting. After compilation, you can back-annotate (copy) the Compiler's device and resource assignments to the project `.qsf` if you want to preserve that same implementation in subsequent compilations.

Click **Assignments ➤ Back-Annotate Assignments** to copy the device resource assignments from the last compilation to the `.qsf` for use in the next compilation. Select the back-annotation type in the **Back-annotation type** list.

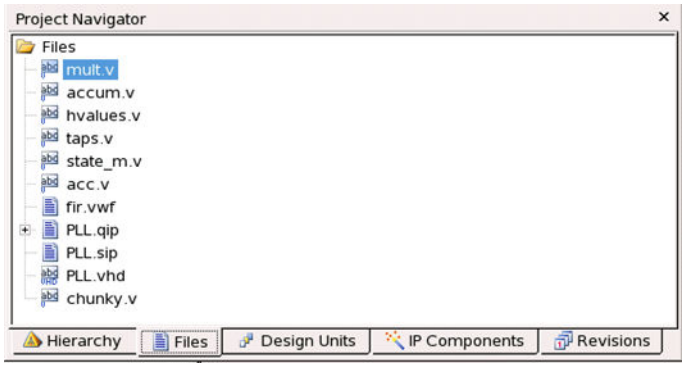
2.4. Managing Logic Design Files

The Intel Quartus Prime software helps you create and manage the logic design files in your project. Logic design files contain the logic that implements your design. When you add a logic design file to the project, the Compiler automatically includes that file in the next compilation. The Compiler synthesizes your logic design files to generate programming files for your target device.

The Intel Quartus Prime software includes full-featured schematic and text editors, as well as HDL templates to accelerate your design work. The Intel Quartus Prime software supports VHDL Design Files (`.vhd`), Verilog HDL Design Files (`.v`), SystemVerilog (`.sv`) and schematic Block Design Files (`.bdf`). The Intel Quartus Prime software also supports Verilog Quartus Mapping (`.vqm`) design files generated by other design entry and synthesis tools. In addition, you can combine your logic design files with Intel and third-party IP core design files, including combining components into a Platform Designer (Standard) system (`.qsys`).

The New Project Wizard prompts you to identify logic design files. Add or remove project files by clicking **Project > Add/Remove Files in Project.** View the project's logic design files in the Project Navigator.

Figure 13. Design and IP Files in Project Navigator



Right-click files in the Project Navigator to:

- **Open** and edit the file
- **Remove File from Project**
- **Set as Top-Level Entity** for the project revision
- **Create a Symbol File for Current File** for display in schematic editors
- Edit file **Properties**

2.4.1. Including Design Libraries

Include design files libraries in your project. Specify libraries for a single project, or for all Intel Quartus Prime projects. The `.qsf` stores project library information.

The `quartus2.ini` file stores global library information.

1. Click **Assignment > Settings**.
2. Click **Libraries** and specify the **Project Library name** or **Global Library name**. Alternatively, you can specify project libraries with SEARCH_PATH in the `.qsf`, and global libraries in the `quartus2.ini` file.

Related Information

Design Library Migration Guidelines on page 27

2.4.2. Creating a Project Copy

Click **Project > Copy Project** to create a separate copy of your project, rather than just a revision within the same project.

The project copy includes separate copies of all design files, any `.qsf` files, and project revisions. You can use this technique to optimize project copies for different applications that require design file differences. For example, you can optimize one project to interface with a 32-bit data bus, and optimize a project copy to interface with a 64-bit data bus.

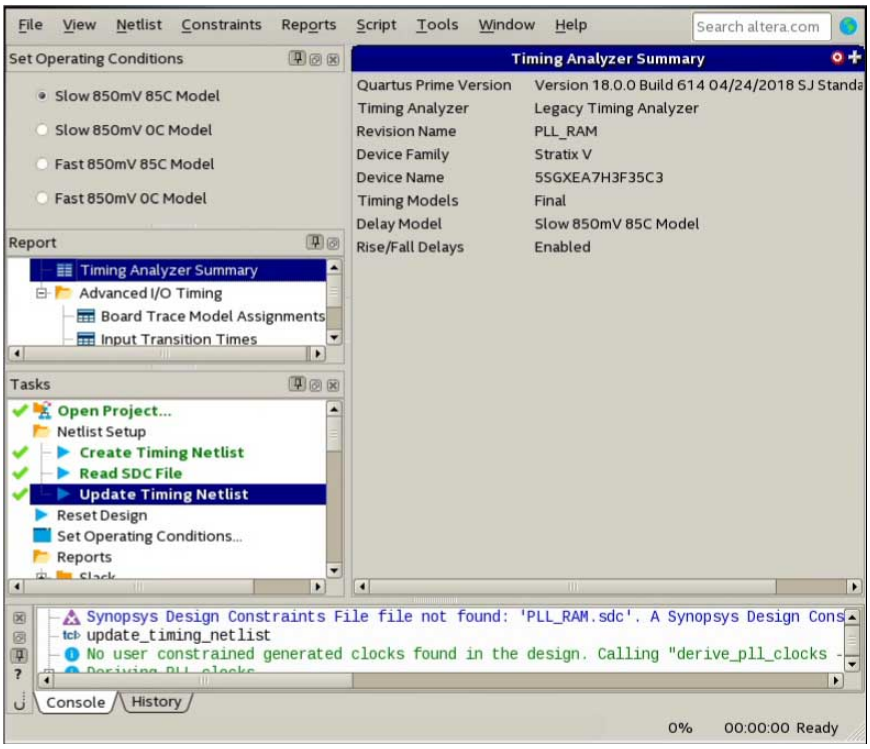
2.5. Managing Timing Constraints

Apply appropriate timing constraints to correctly optimize fitting and analyze timing for your design. The Fitter optimizes the placement of logic in the device to meet your specified timing and routing constraints.

Specify timing constraints in the Timing Analyzer (**Tools > Timing Analyzer**), or in an `.sdc` file. Specify constraints for clock characteristics, timing exceptions, and external signal setup and hold times before running analysis. The Timing Analyzer reports detailed information about the performance of your design compared with constraints in the Compilation Report panel.

Save the constraints you specify in the GUI in an industry-standard Synopsys Design Constraints File (`.sdc`). You can subsequently edit the text-based `.sdc` file directly. If you refer to multiple `.sdc` files in a parent `.sdc` file, the Timing Analyzer reads the `.sdc` files in the order you list.

Figure 14. Timing Analyzer



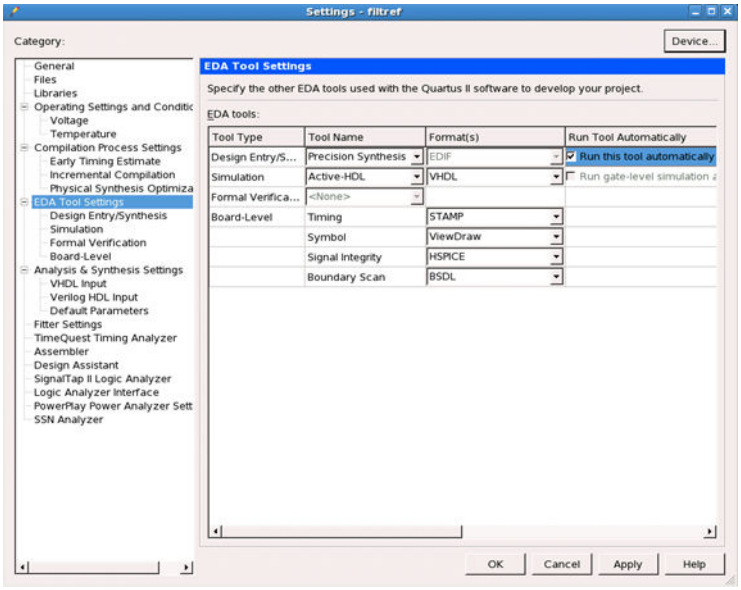
2.6. Integrating Other EDA Tools

Optionally integrate supported EDA design entry, synthesis, simulation, physical synthesis, and formal verification tools into the Intel Quartus Prime design flow. The Intel Quartus Prime software supports netlist files from other EDA design entry and synthesis tools. The Intel Quartus Prime software optionally generates various files for use in other EDA tools.

The Intel Quartus Prime software manages EDA tool files and provides the following integration capabilities:

- Automatically generate files for synthesis and simulation and automatically launch other EDA tools (**Assignments > Settings > EDA Tool Settings > NativeLink Settings**). The Intel Quartus Prime Pro Edition software does not support NativeLink.
- Compile all RTL and gate-level simulation model libraries for your device, simulator, and design language automatically (**Tools > Launch Simulation Library Compiler**).
- Include files generated by other EDA design entry or synthesis tools in your project as synthesized design files (**Project > Add/Remove File from Project**) .
- Automatically generate optional files for board-level verification (**Assignments > Settings > EDA Tool Settings**).

Figure 15. EDA Tool Settings



2.7. Exporting Compilation Results

When you run compilation, the Compiler preserves a database of results in a Quartus Database File (`.qdb`). The `.qdb` contains the data to reproduce similar results in another project, or in a later software version. You can export your project's compilation results database for import to another project or migration to a later Intel Quartus Prime software version.

You can export the `.qdb` for your entire project or for a design partition that you define in your project. When migrating the database for an entire project, you can export the compilation database in a *version-compatible* format to ensure compatibility for import to a later software version.

Table 4. Exporting Compilation Results

To Export Compilation Results For	Method	Description
Complete Design	Click **Project ➤ Export Database**	Saves compilation results for the entire project in a version-compatible format. You can migrate the results to a later version of the Intel Quartus Prime software.
Design Partition	Click **Project ➤ Export Design Partition**	Saves compilation results for a design partition as a Quartus Prime Exported Partition File (`.qxp`) that you can import to another project. You can export the results for the post-fit or post-synthesis netlist.

Related Information

Project Database Commands on page 32

2.7.1. Exporting a Version-Compatible Compilation Database

To export a project compilation database to a format that ensures version-compatibility with a later version of the Intel Quartus Prime software:

1. In the Intel Quartus Prime software, open the project that you want to export.
2. Generate synthesis or final compilation results by running one of the following commands:
   - Click **Processing ➤ Start ➤ Start Analysis & Synthesis** to generate synthesized compilation results.
   - Click **Processing ➤ Start Compilation** to generate final compilation results.
3. Click **Project ➤ Export Database**, specify the **Export directory** name, and click **OK**. The database files export to the location you specify. You can now import this exported database into a later version of the Intel Quartus Prime software.

Note: You can turn on **Assignments > Settings > Compilation Process Settings > Export version-compatible database** if you want to always export a version-compatible database following compilation.

2.7.2. Importing a Version-Compatible Compilation Database

Follow these steps to import a project compilation database into a newer version of the Intel Quartus Prime software:

1. Export a version-compatible compilation database for a complete design, as Exporting a Version-Compatible Compilation Database on page 24 describes.
2. In a newer version of the Intel Quartus Prime software, open the original project. Click **Yes** if prompted to open a project created with a different software version.
3. If you have previously compiled the design you want to export, click **Project ➤ Clean Project** to clean the old compilation database before import.
4. Click **Project > Import Database** and select the exported database directory (by default, `<project directory>/export_db/`). **Timing analysis mode** is available to disable legality checks for certain configuration rule changes from prior versions of the Intel Quartus Prime software. Enable this option only if your design does not successfully import with the option disabled. After you import a design with **Timing analysis mode**, you cannot the project to generate programming files.
5. Click **OK**.

2.7.3. Exporting a Design Partition

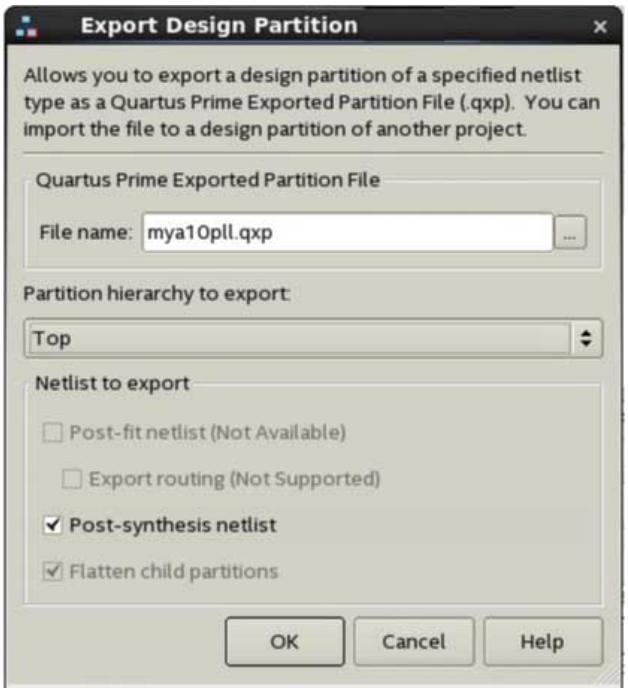
The following steps describe export of design partitions that you create in your project.

Manual Design Partition Export

Follow these steps to manually export a design partition with the **Export Design Partition** dialog box:

1. Open a project and create one or more design partitions.
2. Run synthesis (**Processing ➤ Start ➤ Start Analysis & Synthesis**) or full compilation (**Processing ➤ Start Compilation**), depending on which compilation results that you want to export.
3. Click **Project ➤ Export Design Partition**, and specify one or more options in the **Export Design Partition** dialog box:

Figure 16. Export Design Partition Dialog Box



- Under **Quartus Prime Exported Partition File**, specify a file name.
- Select the **Partition hierarchy** to export.
- Under **Netlist to export**, select the **Post-fit netlist** or **Post-synthesis netlist** for export.

4. Click **OK**. The compilation results for the design partition exports to the file that you specify.

2.7.4. Clearing Compilation Results

You can clean the project database if you want to remove prior compilation results for all project revisions or for specific revisions. For example, you must clear previous compilation results before importing a version-compatible database to an existing project.

1. Click **Project > Clean Project**.
2. Select **All revisions** to clear the databases for all revisions of the current project, or specify a **Revision name** to clear only the revision's database you specify.
3. Click **OK**. A message indicates when the database is clean.

Figure 17. Clean Project Dialog Box Cleans the Project Database

2.8. Migrating Projects Across Operating Systems

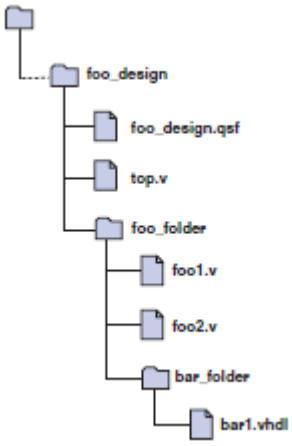
Consider the following cross-platform issues when moving your project from one operating system to another (for example, from Windows* to Linux*).

2.8.1. Migrating Design Files and Libraries

Consider file naming differences when migrating projects across operating systems.

- Use appropriate case for your platform in file path references.
- Use a character set common to both platforms.
- Do not change the forward-slash (`/`) and back-slash (`\`) path separators in the `.qsf`. The Intel Quartus Prime software automatically changes all back-slash (`\`) path separators to forward-slashes (`/`)in the `.qsf`.
- Observe the target platform's file name length limit.
- Use underscore instead of spaces in file and directory names.
- Change library absolute path references to relative paths in the `.qsf`.
- Ensure that any external project library exists in the new platform's file system.
- Specify file and directory paths as relative to the project directory. For example, for a project titled `foo_design`, specify the source files as: `top.v`, `foo_folder /foo1.v`, `foo_folder /foo2.v`, and `foo_folder/bar_folder/bar1.vhdl`.
- Ensure that all the subdirectories are in the same hierarchical structure and relative path as in the original platform.

Figure 18. All Inclusive Project Directory Structure

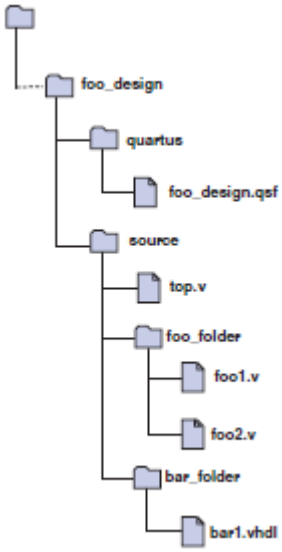


2.8.1.1. Use Relative Paths

Express file paths using relative path notation (`../`).

For example, in the directory structure shown you can specify `top.v` as `../source/top.v` and `foo1.v` as `../source/foo_folder/foo1.v`.

Figure 19. Intel Quartus Prime Project Directory Separate from Design Files



2.8.2. Design Library Migration Guidelines

The following guidelines apply to library migration across computing platforms:

1. The project directory takes precedence over the project libraries.
2. For Linux, the Intel Quartus Prime software creates the file in the `altera.quartus` directory under the <*home*> directory.
3. All library files are relative to the libraries. For example, if you specify the `user_lib1` directory as a project library and you want to add the `/user_lib1/foo1.v` file to the library, you can specify the `foo1.v` file in the `.qsf` as `foo1.v`. The Intel Quartus Prime software includes files in specified libraries.
4. If the directory is outside of the project directory, an absolute path is created by default. Change the absolute path to a relative path before migration.
5. When copying projects that include libraries, you must either copy your project library files along with the project directory or ensure that your project library files exist in the target platform.
   - On Windows, the Intel Quartus Prime software searches for the `quartus2.ini` file in the following directories and order:
   - `USERPROFILE`, for example, `C:\Documents and Settings\<user name>`
   - Directory specified by the TMP environmental variable
   - Directory specified by the TEMP environmental variable
   - Root directory, for example, `C:\`

2.9. Archiving Projects

You can optionally save the elements of a project in a single, compressed Intel Quartus Prime Archive File (`.qar`) by clicking **Project > Archive Project**. The `.qar` preserves logic design, project, and settings files required to restore the project.

Use this technique to share projects between designers, or to transfer your project to a new version of the Intel Quartus Prime software, or to Intel support. Optionally add compilation results, Platform Designer system files, and third-party EDA tool files to the archive.

If you restore the archive in a different version of the Intel Quartus Prime software, you must include the original `.qdf` in the archive to preserve original compilation results.

Related Information

Project Archive Commands on page 32

2.9.1. Manually Adding Files To Archives

Follow these steps to add files to a project archive manually:

1. Click **Project ➤ Archive Project** and specify the archive file name.
2. Click **Advanced**.
3. Select the **File set** for archive or select **Custom**. Turn on **File subsets** for the archive.
4. Click **Add** and select Platform Designer system or EDA tool files. Click **OK**.
5. Click **Archive**.

2.9.2. Archiving Compilation Results

Optionally include compilation results in a project archive to avoid recompilation and preserve original results in the restored project. To archive compilation results, export the post-synthesis or post-fit version compatible database and include this file in the archive.

1. Export the project database.
2. Click **Project > Archive Project** and specify the archive file name.
3. Click **Advanced**.
4. Under **File subsets**, turn on **Version-compatible database files** and click **OK**.
5. Click **Archive**.

To restore an archive containing a version-compatible database, follow these steps:

1. Click **Project > Restore Archived Project**.
2. Select the archive name and destination folder and click **OK**.
3. After restoring the archived project, click **Project > Import Database** and import the version-compatible database.

Related Information

Exporting a Version-Compatible Compilation Database on page 24

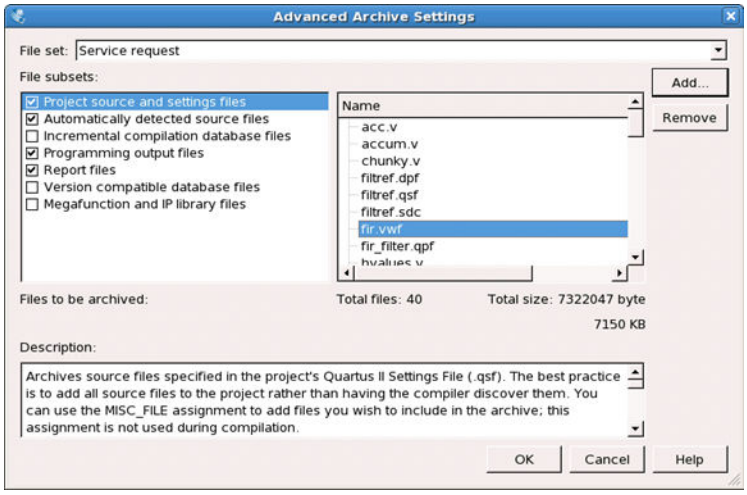
2.9.3. Archiving Projects for Service Requests

When archiving projects for a service request, include all needed file types for proper debugging by customer support.

To identify and include appropriate archive files for an Intel service request:

1. Click **Project > Archive Project** and specify the archive file name.
2. Click **Advanced**.
3. In **File set**, select **Service request** to include files for Intel Support.
   - Project source and setting files (`.v`, `.vhd`, `.vqm`, `.qsf`, `.sdc`, `.qip`, `.qpf`, `.cmp`)
   - Automatically detected source files (various)
   - Programming output files (`.jdi`, `.sof`, `.pof`)
   - Report files (`.rpt`, `.pin`, `.summary`, `.smsg`)
   - Platform Designer system and IP files (`.qsys`, `.qip`)
4. Click **OK**, and then click **Archive**.

Figure 20. Archiving Project for Service Request



2.9.4. Using External Revision Control

Your project may involve different team members with distributed responsibilities, such as sub-module design, device and system integration, simulation, and timing closure. In such cases, it may be useful to track and protect file revisions in an external revision control system.

While Intel Quartus Prime project revisions preserve various project setting and constraint combinations, external revision control systems can also track and merge RTL source code, simulation testbenches, and build scripts. External revision control supports design file version experimentation through branching and merging different versions of source code from multiple designers. Refer to your external revision control documentation for setup information.

2.9.4.1. Files to Include In External Revision Control

Include the following project file types in external revision control systems:

- Logic design files (`.v`, `.vdh`, `.bdf`, `.edf`, `.vqm`)
- Timing constraint files (.sdc)
- Quartus project settings and constraints (`.qdf`, `.qpf`, `.qsf`)
- IP files (`.ip`, `.v`, `.sv`, `.vhd`, `.qip`, `.sip`, `.qsys`)
- Platform Designer (Standard)-generated files (`.qsys`, `.ip`, `.sip`)
- EDA tool files (`.vo`, `.vho`)

Generate or modify these files manually if you use a scripted design flow. If you use an external source code control system, check-in project files anytime you modify assignments and settings.

2.10. Command-Line Interface

You can optionally use command-line executables or scripts to run project commands, rather than using the GUI. This technique can be helpful if you have many settings and wish to track them in a single file or spreadsheet for iterative comparison. The `.qsf` supports only a limited subset of Tcl commands. Therefore, pass settings and constraints using a Tcl script:

1. Create a text file with the extension `.tcl` that contains your assignments in Tcl format.
2. Source the Tcl script file by adding the following line to the .qsf: `set_global_assignment -name SOURCE_TCL_SCR IPT_FILE <file name>`.

2.10.1. Project Revision Commands

create_revision Command

`create_revision` defines the properties of a new project revision.

```
create_revision <name> -based_on <revision_name> -copy_results -set_current
```

Table 5. create_revision Command Options

Option	Description
`based_on` (optional)	Specifies the revision name on which the new revision bases its settings.
`set_current` (optional)	Sets the new revision as the current revision.
`copy_results`	Copies the results from the `based_on` revision.
`-new_rev_type`	Specifies a `base` or `impl` (implementation) type for a new revision.
`root_partition_qdb_file`	Specifies the name of a static region .qdb if already known when creating a revision.

get_project_revisions Command

`get_project_revisions` returns a list of all revisions in the project.

```
get_project_revisions <project_name>
```

delete_revision Command

`delete_revision` deletes the revision you specify from your project.

```
delete_revision <revision name>
```

set_current_revision Command

`set_current_revision` sets the revision you specify as the current revision.

```
set_current_revision -force <revision name>
```

Related Information

Optimize Settings with Project Revisions on page 18

2.10.2. Project Archive Commands

project_archive Command

`project_archive` archives your project into a single, compressed `.qar` file.

```
project_archive <name>.qar
```

Table 6. project_archive Command Options

Options	Description
`-all_revisions`	Includes all revisions of the current project in the archive.
`-auto_common_directory`	Preserves original project directory structure in archive.
`-common_directory /<name>`	Preserves original project directory structure in specified subdirectory.
`-include_libraries`	Includes libraries in archive.
`-include_outputs`	Includes output files in archive.
`-use_file_set <file_set>`	Includes specified fileset in archive.
`-version_compatible_database`	Includes version-compatible database files in archive.

Note: Version-compatible databases are not available for some device families. If you require the database files to reproduce the compilation results in the same Intel Quartus Prime software version, use the `-use_file_set full_db` option to archive the complete database.

restore_archive Command

Restores an archived project to a destination directory with optional overwriting of current contents.

```
project_restore <name>.qar -destination <directory name> -overwrite
```

Related Information

Archiving Projects on page 28

2.10.3. Project Database Commands

Related Information

Exporting Compilation Results on page 23

2.10.3.1. Import and Export Version-Compatible Databases from a Flow Package

The following are Tcl commands from the `flow` package to import or export version-compatible databases. If you use the `flow` package, you must specify the database directory variable name. `flow` and `database_manager` packages contain commands to manage version-compatible databases.

- `set_global_assignment -name VER_COMPATIBLE_DB_DIR <directory>`
- *`execute_flow –flow export_database`*
- execute_flow –flow import_database

2.10.3.2. quartus_cdb and quartus_sh Executables to Manage Version-Compatible Databases

Use the following commands to manage version-compatible databases:

- `quartus_cdb <project> -c <revision> --export_database=<directory>`
- `quartus_cdb <project> -c <revision> --import_database=<directory>`
- `quartus_sh –flow export_database <project> -c \ <revision>`
- `quartus_sh –flow import_database <project> -c \ <revision>`

2.10.4. Project Library Commands

Use the following commands to script project library changes.

2.10.4.1. Specify Project Libraries With SEARCH_PATH Assignment

In Tcl, use commands in the `:: quartus ::project` package to specify project libraries, and the `set_global_assignment` command.

Use the following commands to script project library changes:

- `set_global_assignment -name SEARCH_PATH "../other_dir/library1"`
- `set_global_assignment -name SEARCH_PATH "../other_dir/library2"`
- `set_global_assignment -name SEARCH_PATH "../other_dir/library3"`

2.10.4.2. Report Specified Project Libraries Commands

To report any project libraries specified for a project and any global libraries specified for the current installation of the Quartus software, use the `get_global_assignment` and `get_user_option` Tcl commands.

Use the following commands to report specified project libraries:

- `get_global_assignment -name SEARCH_PATH`
- `get_user_option -name SEARCH_PATH`

2.10.4.3. Generate Version-Compatible Database After Compilation

Use the following commands to generate a version-compatible database after compilation:

- `set_global_assignment -name AUTO_EXPORT_VER_COMPATIBLE_DB ON`
- `set_global_assignment-name VER_COMPATIBLE_DB_DIR <directory>`

2.11. Managing Projects Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	• Subdivided "Exporting, Archiving, and Migrating Projects" into separate sections. • Added "Specifying the Target Device or Board" topic. • Divided "Introduction to Intel FPGA IP Cores" into separate chapter. • Moved "IP Core Best Practices" topic to *Introduction to Intel FPGA IP Cores* chapter. • Moved "Factors Affecting Compilation Results" topic to *Design Compilation: Intel Quartus Prime Standard Edition User Guide*.
2018.02.11	18.0.0	• Added description of | as root partition hierarchy path in Design Partitions Window. • Removed "Scripting IP Simulation" and "Generating a Combined Simulation Script" topics. These features are supported only for Intel Arria 10 devices in Intel Quartus Prime Standard Edition. • Added link to "Scripting IP Simulation" in the *Introduction to Intel FPGA IP Cores*.

Date	Version	Changes
2017.11.06	17.1.0	• Revised product branding for Intel standards. • Changed instances of *Qsys* to *Platform Designer (Standard)* • Revised topics on Intel FPGA IP Evaluation Mode (formerly OpenCore). • Removed `-compatible` attribute from `export_design` command content. • Updated IP Core Upgrade Status table with new icons, and added row for IP Component Outdated status.
2017.05.08	17.0.0	• Added topic on Back-Annotate Assignments command.
2016.10.31	16.1.0	• Updated screenshots.
2016.05.03	16.0.0	Removed statements about serial equivalence when using multiple processors.
2016.02.09	15.1.1	• Clarified instructions for Generating a Combined Simulator Setup Script. • Clarified location of **Save project output files in specified directory** option.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2015.05.04	15.0.0	• Added description of design templates feature. • Updated screenshot for DSE II GUI. • Added qsys_script IP core instantiation information. • Described changes to generating and processing of instance and entity names.

continued...

Date	Version	Changes
		• Added description of upgrading IP cores at the command line. • Updated procedures for upgrading and migrating IP cores. • Gate level timing simulation supported only for Cyclone IV and Stratix IV devices.
2014.12.15	14.1.0	• Updated content for DSE II GUI and optimizations. • Added information about new **Assignments ➤ Settings ➤ IP Settings** that control frequency of synthesis file regeneration and automatic addition of IP files to the project.
2014.08.18	14.0a10.0	• Added information about specifying parameters for IP cores targeting Arria 10 devices. • Added information about the latest IP output for version 14.0a10 targeting Arria 10 devices. • Added information about individual migration of IP cores to the latest devices. • Added information about editing existing IP variations.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor.
November 2013	13.1.0	• Conversion to DITA format
May 2013	13.0.0	• Overhaul for improved usability and updated information.
June 2012	12.0.0	• Removed survey link. • Updated information about `VERILOG_INCLUDE_FILE`.
November 2011	10.1.1	Template update.
December 2010	10.1.0	• Changed to new document template. • Removed Figure 4–1, Figure 4–6, Table 4–2. • Moved "Hiding Messages" to Help. • Removed references about the `set_user_option` command. • Removed Classic Timing Analyzer references.

Related Information

Documentation Archive

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

3. Design Planning

3.1. Design Planning

Design planning is an essential step in advanced FPGA design. System architects must consider the target device characteristics in order to plan for interface I/O, integration of IP, on-chip debugging tools, and use of other EDA tools. Designers must consider device power consumption and programming methods when planning the layout. You can solve potential problems early in the design cycle by following the design planning considerations in this chapter.

By default, the Intel Quartus Prime software optimizes designs for the best overall results; however, you can adjust settings to better optimize one aspect of your design, such as performance, routability, area, or power utilization. Consider your own design priorities and trade-offs when reviewing the techniques in this chapter. For example, certain device features, density, and performance requirements can increase system cost. Signal integrity and board issues can impact I/O pin locations. Power, timing performance, and area utilization all affect one another. Compilation time is affected when optimizing these priorities.

Determining your design priorities early on helps you to choose the best device, tools, features, and methodologies for your design.

3.2. Create a Design Specification and Test Plan

Before you create your design logic or complete your system design, it is best practice to create detailed design specifications that define the system, specify the I/O interfaces for the FPGA, identify the different clock domains, and include a block diagram of basic design functions.

In addition, creating a test plan helps you to design for verification and ease of manufacture. For example, your test plan can include validation of interfaces incorporated in your design. To perform any built-in self-test functions to drive interfaces, you can use a UART interface with a Nios® II processor inside the FPGA device.

If more than one designer contributes to the design, consider a common design directory structure or source control system to make design integration easier. Consider whether you want to standardize on an interface protocol for each design block.

3.3. Plan for the Target Device

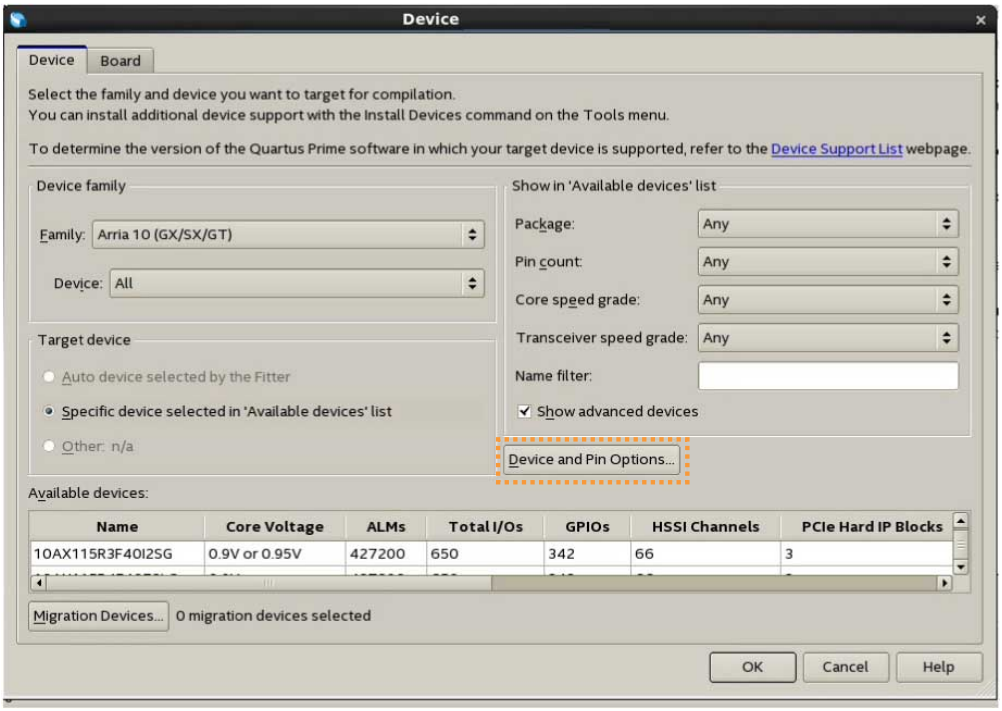
Intel offers a broad portfolio of FPGA and PLD devices. The Intel device that you select determines factors of performance, density, and board layout. To avoid costly design changes, it is best to carefully consider and determine the target device family early in

the design cycle. Intel FPGA device families differ in cost, size, density, performance, power consumption, packaging, I/O standards, and other factors. Select the device family that best suits your most critical design requirements.

Device Family Selection Guidelines

- Refer to the Product Selector tool on the Intel website to quickly find and compare the specifications and features of Intel FPGA devices and development kits.
- Once you identify the target device family, refer to the device family technical documentation for detailed device characteristics. Each device family includes complete documentation, including a datasheet and user guide or handbook. You can also view a summary of each device's resources by selecting a device in the **Device** dialog box (**Assignments ➤ Device**)

Figure 21. Device Dialog Box



- Consider whether the device family meets any requirements you have for high-speed transceivers, global or regional clock networks, and the number of phase-locked loops (PLLs)
- Consider the density requirements of your design. Devices with more logic resources and higher I/O counts can implement larger and more complex designs, but at a higher cost. Smaller devices use lower static power. Select a device larger than what your design requires if you may want to add more logic later in the design cycle, or to reserve logic and memory for on-chip debugging.
- Consider requirements for types of dedicated logic blocks, such as memory blocks of different sizes, or digital signal processing (DSP) blocks to implement certain arithmetic functions.

Related Information

Product Selector Guide Tool
To help you choose your device.

3.3.1. Device Migration Planning

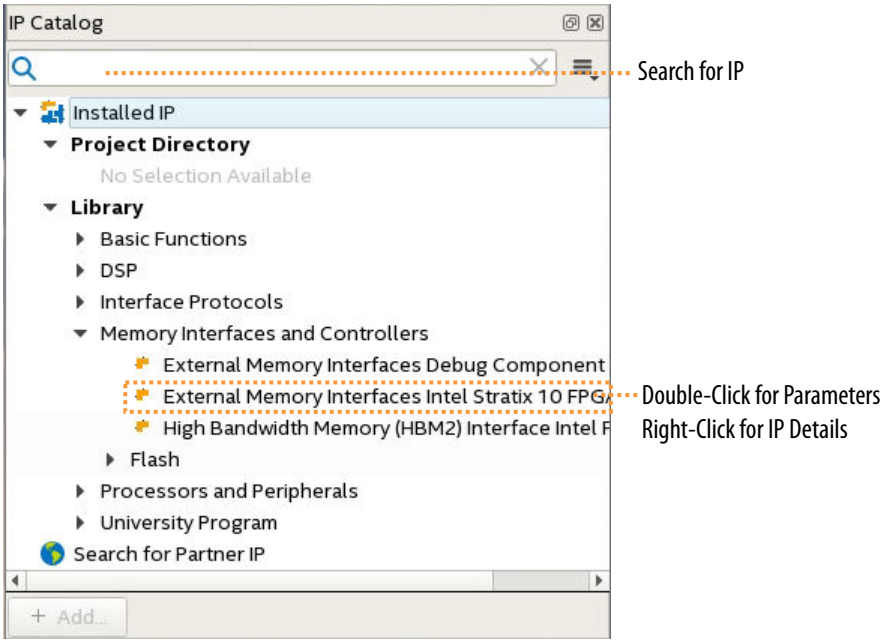
Determine whether you want to migrate your design to another device density to allow flexibility when your design nears completion. You may want to target a smaller (and less expensive) device and then move to a larger device if necessary to meet your design requirements. Other designers may prototype their design in a larger device to reduce optimization time and achieve timing closure more quickly, and then migrate to a smaller device after prototyping. If you want the flexibility to migrate your design, you must specify these migration options in the Intel Quartus Prime software at the beginning of your design cycle.

Selecting a migration device impacts pin placement because some pins may serve different functions in different device densities or package sizes. If you make pin assignments in the Intel Quartus Prime software, the Pin Migration View in the Pin Planner highlights pins that change function between your migration devices.

3.4. Plan for Intellectual Property Cores

Intel and third-party intellectual property (IP) partners offer a large selection of standardized IP cores optimized for Intel FPGA devices. The IP you select often affects system design and performance, especially if the FPGA interfaces with other devices in the system. Plan which I/O interfaces or other blocks in the system that you want to implement using IP cores. Whenever possible, plan to incorporate these functions into your design using Intel FPGA IP cores, many of which are available for production use in the Intel Quartus Prime software without additional license.

Figure 22. IP Catalog



For IP cores that require additional license for production use, the Intel FPGA IP Evaluation Mode, allows you to program the FPGA to verify the IP in the hardware before you purchase the IP license. Refer to Introduction to Intel FPGA IP Cores on page 52 for general information on using Intel FPGA IP cores.

Related Information

- Introduction to Intel FPGA IP Cores on page 52
- Intel FPGA IP Portfolio Web Page
  For descriptions and documentation for all available Intel FPGA and partner IP cores.

3.5. Plan for Standard Interfaces

To reduce design iterations and costly design changes, plan for use of standard interfaces in system design. Using standard interfaces ensures compatibility between design blocks from different design teams or vendors. Standard interfaces simplify the interface logic to each design block, and enable individual team members to test their individual design blocks against the specification for the interface protocol to ease system integration.

You can use the Intel Quartus Prime Platform Designer system integration tool to use standard interfaces and speed-up system-level integration. Platform Designer components use Avalon® standard interfaces for physical connections, allowing you to connect any logical device (either on-chip or off-chip) that has an Avalon interface. Platform Designer allows you to define system components in a GUI, and then automatically generates the required interconnect logic, along with clock-crossing and width adapters.

The Avalon standard includes two interface types:

- Avalon Memory-Mapped (Avalon-MM)—allow a component to use an address-mapped read or write protocol that connects master components to slave components.
- Avalon Streaming (Avalon-ST)—enables point-to-point connections between streaming components that send and receive data using a high-speed, unidirectional system interconnect between source and sink ports.

Related Information

Creating a System with Platform Designer

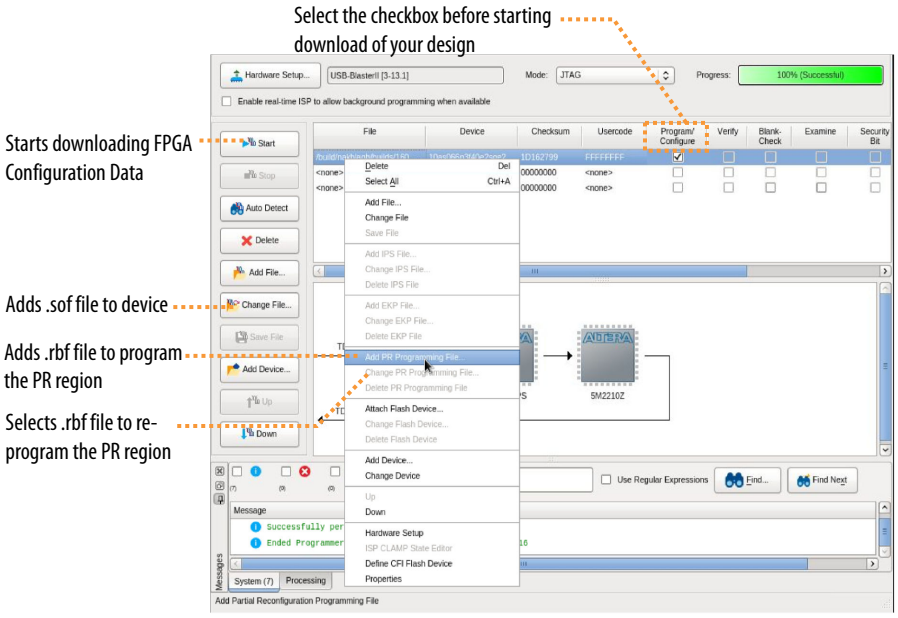
3.6. Plan for Device Programming

You must plan for the devices and hardware that you require for programming or configuration of the device. Comprehensive system planning includes determining what companion devices, if any, your system requires. Your programming or configuration method also impacts the board layout planning. For example, some programming options require a JTAG interface connection, requiring a JTAG chain on the board.

You can define a configuration scheme on the **Configuration** tab of the **Device and Pin Options** dialog box. The Intel Quartus Prime software uses the settings for the configuration scheme, configuration device, and configuration device voltage to enable

the appropriate dual purpose pins as regular I/O pins after you complete configuration. The Intel Quartus Prime software performs voltage compatibility checks of those pins during compilation of your design.

Figure 23. Intel Quartus Prime Programmer



The technical documentation for each device family describes the available configuration options.

3.7. Plan for Device Power Consumption

You can use the Intel Quartus Prime power estimation and analysis tools to estimate power consumption and guide PCB board and system design. You must accurately estimate device power consumption to develop an appropriate power budget and to design the power supplies, voltage regulators, heat sink, and cooling system. You can use the Early Power Estimator (EPE) spreadsheet to estimate power consumption before running a compilation or creating any source code. Then, you can use the Intel Quartus Prime Power Analyzer to perform a more accurate analysis after your design is complete.

Note: Because power consumption is heavily dependent on actual design and environmental conditions, make sure to verify the actual power consumption during device operation.

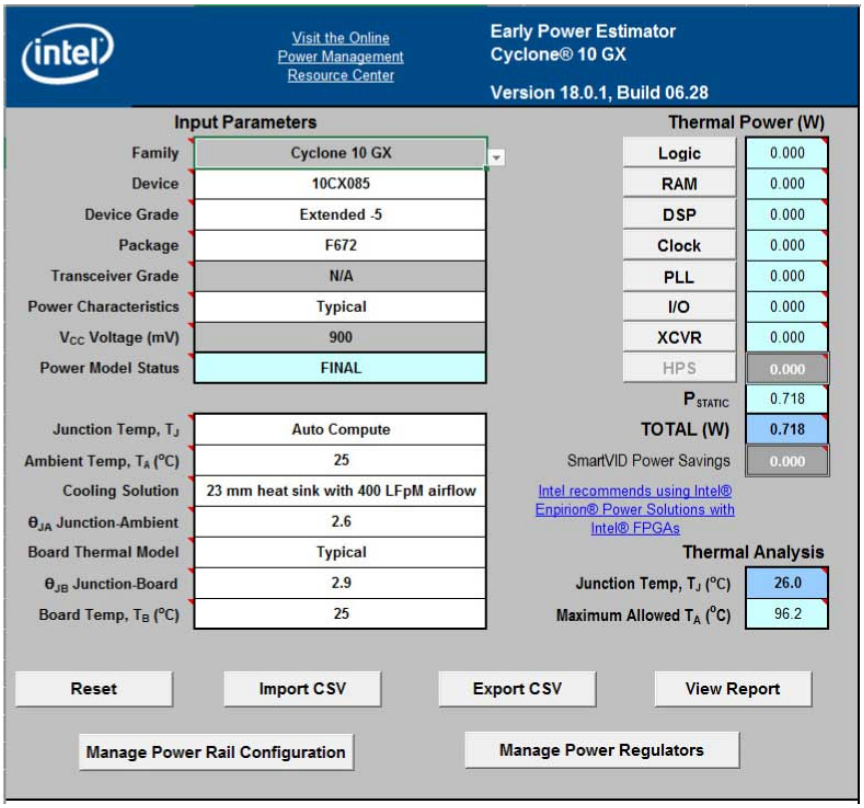
Power estimation and analysis helps you ensure that your design satisfies thermal and power supply requirements:

- Thermal—ensure that the cooling solution is sufficient to dissipate the heat generated by the device. The computed junction temperature must fall within normal device specifications.
- Power supply—ensure that the power supplies provide adequate current to support device operation.

Early Power Estimator (EPE) Spreadsheet

The Early Power Estimator (EPE) spreadsheet allows you to estimate power utilization for your design. Estimating power consumption early in the design cycle allows planning of power budgets and avoids unexpected results when designing the PCB.

Figure 24. Early Power Estimator (EPE) Spreadsheet



You can manually enter data into the EPE spreadsheet, or use the Intel Quartus Prime software to generate device resource information for your design.

To manually enter data into the EPE spreadsheet, enter the device resources, operating frequency, toggle rates, and other parameters for your design. If you do not have an existing design, estimate the number of device resources used in your design, and then enter the data into the EPE spreadsheet manually.

If you have an existing design or a partially completed design, you can use the Intel Quartus Prime software to generate the Early Power Estimator File (`.txt`, `.csv`) to assist you in completing the EPE spreadsheet.

The EPE spreadsheet includes the Import Data macro that parses the information in the EPE File and transfers the information into the spreadsheet. If you do not want to use the macro, you can manually transfer the data into the EPE spreadsheet. For example, after importing the EPE File information into the EPE spreadsheet, you can add device resource information. If the existing Intel Quartus Prime project represents only a portion of your full design, manually enter the additional device resources you use in the final design.

Intel Quartus Prime Power Analyzer

After you complete your design, you can use the Intel Quartus Prime Power Analyzer to perform a complete post-fit power analysis to check the power consumption more accurately. The Power Analyzer provides an accurate estimation of power, ensuring that thermal and supply limitations are met.

Related Information

Early Power Estimator and Power Analyzer Web Page

3.8. Plan for Interface I/O Pins

In many design environments, FPGA designers want to plan the top-level FPGA I/O pins early to help board designers begin the PCB design and layout. The I/O capabilities and board layout guidelines of the FPGA device influence pin locations and other types of assignments. If the board design team specifies an FPGA pin-out, the pin locations must be verified in the FPGA placement and routing software to avoid board design changes.

You can create a preliminary pin-out for an Intel FPGA with the Intel Quartus Prime Pin Planner before you develop the source code, based on standard I/O interfaces (such as memory and bus interfaces) and any other I/O requirements for your system.

The Intel Quartus Prime I/O Assignment Analysis checks that the pin locations and assignments are supported in the target FPGA architecture. You can then use I/O Assignment Analysis to validate I/O-related assignments that you create or modify throughout the design process. When you compile your design in the Intel Quartus Prime software, I/O Assignment Analysis runs automatically in the Fitter to validate that the assignments meet all the device requirements and generates error messages.

Early in the design process, before creating the source code, the system architect has information about the standard I/O interfaces (such as memory and bus interfaces), the IP cores in your design, and any other I/O-related assignments defined by system requirements. You can use this information with the **Early Pin Planning** feature in the Pin Planner to specify details about the design I/O interfaces. You can then create a top-level design file that includes all I/O information.

The Pin Planner interfaces with the IP core parameter editor, which allows you to create or import custom IP cores that use I/O interfaces. You can configure how to connect the functions and cores to each other by specifying matching node names for selected ports. You can create other I/O-related assignments for these interfaces or other design I/O pins in the Pin Planner, as described in this section. The Pin Planner creates virtual pin assignments for internal nodes, so internal nodes are not assigned to device pins during compilation.

After analysis and synthesis of the newly generated top-level wrapper file, use the generated netlist to perform I/O Analysis with the **Start I/O Assignment Analysis** command.

You can use the I/O analysis results to change pin assignments or IP parameters even before you create your design, and repeat the checking process until the I/O interface meets your design requirements and passes the pin checks in the Intel Quartus Prime software. When you complete initial pin planning, you can create a revision based on the Intel Quartus Prime-generated netlist. You can then use the generated netlist to develop the top-level design file for your design, or disregard the generated netlist and use the generated Intel Quartus Prime Settings File (`.qsf`) with your design.

During this early pin planning, after you have generated a top-level design file, or when you have developed your design source code, you can assign pin locations and assignments with the Pin Planner.

With the Pin Planner, you can identify I/O banks, voltage reference (VREF) groups, and differential pin pairings to help you through the I/O planning process. If you selected a migration device, the **Pin Migration View** highlights the pins that have changed functions in the migration device when compared to the currently selected device. Selecting the pins in the Device Migration view cross-probes to the rest of the Pin Planner, so that you can use device migration information when planning your pin assignments. You can also configure board trace models of selected pins for use in "board-aware" signal integrity reports generated with the **Enable Advanced I/O Timing** option . This option ensures that you get accurate I/O timing analysis. You can use a Microsoft Excel spreadsheet to start the I/O planning process if you normally use a spreadsheet in your design flow, and you can export a Comma-Separated Value File (`.csv`) containing your I/O assignments for spreadsheet use when you assign all pins.

When you complete your pin planning, you can pass pin location information to PCB designers. The Pin Planner is tightly integrated with certain PCB design EDA tools, and can read pin location changes from these tools to check suggested changes. Your pin assignments must match between the Intel Quartus Prime software and your schematic and board layout tools to ensure the FPGA works correctly on the board, especially if you must make changes to the pin-out. The system architect uses the Intel Quartus Prime software to pass pin information to team members designing individual logic blocks, allowing them to achieve better timing closure when they compile their design.

Start FPGA planning before you complete the HDL for your design to improve the confidence in early board layouts, reduce the chance of error, and improve the overall time to market of the design. When you complete your design, use the Fitter reports for the final sign-off of pin assignments. After compilation, the Intel Quartus Prime software generates the Pin-Out File (`.pin`), and you can use this file to verify that each pin is correctly connected in board schematics.

Related Information

- Intel Quartus Prime Standard Edition User Guide: Design Optimization
  For more information about I/O assignment and analysis.
- Mentor Graphics PCB Design Tools Support
- Cadence PCB Design Tools Support
  For more information about passing I/O information between the Intel Quartus Prime software and third-party EDA tools.

3.8.1. Simultaneous Switching Noise Analysis

Simultaneous switching noise (SSN) is a noise voltage inducted onto a victim I/O pin of a device due to the switching behavior of other aggressor I/O pins in the device.

Intel provides tools for SSN analysis and estimation, including SSN characterization reports, an Early SSN Estimator (ESE) spreadsheet tool, and the SSN Analyzer in the Intel Quartus Prime software. SSN often leads to the degradation of signal integrity by causing signal distortion, thereby reducing the noise margin of a system. You must address SSN with estimation early in your system design, to minimize later board design changes. When your design is complete, verify your board design by performing a complete SSN analysis of your FPGA in the Intel Quartus Prime software.

3.9. Plan for other EDA Tools

Your complete FPGA design flow may include third-party EDA tools in addition to the Intel Quartus Prime software. Determine which tools you want to use with the Intel Quartus Prime software to ensure that they are supported and set up properly, and that you are aware of their capabilities.

3.9.1. Third-Party Synthesis Tools

You can use supported standard third-party EDA synthesis tools to synthesize your Verilog HDL or VHDL design, and then compile the resulting output netlist file in the Intel Quartus Prime software. The Intel Quartus Prime Standard Edition software includes integrated synthesis that supports Verilog HDL, VHDL, Altera Hardware Description Language (AHDL), and schematic design entry.

Different synthesis tools may give different results for each design. To determine the best tool for your application, you can experiment by synthesizing typical designs for your application and coding style. Perform placement and routing in the Intel Quartus Prime software to get accurate timing analysis and logic utilization results.

The synthesis tool you choose may allow you to create an Intel Quartus Prime project and pass constraints, such as the EDA tool setting, device selection, and timing requirements that you specified in your synthesis project. You can save time when setting up your Intel Quartus Prime project for placement and routing.

Tool vendors frequently add new features, fix tool issues, and enhance performance for Intel devices, you must use the most recent version of third-party synthesis tools.

3.9.2. Third-Party Simulation Tools

Intel provides the Mentor Graphics ModelSim* - Intel FPGA Edition simulator with the Intel Quartus Prime software. You can also purchase the ModelSim - Intel FPGA Edition or a full license of the ModelSim software to support large designs and achieve faster simulation performance. The Intel Quartus Prime software generates both functional and timing netlist files for ModelSim and other supported third-party simulators.

Use the simulator version that your Intel Quartus Prime software version supports for best results. You must also use the model libraries provided with your Intel Quartus Prime software version. Libraries can change between versions, which might cause a mismatch with your simulation netlist.

3.10. Plan for On-Chip Debugging Tools

Consider whether to include on-chip debugging tools early in the design process. Adding the debugging tools late in the design process can be more time consuming and error prone.

The Intel Quartus Prime in-system debugging tools offer different advantages and trade-offs, depending on the characteristics of your design. Consider the following debugging requirements when planning your design to support debugging tools:

- JTAG connections—required to perform in-system debugging with JTAG tools. Plan your system and board with JTAG ports that are available for debugging.
- Additional logic resources (ALR)—required to implement JTAG hub logic. If you set up the appropriate tool early in your design cycle, you can include these device resources in your early resource estimations to ensure that you do not overload the device with logic.
- Reserve device memory—required if your tool uses device memory to capture data during system operation. To ensure that you have enough memory resources to take advantage of this debugging technique, consider reserving device memory to use during debugging.
- Reserve I/O pins—required if you use the Logic Analyzer Interface (LAI) or Signal Probe tools, which require I/O pins for debugging. If you reserve I/O pins for debugging, you do not have to later change your design or board. The LAI can multiplex signals with design I/O pins if required. Ensure that your board supports a debugging mode, in which debugging signals do not affect system operation.
- Instantiate an IP core in your HDL code—required if your debugging tool uses an Intel FPGA IP core.
- Instantiate the Signal Tap Logic Analyzer IP core—required if you want to manually connect the Signal Tap Logic Analyzer to nodes in your design and ensure that the tapped node names do not change during synthesis.

  Note: You can add the Signal Tap Logic Analyzer as a separate design partition for incremental compilation to minimize recompilation times.

Table 7. Factors to Consider When Using Debugging Tools During Design Planning Stages

Design Planning Factor	Signal Tap Logic Analyzer	System Console	In-System Memory Content Editor	Logic Analyzer Interface (LAI)	Signal Probe	In-System Sources and Probes	Virtual JTAG IP Core
JTAG connections	Yes	Yes	Yes	Yes	—	Yes	Yes
Additional logic resources	—	Yes	—	—	—	—	Yes
Reserve device memory	Yes	Yes	—	—	—	—	—
Reserve I/O pins	—	—	—	Yes	Yes	—	—
Instantiate IP core in your HDL code	—	—	—	—	—	Yes	Yes

Related Information

Intel Quartus Prime Standard Edition User Guide: Debug Tools

3.11. Plan HDL Coding Styles

When you develop complex FPGA designs, design practices and coding styles have an enormous impact on the timing performance, logic utilization, and system reliability of your device.

3.11.1. Design Recommendations

Use synchronous design practices to consistently meet your design goals. Problems with asynchronous design techniques include reliance on propagation delays in a device, incomplete timing analysis, and possible glitches.

In a synchronous design, a clock signal triggers all events. When you meet all register timing requirements, a synchronous design behaves in a predictable and reliable manner for all process, voltage, and temperature (PVT) conditions. You can easily target synchronous designs to different device families or speed grades.

Clock signals have a large effect on the timing accuracy, performance, and reliability of your design. Problems with clock signals can cause functional and timing problems in your design. Use dedicated clock pins and clock routing for best results, and if you have PLLs in your target device, use the PLLs for clock inversion, multiplication, and division. For clock multiplexing and gating, use the dedicated clock control block or PLL clock switchover feature instead of combinational logic, if these features are available in your device. If you must use internally-generated clock signals, register the output of any combinational logic used as a clock signal to reduce glitches.

The Design Assistant in the Intel Quartus Prime software is a design-rule checking tool that enables you to verify design issues. The Design Assistant checks your design for adherence to Intel-recommended design guidelines. You can also use third-party lint tools to check your coding style. The Design Assistant does not support Max 10 and Intel Arria 10 devices.

Consider the architecture of the device you choose so that you can use specific features in your design. For example, the control signals should use the dedicated control signals in the device architecture. Sometimes, you might need to limit the number of different control signals used in your design to achieve the best results.

3.11.2. Recommended HDL Coding Styles

HDL coding styles can have a significant effect on the quality of results for programmable logic designs.

If you design memory and DSP functions, you must understand the target architecture of your device so you can use the dedicated logic block sizes and configurations. Follow the coding guidelines for inferring Intel FPGA IP and targeting dedicated device hardware, such as memory and DSP blocks.

Related Information

Intel Quartus Prime Standard Edition User Guide: Design Recommendations

3.11.3. Managing Metastability

Metastability problems can occur in digital design when a signal is transferred between circuitry in unrelated or asynchronous clock domains, because the designer cannot guarantee that the signal meets the setup and hold time requirements during the signal transfer.

Designers commonly use a synchronization chain to minimize the occurrence of metastable events. Ensure that your design accounts for synchronization between any asynchronous clock domains. Consider using a synchronizer chain of more than two registers for high-frequency clocks and frequently-toggling data signals to reduce the chance of a metastability failure.

You can use the Intel Quartus Prime software to analyze the average mean time between failures (MTBF) due to metastability when a design synchronizes asynchronous signals, and optimize your design to improve the metastability MTBF. The MTBF due to metastability is an estimate of the average time between instances when metastability could cause a design failure. A high MTBF (such as hundreds or thousands of years between metastability failures) indicates a more robust design. Determine an acceptable target MTBF given the context of your entire system and the fact that MTBF calculations are statistical estimates.

The Intel Quartus Prime software can help you determine whether you have enough synchronization registers in your design to produce a high enough MTBF at your clock and data frequencies.

Related Information

Managing Metastability, Intel Quartus Prime Standard Edition User Guide: Design Recommendations

3.12. Plan for Hierarchical and Team-Based Designs

The Intel Quartus Prime Compiler supports hierarchical design methodologies to reduce design compilation times and preserve performance. In a flat compilation flow, the design hierarchy is flattened without design partitions. In block-based (hierarchical) flows, you can subdivide your design by creating design partitions.

Hierarchical flows allow you to isolate, optimize, and preserve compilation results for specific design blocks, but require more design planning to ensure effective results.

3.12.1. Flat Compilation without Design Partitions

In a flat compilation flow without any design partitions, the Intel Quartus Prime software compiles the entire design in a “flat” netlist.

Although the source code may be hierarchical, the Compiler flattens and synthesizes all the design logic. Whenever you re-compile the project, the Compiler re-performs all available logic and placement optimizations on the entire design.

The flat compilation flow does not require any planning for design partitions. However, because the Intel Quartus Prime software recompiles the entire design whenever you change your design, flat design practices may require more overall compilation time for large designs. Additionally, you may find that the results for one part of the design change when you change a different part of your design. You can run **Rapid Recompile** to preserve portions of previous placement and routing in subsequent compilations. **Rapid Recompile** can reduce your compilation time in a flat or partitioned design when you make small changes to your design.

3.12.2. Incremental Compilation with Design Partitions

In an incremental compilation flow, the system architect splits a large design into partitions. When hierarchical design partitions are well chosen and placed in the device floorplan, you can speed up your design compilation time while maintaining the quality of results.

Incremental compilation preserves the compilation results and performance of unchanged partitions in the design, greatly reducing design iteration time by focusing new compilations on changed design partitions only. Incremental compilation then merges new compilation results with the previous compilation results from unchanged design partitions. Additionally, you can target optimization techniques to specific design partitions, while leaving other partitions unchanged. You can also use empty partitions to indicate that parts of your design are incomplete or missing, while you compile the rest of your design.

Third-party IP designers can also export logic blocks to be integrated into the top-level design. Team members can work on partitions independently, which can simplify the design process and reduce compilation time. With exported partitions, the system architect must provide guidance to designers or IP providers to ensure that each partition uses the appropriate device resources. Because the designs may be developed independently, each designer has no information about the overall design or how their partition connects with other partitions. This lack of information can lead to problems during system integration. The top-level project information, including pin locations, physical constraints, and timing requirements, must be communicated to the designers of lower-level partitions before they start their design.

The system architect plans design partitions at the top level and allows third-party designs to access the top-level project framework. By designing in a copy of the top-level project (or by checking out the project files in a source control environment), the designers of the lower-level block have full information about the entire project, which helps to ensure optimal results.

When you plan your design code and hierarchy, ensure that each design entity is created in a separate file so that the entities remain independent when you make source code changes in the file. If you use a third-party synthesis tool, create separate Verilog Quartus Mapping or EDIF netlists for each design partition in your synthesis tool. You may have to create separate projects in your synthesis tool, so that the tool synthesizes each partition separately and generates separate output netlist files. The netlists are then considered the source files for incremental compilation.

3.12.3. Planning Design Partitions and Floorplan Location Assignments

Partitioning a design for an FPGA requires planning to ensure optimal results when you integrate the partitions. Following Intel's recommendations for creating design partitions should improve the overall quality of results.

For example, registering partition I/O boundaries keeps critical timing paths inside one partition that can be optimized independently. When you specify the design partitions, you can use the Incremental Compilation Advisor to ensure that partitions meet Intel's recommendations.

If you have timing-critical partitions that are changing through the design flow, or partitions exported from another Intel Quartus Prime project, you can create design floorplan assignments to constrain the placement of the affected partitions. Good

partition and floorplan design helps partitions meet top-level design requirements when integrated with the rest of your design, reducing time you spend integrating and verifying the timing of the top-level design.

Related Information

Analyzing and Optimizing the Design Floorplan

3.13. Design Planning Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	• Moved information about specifying the target board to "Specifying the Target Device or Board" in *Managing Projects* chapter. • Retitled "Creating Design Specifications" to "Create a Design Specification and Test Plan." • Retitled "Selecting Intellectual Property Cores" to "Plan for Intellectual Property Cores." • Retitled "Using Standard Interfaces" to "Plan for Standard Interfaces." Corrected references to Platform Designer. • Retitled "Device Selection" to "Plan for the Target Device." Updated this content to correct Platform Designer names. • Moved "Setting Pin Assignments" to *Managing Projects* chapter as "Generating Pin Assignments for a Target Board." • Retitled "Estimating Power" to "Plan for Device Power Consumption." Reorganized this topic into sections for EPE and Power Analyzer. • Added link to "Simulator Support, *Third-Party Simulation User Guide* • Retitled "Planning for Device Programming or Configuration" to "Plan for Device Programming" • Retitled "Early Pin Planning and I/O Analysis" to "Plan for Interface I/O Pins." • Retitled "Selecting Third-Party EDA Tools" to "Plan for other EDA Tools." • Retitled "Planning for On-Chip Debugging Tools" to "Plan for On-Chip Debugging Tools." • Revised some wording in "Planning for Hierarchical and Team-Based Design" • Retitled *Design Planning with the Intel Quartus Prime Software* to *Design Planning*

Date	Version	Changes
2017.11.06	17.1.0	• Changed instances of OpenCore Plus to Intel FPGA IP Evaluation Mode. • Changed instances of Qsys to Platform Designer (Standard) (Standard)
2016.05.03	16.0.0	Added information about Development Kit selection.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2015.05.04	15.0.0	Remove support for Early Timing Estimate feature.
2014.06.30	14.0.0	Updated document format.
November 2013	13.1.0	Removed HardCopy device information.
November, 2012	12.1.0	Update for changes to early pin planning feature

continued...

Date	Version	Changes
June 2012	12.0.0	Editorial update.
November 2011	11.0.1	Template update.
May 2011	11.0.0	• Added link to System Design with Qsys in "Creating Design Specifications" on page 1–2 • Updated "Simultaneous Switching Noise Analysis" on page 1–8 • Updated "Planning for On-Chip Debugging Tools" on page 1–10 • Removed information from "Planning Design Partitions and Floorplan Location Assignments" on page 1–15
December 2010	10.1.0	• Changed to new document template • Updated "System Design and Standard Interfaces" on page 1–3 to include information about the Qsys system integration tool • Added link to the Product Selector in "Device Selection" on page 1–3 • Converted information into new table (Table 1–1) in "Planning for On-Chip Debugging Options" on page 1–10 • Simplified description of incremental compilation usages in "Incremental Compilation with Design Partitions" on page 1–14 • Added information about the Rapid Recompile option in "Flat Compilation Flow with No Design Partitions" on page 1–14 • Removed details and linked to Intel Quartus Prime Help in "Fast Synthesis and Early Timing Estimation" on page 1–16
July 2010	10.0.0	• Added new section "System Design" on page 1–3 • Removed details about debugging tools from "Planning for On-Chip Debugging Options" on page 1–10 and referred to other handbook chapters for more information • Updated information on recommended design flows in "Incremental Compilation with Design Partitions" on page 1–14 and removed "Single-Project Versus Multiple-Project Incremental Flows" heading • Merged the "Planning Design Partitions" section with the "Creating a Design Floorplan" section. Changed heading title to "Planning Design Partitions and Floorplan Location Assignments" on page 1–15 • Removed "Creating a Design Floorplan" section • Removed "Referenced Documents" section • Minor updates throughout chapter
November 2009	9.1.0	• Added details to "Creating Design Specifications" on page 1–2 • Added details to "Intellectual Property Selection" on page 1–2 • Updated information on "Device Selection" on page 1–3 • Added reference to "Device Migration Planning" on page 1–4 • Removed information from "Planning for Device Programming or Configuration" on page 1–4 • Added details to "Early Power Estimation" on page 1–5 • Updated information on "Early Pin Planning and I/O Analysis" on page 1–6 • Updated information on "Creating a Top-Level Design File for I/O Analysis" on page 1–8 • Added new "Simultaneous Switching Noise Analysis" section • Updated information on "Synthesis Tools" on page 1–9 • Updated information on "Simulation Tools" on page 1–9 • Updated information on "Planning for On-Chip Debugging Options" on page 1–10

continued...

Date	Version	Changes
		• Added new "Managing Metastability" section • Changed heading title "Top-Down Versus Bottom-Up Incremental Flows" to "Single-Project Versus Multiple-Project Incremental Flows" • Updated information on "Creating a Design Floorplan" on page 1–18 • Removed information from "Fast Synthesis and Early Timing Estimation" on page 1–18
March 2009	9.0.0	• No change to content
November 2008	8.1.0	• Changed to 8-1/2 x 11 page size. No change to content.
May 2008	8.0.0	• Organization changes • Added "Creating Design Specifications" section • Added reference to new details in the In-System Design Debugging section of volume 3 • Added more details to the "Design Practices and HDL Coding Styles" section • Added references to the new Best Practices for Incremental Compilation and Floorplan Assignments chapter • Added reference to the Intel Quartus Prime Language Templates

Related Information

Documentation Archive

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

4. Introduction to Intel FPGA IP Cores

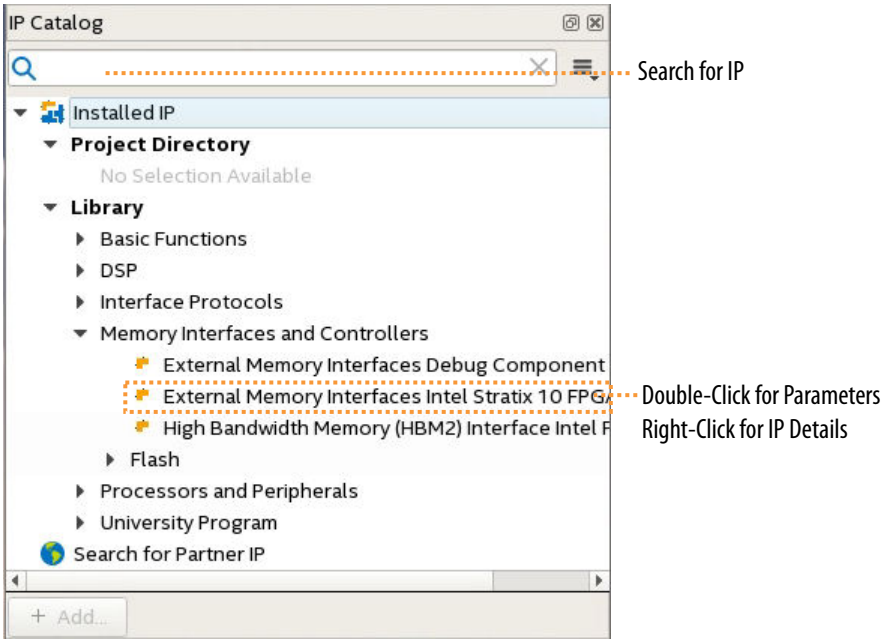
Intel and strategic IP partners offer a broad portfolio of configurable IP cores optimized for Intel FPGA devices.

The Intel Quartus Prime software installation includes the Intel FPGA IP library. Integrate optimized and verified Intel FPGA IP cores into your design to shorten design cycles and maximize performance. The Intel Quartus Prime software also supports integration of IP cores from other sources. Use the IP Catalog (**Tools ➤ IP Catalog**) to efficiently parameterize and generate synthesis and simulation files for your custom IP variation. The Intel FPGA IP library includes the following types of IP cores:

- Basic functions
- DSP functions
- Interface protocols
- Low power functions
- Memory interfaces and controllers
- Processors and peripherals

This document provides basic information about parameterizing, generating, upgrading, and simulating stand-alone IP cores in the Intel Quartus Prime software.

Figure 25. IP Catalog



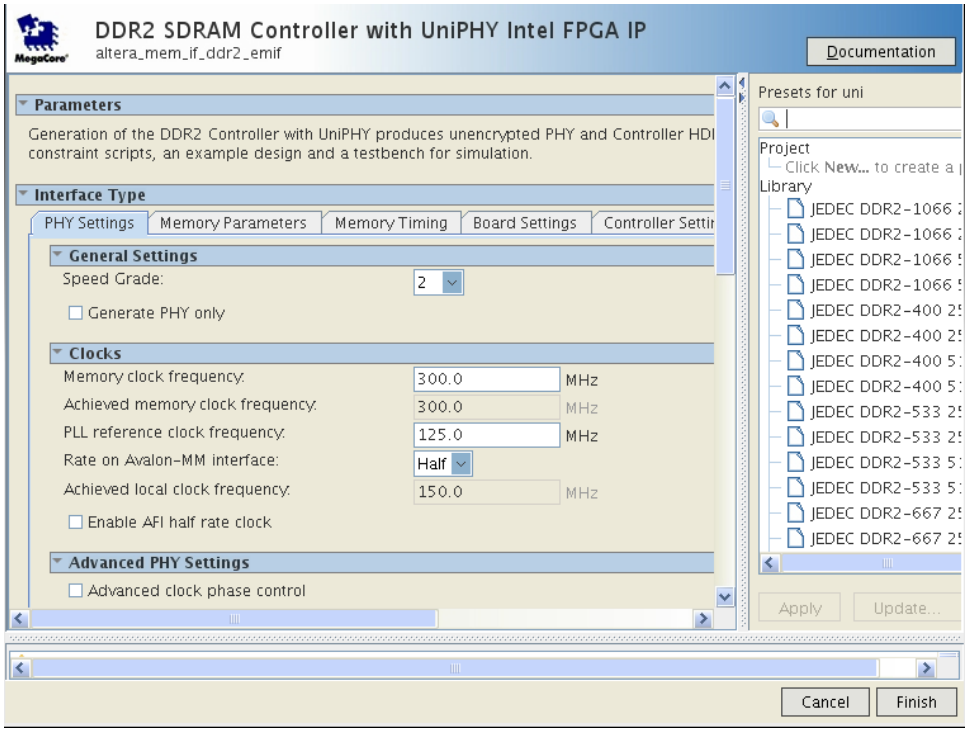
4.1. IP Catalog and Parameter Editor

The IP Catalog displays the IP cores available for your project, including Intel FPGA IP and other IP that you add to the IP Catalog search path.. Use the following features of the IP Catalog to locate and customize an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**. If you have no project open, select the **Device Family** in IP Catalog.
- Type in the Search field to locate any full or partial IP core name in IP Catalog.
- Right-click an IP core name in IP Catalog to display details about supported devices, to open the IP core's installation folder, and for links to IP documentation.
- Click **Search for Partner IP** to access partner IP information on the web.

The parameter editor generates a top-level Quartus IP file (`.qip`) for an IP variation in Intel Quartus Prime Standard Edition projects. These files represent the IP variation in the project, and store parameterization information.

Figure 26. IP Parameter Editor (Intel Quartus Prime Standard Edition)



4.1.1. The Parameter Editor

The parameter editor helps you to configure IP core ports, parameters, and output file generation options. The basic parameter editor controls include the following:

- Use the **Presets** window to apply preset parameter values for specific applications (for select cores).
- Use the **Details** window to view port and parameter descriptions, and click links to documentation.
- Click **Generate ➤ Generate Testbench System** to generate a testbench system (for select cores).
- Click **Generate ➤ Generate Example Design** to generate an example design (for select cores).

The IP Catalog is also available in Platform Designer (**View ➤ IP Catalog**). The Platform Designer IP Catalog includes exclusive system interconnect, video and image processing, and other system-level IP that are not available in the Intel Quartus Prime IP Catalog. Refer to *Creating a System with Platform Designer* or *Creating a System with Platform Designer (Standard)* for information on use of IP in Platform Designer (Standard) and Platform Designer, respectively.

Related Information

Creating a System with Platform Designer (Standard)

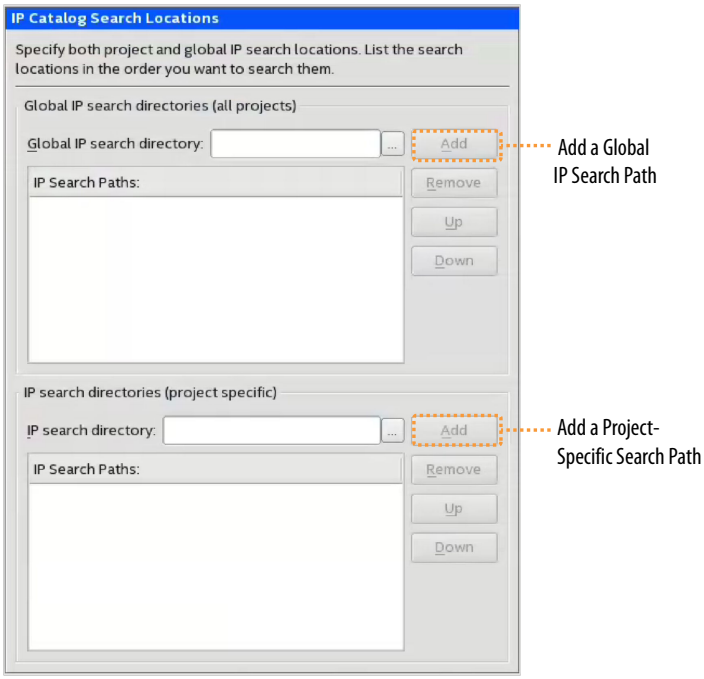
4.1.2. Adding IP Cores to the IP Catalog

By default, the IP Catalog automatically displays IP cores located in the project directory, in the default Intel Quartus Prime installation directory, and in the IP search path.

To include custom and third-party IP components in the IP Catalog:

1. Click **Tools ➤ Options ➤ IP Search Path**) to open the **IP Search Path Options** dialog box.

Figure 27. Specifying IP Search Locations



2. Click **Add** or **Remove** to add/remove new search locations.
   The Intel Quartus Prime project automatically updates to reflect these modifications.
3. Update the IP Catalog by clicking **Refresh IP Catalog** in the drop-down list

   Alternatively, in Platform Designer (Standard), click **File ➤ Refresh System** to update the IP Catalog.

4.1.3. IP General Settings

The following settings control how the Intel Quartus Prime software manages IP cores in a project:

Table 8. IP Core General Settings Locations

Setting	Description	Location
Maximum Platform Designer memory usage size	Increase if you experience slow processing for large systems, or for out of memory errors.	**Tools ➤ Options ➤ IP Settings**
IP generation HDL preference	The parameter editor generates the HDL you specify for IP variations.	
Automatically add Intel Quartus Prime IP files	Disable this option to manually add the IP files.	
IP Regeneration Policy	Controls when synthesis files regenerate for each IP variation. Typically, you **Always regenerate synthesis files for IP cores** after making changes to an IP variation.	
Additional project and global IP search locations. The Intel Quartus Prime software searches for IP cores in the project directory, in the Intel Quartus Prime installation directory, and in the IP search path.		**Tools ➤ Options ➤ IP Catalog Search Locations**

4.1.4. Installing and Licensing Intel FPGA IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 28. IP Core Installation Path

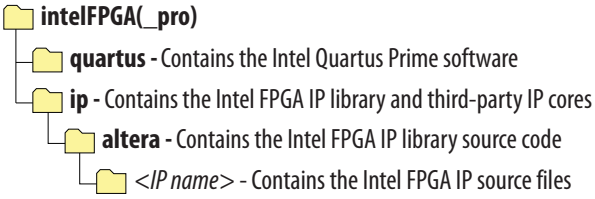


Table 9. IP Core Installation Locations

Location	Software	Platform
`<drive>:\intelFPGA_pro\quartus\ip\altera`	Intel Quartus Prime Pro Edition	Windows
`<drive>:\intelFPGA\quartus\ip\altera`	Intel Quartus Prime Standard Edition	Windows
`<home directory>:/intelFPGA_pro/quartus/ip/altera`	Intel Quartus Prime Pro Edition	Linux
`<home directory>:/intelFPGA/quartus/ip/altera`	Intel Quartus Prime Standard Edition	Linux

Note: The Intel Quartus Prime software does not support spaces in the installation path.

4.1.4.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

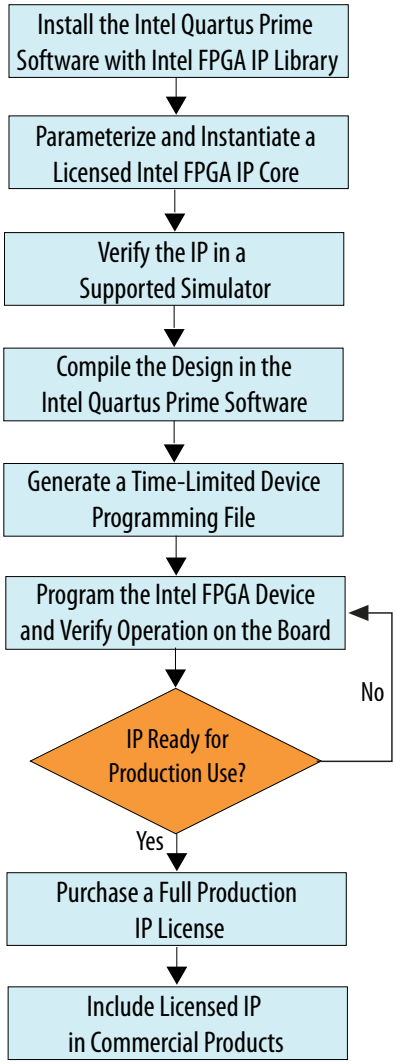
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Intel Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Intel Quartus Prime software, and requires no Intel Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Intel Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>`_time_limited.sof`) that expires at the time limit.

Figure 29. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

Intel licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the Self-Service Licensing Center or contact your local Intel FPGA representative.

The Intel FPGA Software License Agreements govern the installation and use of licensed IP cores, the Intel Quartus Prime design software, and all unlicensed IP cores.

Related Information

- Intel Quartus Prime Licensing Site
- Intel FPGA Software Installation and Licensing

4.1.5. Best Practices for Intel FPGA IP

Use the following best practices when working with Intel FPGA IP:

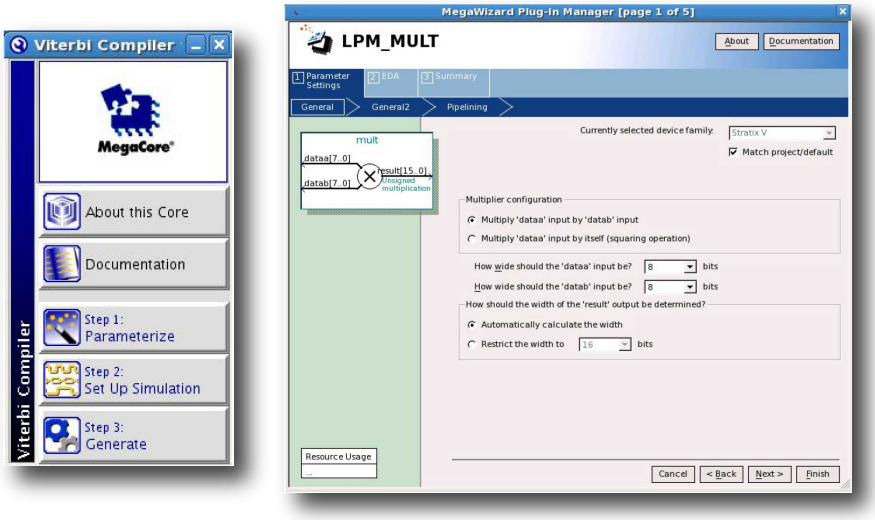
- Do not manually edit or write your own `.qsys`, `.ip`, or `.qip` file. Use the Intel Quartus Prime software tools to create and edit these files.

  Note: When generating IP cores, do not generate files into a directory that has a space in the directory name or path. Spaces are not legal characters for IP core paths or names.
- When you generate an IP core using the IP Catalog, the Intel Quartus Prime software generates a `.qsys` (for Platform Designer (Standard)-generated IP cores) or a `.ip` file (for Intel Quartus Prime Pro Edition) or a `.qip` file. The Intel Quartus Prime Pro Edition software automatically adds the generated `.ip` to your project. In the Intel Quartus Prime Standard Edition software, add the `.qip` to your project. Do not add the parameter editor generated file (`.v` or `.vhd`) to your design without the `.qsys` or `.qip` file. Otherwise, you cannot use the IP upgrade or IP parameter editor feature.
- Plan your directory structure ahead of time. Do not change the relative path between a `.qsys` file and it's generation output directory. If you must move the `.qsys` file, ensure that the generation output directory remains with the `.qsys` file.
- Do not add IP core files directly from the `/quartus/libraries/megafunctions` directory in your project. Otherwise, you must update the files for each subsequent software release. Instead, use the IP Catalog and then add the `.qip` to your project.
- Do not use IP files that the Intel Quartus Prime software generates for RAM or FIFO blocks targeting older device families (even though the Intel Quartus Prime software does not issue an error). The RAM blocks that Intel Quartus Prime generates for older device families are not optimized for the latest device families.
- When generating a ROM function, save the resulting `.mif` or `.hex` file in the same folder as the corresponding IP core's `.qsys` or `.qip` file. For example, moving all of your project's `.mif` or `.hex` files to the same directory causes relative path problems after archiving the design.
- Always use the Intel Quartus Prime `ip-setup-simulation` and `ip-make-simscript` utilities to generate simulation scripts for each IP core or Platform Designer (Standard) system in your design. These utilities produce a single simulation script that does not require manual update for upgrades to Intel Quartus Prime software or IP versions, as Simulating Intel FPGA IP Cores on page 68 describes.

4.2. Generating IP Cores (Intel Quartus Prime Standard Edition)

This topic describes parameterizing and generating an IP variation using a legacy parameter editor in the Intel Quartus Prime Standard Edition software.

Figure 30. Legacy Parameter Editors



Note: The legacy parameter editor generates a different output file structure than the Intel Quartus Prime Pro Edition software.

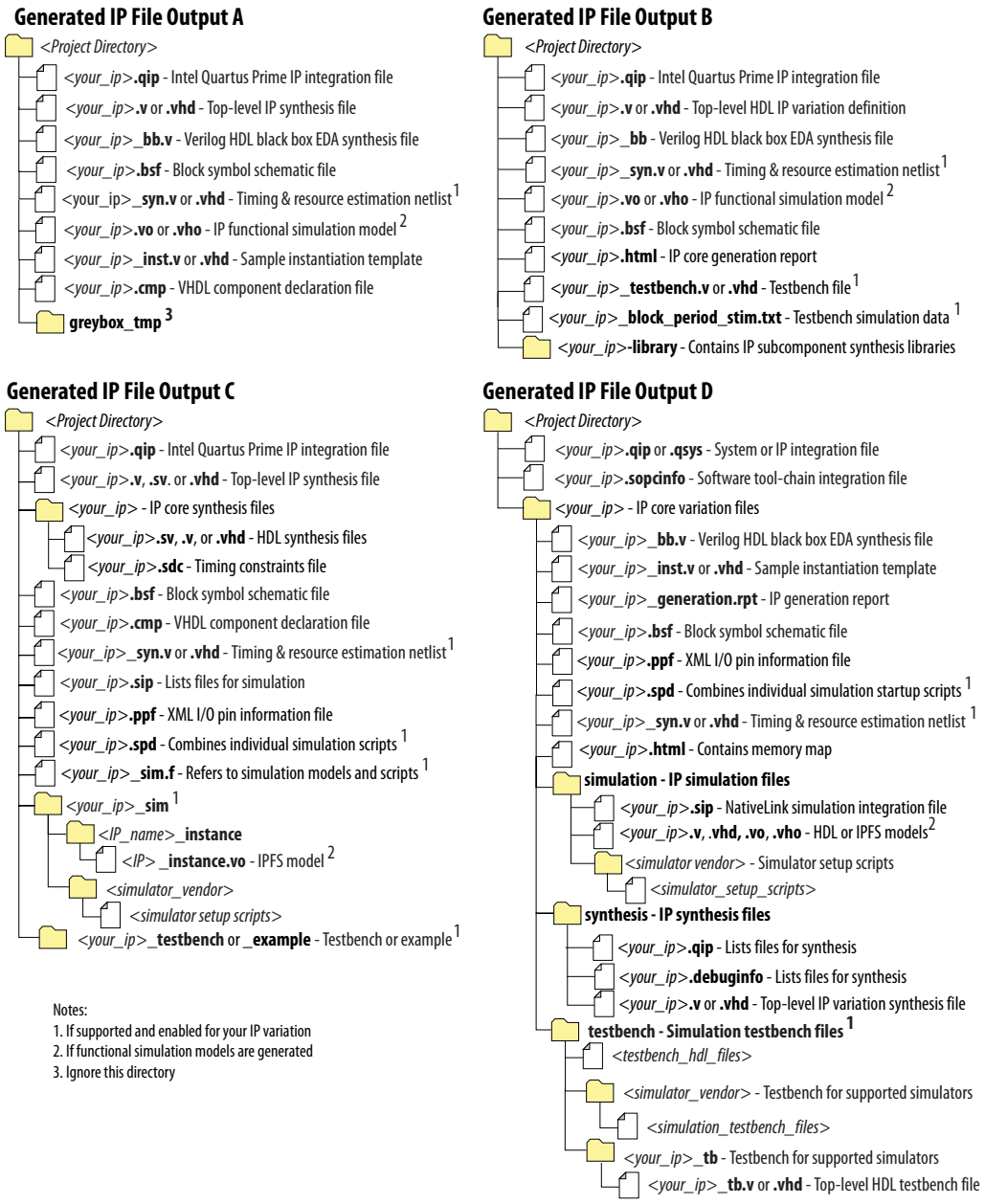
1. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name and output HDL file type for your IP variation. This name identifies the IP core variation files in your project. Click **OK**. Do not include spaces in IP variation names or paths.
3. Specify the parameters and options for your IP variation in the parameter editor. Refer to your IP core user guide for information about specific IP core parameters.
4. Click **Finish** or **Generate** (depending on the parameter editor version). The parameter editor generates the files for your IP variation according to your specifications. Click **Exit** if prompted when generation is complete. The parameter editor adds the top-level `.qip` file to the current project automatically.

   Note: For devices released prior to Intel Arria 10 devices, the generated `.qip` and `.sip` files must be added to your project to represent IP and Platform Designer systems. To manually add an IP variation generated with legacy parameter editor to a project, click **Project ➤ Add/Remove Files in Project** and add the IP variation `.qip` file.

4.2.1. IP Core Generation Output (Intel Quartus Prime Standard Edition)

The Intel Quartus Prime Standard Edition software generates one of the following output file structures for individual IP cores that use one of the legacy parameter editors.

Figure 31. IP Core Generated Files (Legacy Parameter Editors)



4.3. Modifying an IP Variation

After generating an IP core variation, use any of the following methods to modify the IP variation in the parameter editor.

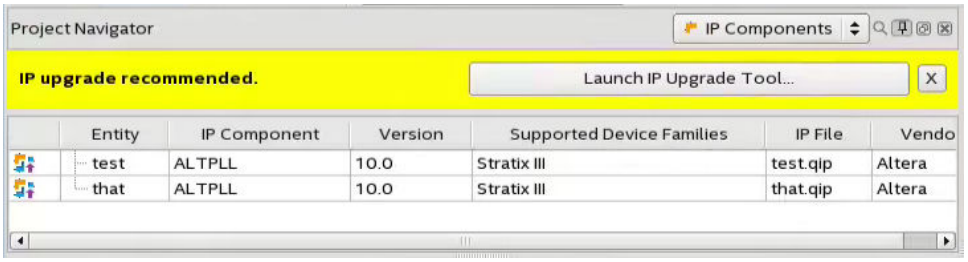
Table 10. Modifying an IP Variation

Menu Command	Action
File ➤ Open	Select the top-level HDL (`.v`, or `.vhd`) IP variation file to launch the parameter editor and modify the IP variation. Regenerate the IP variation to implement your changes.
View ➤ Utility Windows ➤ Project Navigator ➤ IP Components	Double-click the IP variation to launch the parameter editor and modify the IP variation. Regenerate the IP variation to implement your changes.
Project ➤ Upgrade IP Components	Select the IP variation and click **Upgrade in Editor** to launch the parameter editor and modify the IP variation. Regenerate the IP variation to implement your changes.

4.4. Upgrading IP Cores

Any Intel FPGA IP variations that you generate from a previous version or different edition of the Intel Quartus Prime software, may require upgrade before compilation in the current software edition or version. The Project Navigator displays a banner indicating the IP upgrade status. Click **Launch IP Upgrade Tool** or **Project ➤ Upgrade IP Components** to upgrade outdated IP cores.

Figure 32. IP Upgrade Alert in Project Navigator



Icons in the **Upgrade IP Components** dialog box indicate when IP upgrade is required, optional, or unsupported for an IP variation in the project. Upgrade IP variations that require upgrade before compilation in the current version of the Intel Quartus Prime software.

Note: Upgrading IP cores may append a unique identifier to the original IP core entity names, without similarly modifying the IP instance name. There is no requirement to update these entity references in any supporting Intel Quartus Prime file, such as the Intel Quartus Prime Settings File (`.qsf`), Synopsys* Design Constraints File (`.sdc`), or Signal Tap File (`.stp`), if these files contain instance names. The Intel Quartus Prime software reads only the instance name and ignores the entity name in paths that specify both names. Use only instance names in assignments.

Table 11. IP Core Upgrade Status

IP Core Status	Description
IP Upgraded 	Indicates that your IP variation uses the latest version of the Intel FPGA IP core.

continued...

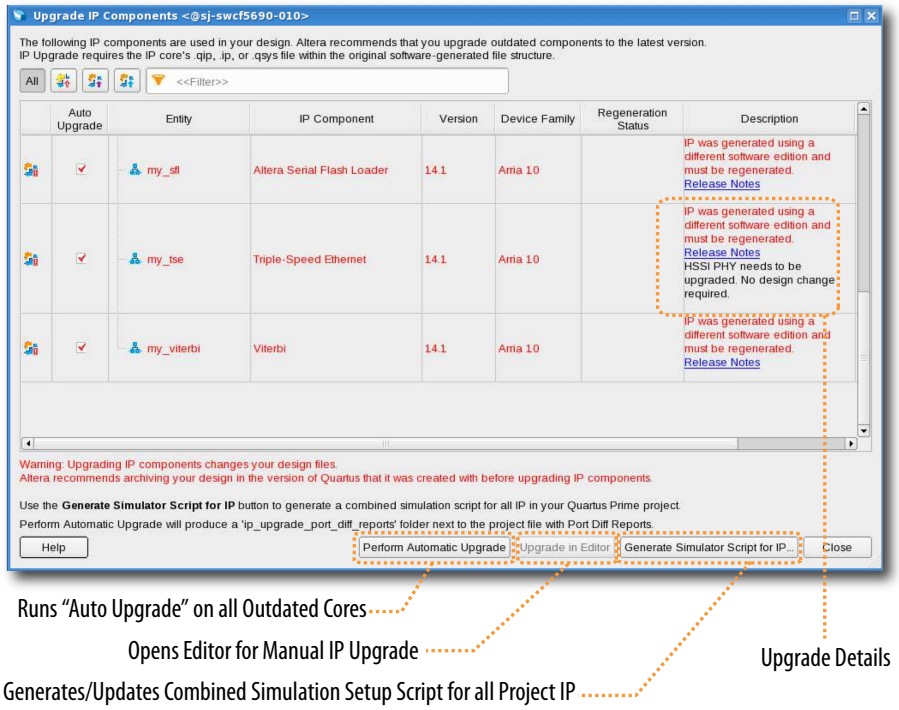
IP Core Status	Description
IP Component Outdated 	Indicates that your IP variation uses an outdated version of the IP core.
IP Upgrade Optional 	Indicates that upgrade is optional for this IP variation in the current version of the Intel Quartus Prime software. You can upgrade this IP variation to take advantage of the latest development of this IP core. Alternatively, you can retain previous IP core characteristics by declining to upgrade. Refer to the Description for details about IP core version differences. If you do not upgrade the IP, the IP variation synthesis and simulation files are unchanged and you cannot modify parameters until upgrading.
IP Upgrade Required 	Indicates that you must upgrade the IP variation before compiling in the current version of the Intel Quartus Prime software. Refer to the Description for details about IP core version differences.
IP Upgrade Unsupported 	Indicates that upgrade of the IP variation is not supported in the current version of the Intel Quartus Prime software due to incompatibility with the current version of the Intel Quartus Prime software. The Intel Quartus Prime software prompts you to replace the unsupported IP core with a supported equivalent IP core from the IP Catalog. Refer to the Description for details about IP core version differences and links to Release Notes.
IP End of Life 	Indicates that Intel designates the IP core as end-of-life status. You may or may not be able to edit the IP core in the parameter editor. Support for this IP core discontinues in future releases of the Intel Quartus Prime software.
IP Upgrade Mismatch Warning 	Provides warning of non-critical IP core differences in migrating IP to another device family.
IP has incompatible subcores 	Indicates that the current version of the Intel Quartus Prime software does not support compilation of your IP variation, because the IP has incompatible subcores
Compilation of IP Not Supported 	Indicates that the current version of the Intel Quartus Prime software does not support compilation of your IP variation. This can occur if another edition of the Intel Quartus Prime software generated this IP. Replace this IP component with a compatible component in the current edition.

Follow these steps to upgrade IP cores:

1. In the latest version of the Intel Quartus Prime software, open the Intel Quartus Prime project containing an outdated IP core variation. The **Upgrade IP Components** dialog box automatically displays the status of IP cores in your project, along with instructions for upgrading each core. To access this dialog box manually, click **Project ➤ Upgrade IP Components**.

2. To upgrade one or more IP cores that support automatic upgrade, ensure that you turn on the **Auto Upgrade** option for the IP cores, and click . The **Status** and **Version** columns update when upgrade is complete. Example designs that any Intel FPGA IP core provides regenerate automatically whenever you upgrade an IP core.
3. To manually upgrade an individual IP core, select the IP core and click **Upgrade in Editor** (or simply double-click the IP core name). The parameter editor opens, allowing you to adjust parameters and regenerate the latest version of the IP core.

Figure 33. Upgrading IP Cores



Note: Intel FPGA IP cores older than Intel Quartus Prime software version 12.0 do not support upgrade. Intel verifies that the current version of the Intel Quartus Prime software compiles the previous two versions of each IP core. The *Intel FPGA IP Core Release Notes* reports any verification exceptions for Intel FPGA IP cores. Intel does not verify compilation for IP cores older than the previous two releases.

Related Information

Intel FPGA IP Release Notes

4.4.1. Upgrading IP Cores at Command-Line

Optionally, upgrade an Intel FPGA IP core at the command-line, rather than using the GUI. IP cores that do not support automatic upgrade do not support command-line upgrade.

- To upgrade a single IP core at the command-line, type the following command:

```
quartus_sh –ip_upgrade –variation_files <my_ip>.<qsys,.v, .vhd> \
     <quartus_project>

Example:
quartus_sh -ip_upgrade -variation_files mega/pll25.qsys hps_testx
```

- To simultaneously upgrade multiple IP cores at the command-line, type the following command:

```
quartus_sh –ip_upgrade –variation_files “<my_ip1>.<qsys,.v, .vhd>> \
     ;  <my_ip_filepath/my_ip2>.<hdl>”  <quartus_project>

Example:
quartus_sh -ip_upgrade -variation_files "mega/pll_tx2.qsys;mega/
pll3.qsys" hps_testx
```

4.4.2. Migrating IP Cores to a Different Device

Migrate an Intel FPGA IP variation when you want to target a different (often newer) device. Most Intel FPGA IP cores support automatic migration. Some IP cores require manual IP regeneration for migration. A few IP cores do not support device migration, requiring you to replace them in the project. The **Upgrade IP Components** dialog box identifies the migration support level for each IP core in the design.

1. To display the IP cores that require migration, click **Project ➤ Upgrade IP Components**. The **Description** field provides migration instructions and version differences.
2. To migrate one or more IP cores that support automatic upgrade, ensure that the **Auto Upgrade** option is turned on for the IP cores, and click **Perform Automatic Upgrade**. The **Status** and **Version** columns update when upgrade is complete.
3. To migrate an IP core that does not support automatic upgrade, double-click the IP core name, and click **OK**. The parameter editor appears. If the parameter editor specifies a **Currently selected device family**, turn off **Match project/default**, and then select the new target device family.
4. Click **Generate HDL**, and confirm the **Synthesis** and **Simulation** file options. Verilog HDL is the default output file format. If you specify VHDL as the output format, select **VHDL** to retain the original output format.
5. Click **Finish** to complete migration of the IP core. Click **OK** if the software prompts you to overwrite IP core files. The **Device Family** column displays the new target device name when migration is complete.
6. To ensure correctness, review the latest parameters in the parameter editor or generated HDL.

   Note: IP migration may change ports, parameters, or functionality of the IP variation. These changes may require you to modify your design or to re-parameterize your IP variant. During migration, the IP variation's HDL generates into a library that is different from the original output location of the IP core. Update any assignments that reference outdated locations. If a symbol in a supporting Block Design File schematic represents your upgraded IP core, replace the symbol with the newly generated *<my_ip>*`.bsf`. Migration of some IP cores requires installed support for the original and migration device families.

Related Information

Intel FPGA IP Release Notes

4.4.3. Troubleshooting IP or Platform Designer System Upgrade

The **Upgrade IP Components** dialog box reports the version and status of each IP core and Platform Designer system following upgrade or migration.

If any upgrade or migration fails, the **Upgrade IP Components** dialog box provides information to help you resolve any errors.

Note: Do not use spaces in IP variation names or paths.

During automatic or manual upgrade, the Messages window dynamically displays upgrade information for each IP core or Platform Designer system. Use the following information to resolve upgrade errors:

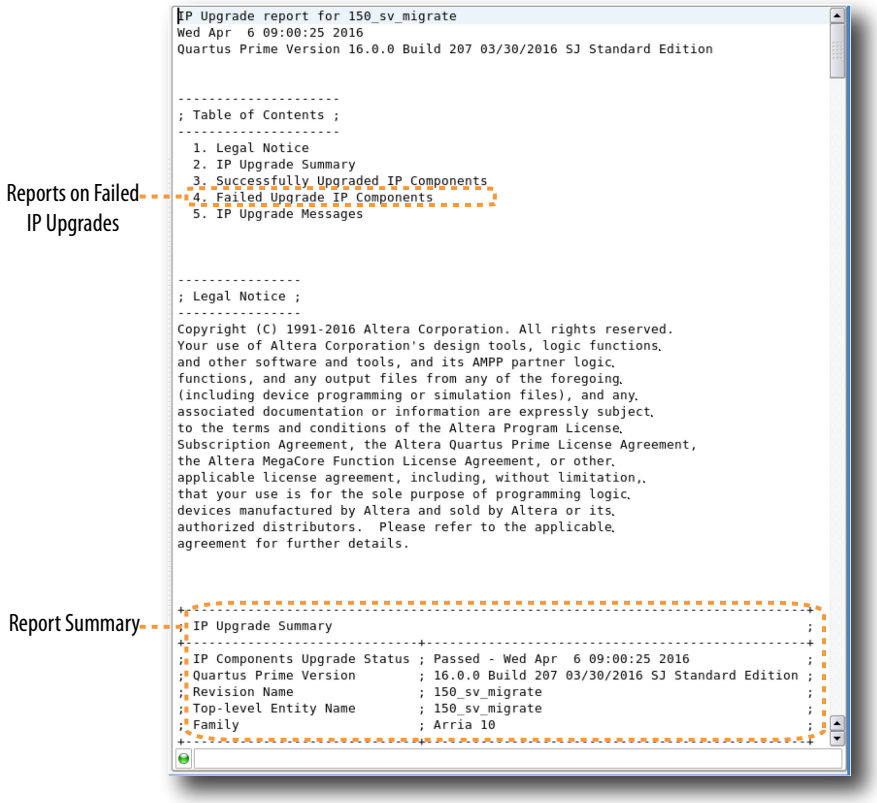
Table 12. IP Upgrade Error Information

Upgrade IP Components Field	Description
Status	Displays the "Success" or "Failed" status of each upgrade or migration. Click the status of any upgrade that fails to open the **IP Upgrade Report**.
Version	Dynamically updates the version number when upgrade is successful. The text is red when the IP requires upgrade.
Device Family	Dynamically updates to the new device family when migration is successful. The text is red when the IP core requires upgrade.
Auto Upgrade	Runs automatic upgrade on all IP cores that support auto upgrade. Also, automatically generates a <*Project Directory*>/ip_upgrade_port_diff_report report for IP cores or Platform Designer systems that fail upgrade. Review these reports to determine any port differences between the current and previous IP core version.

Use the following techniques to resolve errors if your IP core or Platform Designer system "Failed" to upgrade versions or migrate to another device. Review and implement the instructions in the **Description** field, including one or more of the following:

- If the current version of the software does not support the IP variant, right-click the component and click **Remove IP Component from Project**. Replace this IP core or Platform Designer system with the one supported in the current version of the software.
- If the current target device does not support the IP variant, select a supported device family for the project, or replace the IP variant with a suitable replacement that supports your target device.
- If an upgrade or migration fails, click **Failed** in the **Status** field to display and review details of the **IP Upgrade Report**. Click the **Release Notes** link for the latest known issues about the IP core. Use this information to determine the nature of the upgrade or migration failure and make corrections before upgrade.
- Run **Auto Upgrade** to automatically generate an **IP Ports Diff** report for each IP core or Platform Designer system that fails upgrade. Review the reports to determine any port differences between the current and previous IP core version. Click **Upgrade in Editor** to make specific port changes and regenerate your IP core or Platform Designer system.
- If your IP core or Platform Designer system does not support **Auto Upgrade**, click **Upgrade in Editor** to resolve errors and regenerate the component in the parameter editor.

Figure 34. IP Upgrade Report



4.5. Simulating Intel FPGA IP Cores

The Intel Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. Use the functional simulation model and any testbench or example design for simulation. IP generation output may also include scripts to compile and run any testbench. The scripts list all models or libraries you require to simulate your IP core.

The Intel Quartus Prime software provides integration with many simulators and supports multiple simulation flows, including your own scripted and custom simulation flows. Whichever flow you choose, IP core simulation involves the following steps:

1. Generate simulation model, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

4.5.1. Generating IP Simulation Files

The Intel Quartus Prime software optionally generates the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts when you generate an IP core. To control the generation of IP simulation files:

- To specify your supported simulator and options for IP simulation file generation, click **Assignment ➤ Settings ➤ EDA Tool Settings ➤ Simulation**.
- To parameterize a new IP variation, enable generation of simulation files, and generate the IP core synthesis and simulation files, click **Tools ➤ IP Catalog**.
- To edit parameters and regenerate synthesis or simulation files for an existing IP core variation, click **View ➤ Project Navigator ➤ IP Components**.
- To edit parameters and regenerate synthesis or simulation files for an existing IP core variation, click **View ➤ Utility Windows ➤ Project Navigator ➤ IP Components**.

Table 13. Intel FPGA IP Simulation Files

File Type	Description	File Name
Simulator setup scripts	Vendor-specific scripts to compile, elaborate, and simulate Intel FPGA IP models and simulation model library files.	<*my_dir*>/aldec/riviera_setup.tcl <*my_dir*>/cadence/ncsim__setup.sh <*my_dir*>/mentor/msim_setup.tcl <*my_dir*>/synopsys/vcs/vcs_setup.sh <*my_dir*>/synopsys/vcsmx/vcsmx_setup.sh

continued...

File Type	Description	File Name
	Note: For Intel Arria 10 designs, you can use the Intel Quartus Prime software to automatically create a combined simulator setup script. Refer to *Scripting IP Simulation* in the *Introduction to Intel FPGA IP Cores* for more information.	
Simulation IP File (Intel Quartus Prime Standard Edition)	Contains IP core simulation library mapping information. To use NativeLink, add the `.qip` and `.sip` files generated for IP to your project.	`<design name>.sip`
IP functional simulation models (Intel Quartus Prime Standard Edition)	IP functional simulation models are cycle-accurate VHDL or Verilog HDL models a that the Intel Quartus Prime software generates for some Intel FPGA IP cores. IP functional simulation models support fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.	`<my_ip>.vho` `<my_ip>.vo`
IEEE encrypted models (Intel Quartus Prime Standard Edition)	Intel provides Arria V, Cyclone V, Stratix V, and newer simulation model libraries and IP simulation models in Verilog HDL and IEEE-encrypted Verilog HDL. Your simulator's co-simulation capabilities support VHDL simulation of these models. IEEE encrypted Verilog HDL models are significantly faster than IP functional simulation models. The Intel Quartus Prime Pro Edition software does not support these models.	`<my_ip>.v`

Note: Intel FPGA IP cores support a variety of cycle-accurate simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some IP cores, generation only produces the plain text RTL model, and you can simulate that model. Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

4.5.2. Using NativeLink Simulation (Intel Quartus Prime Standard Edition)

The NativeLink feature integrates your EDA simulator with the Intel Quartus Prime Standard Edition software by automating the following:

- Generation of simulator-specific files and simulation scripts.
- Compilation of simulation libraries.
- Launches your simulator automatically following Intel Quartus Prime Analysis & Elaboration, Analysis & Synthesis, or after a full compilation.

Note: The Intel Quartus Prime Pro Edition does not support NativeLink simulation. If you use NativeLink for Intel Arria 10 devices in the Intel Quartus Prime Standard Edition, you must add the `.qsys` file generated for the IP or Platform Designer (Standard) system to your Intel Quartus Prime project. If you use NativeLink for any other supported device family, you must add the `.qip` and `.sip` files to your project.

4.5.2.1. Setting Up NativeLink Simulation (Intel Quartus Prime Standard Edition)

Before running NativeLink simulation, specify settings for your simulator in the Intel Quartus Prime software.

To specify NativeLink settings in the Intel Quartus Prime Standard Edition software, follow these steps:

1. Open an Intel Quartus Prime Standard Edition project.
2. Click **Tools > Options** and specify the location of your simulator executable file.

Table 14. Execution Paths for EDA Simulators

Simulator	Path
Mentor Graphics ModelSim-AE	*<drive letter>*:*<simulator install path>*\win32aloem (Windows) /*<simulator install path>*/bin (Linux)
Mentor Graphics ModelSim Mentor Graphics QuestaSim	*<drive letter>*:*<simulator install path>*\win32 (Windows) *<simulator install path>*/bin (Linux)
Synopsys VCS/VCS MX	*<simulator install path>*/bin (Linux)
Cadence Incisive Enterprise	*<simulator install path>*/tools/bin (Linux)
Aldec Active-HDL Aldec Riviera-PRO	*<drive letter>*:*<simulator install path>*\bin (Windows) *<simulator install path>*/bin (Linux)

3. Click **Assignments ➤ Settings** and specify options on the **Simulation** page and the **More NativeLink Settings** dialog box. Specify default options for simulation library compilation, netlist and tool command script generation, and for launching RTL or gate-level simulation automatically following compilation.
4. If your design includes a testbench, turn on **Compile test bench**. Click **Test Benches** to specify options for each testbench. Alternatively, turn on **Use script to compile testbench** and specify the script file.
5. To use a script to setup a simulation, turn on **Use script to setup simulation**.

4.5.2.2. Generating IP Functional Simulation Models (Intel Quartus Prime Standard Edition)

Intel provides IP functional simulation models for some Intel FPGA IP supporting 40nm FPGA devices.

To generate IP functional simulation models:

1. Turn on the **Generate Simulation Model** option when parameterizing the IP core.
2. When you simulate your design, compile only the `.vo` or `.vho` for these IP cores in your simulator. Do not compile the corresponding HDL file. The encrypted HDL file supports synthesis by only the Intel Quartus Prime software.

   Note:
   - Intel FPGA IP cores that do not require IP functional simulation models for simulation, do not provide the **Generate Simulation Model** option in the IP core parameter editor.
   - Many recently released Intel FPGA IP cores support RTL simulation using IEEE Verilog HDL encryption. IEEE encrypted models are significantly faster than IP functional simulation models. Simulate the models in both Verilog HDL and VHDL designs.

Related Information

AN 343: Intel FPGA IP Evaluation Mode of AMPP IP

4.6. Synthesizing IP Cores in Other EDA Tools

Optionally, use another supported EDA tool to synthesize a design that includes Intel FPGA IP cores. When you generate the IP core synthesis files for use with third-party EDA synthesis tools, you can create an area and timing estimation netlist. To enable generation, turn on **Create timing and resource estimates for third-party EDA synthesis tools** when customizing your IP variation.

The area and timing estimation netlist describes the IP core connectivity and architecture, but does not include details about the true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to achieve timing-driven optimizations and improve the quality of results.

The Intel Quartus Prime software generates the <*variant name*>_syn.v netlist file in Verilog HDL format, regardless of the output file format you specify. If you use this netlist for synthesis, you must include the IP core wrapper file <*variant name*>.v or <*variant name*> .vhd in your Intel Quartus Prime project.

4.7. Instantiating IP Cores in HDL

Instantiate an IP core directly in your HDL code by calling the IP core name and declaring the IP core's parameters. This approach is similar to instantiating any other module, component, or subdesign. When instantiating an IP core in VHDL, you must include the associated libraries.

4.7.1. Example Top-Level Verilog HDL Module

Verilog HDL ALTFP_MULT in Top-Level Module with One Input Connected to Multiplexer.

```
module MF_top (a, b, sel, datab, clock, result);
       input [31:0] a, b, datab;
       input clock, sel;
       output [31:0] result;
       wire [31:0] wire_dataa;

       assign wire_dataa = (sel)? a : b;
       altfp_mult inst1
(.dataa(wire_dataa), .datab(datab), .clock(clock), .result(result));

       defparam
              inst1.pipeline = 11,
              inst1.width_exp = 8,
              inst1.width_man = 23,
              inst1.exception_handling = "no";
endmodule
```

4.7.2. Example Top-Level VHDL Module

VHDL ALTFP_MULT in Top-Level Module with One Input Connected to Multiplexer.

```
library ieee;
use ieee.std_logic_1164.all;
library altera_mf;
use altera_mf.altera_mf_components.all;

entity MF_top is
        port (clock, sel  : in  std_logic;
              a, b, datab : in  std_logic_vector(31 downto 0);
```

```
                result      : out std_logic_vector(31 downto 0));
end entity;

architecture arch_MF_top of MF_top is
signal wire_dataa : std_logic_vector(31 downto 0);
begin

wire_dataa <= a when (sel = '1') else b;

inst1 : altfp_mult
        generic map    (
                   pipeline => 11,
                   width_exp => 8,
                   width_man => 23,
                   exception_handling => "no")
        port map (
                   dataa => wire_dataa,
                   datab => datab,
                   clock => clock,
                   result => result);
end arch_MF_top;
```

4.8. Introduction to Intel FPGA IP Cores Revision History

This chapter has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	• Added statement that the Intel Quartus Prime software installer does not support spaces in the installation path. • Added "Intel FPGA IP Best Practices" topic. • Divided "Introduction to Intel FPGA IP Cores" into separate chapter of *Getting Started User Guide*.
2018.05.07	18.0.0	• Updated screenshots of IP Catalog and Parameter Editor for latest IP names. • Added note about Generate Combined Simulator Setup Scripts command limitations. • Added information about generation of simulation files for Xcelium*
2017.11.06	17.1.0	• Revised product branding for Intel standards. • Revised topics on Intel FPGA IP Evaluation Mode (formerly OpenCore).
2017.05.08	17.0.0	• Added note that IP core encryption is supported only in Intel Quartus Prime Pro Edition. • Revised product branding for Intel standards.
2016.10.31	16.1.0	• Removed references to `.qsys` file creation during Intel Quartus Prime Pro Edition stand-alone IP generation. • Added references to `.ip` file creation during Intel Quartus Prime Pro Edition stand-alone IP generation. • Updated IP Core Generation Output files list and diagram. • Indicated distinctions between Intel Quartus Prime Pro Edition and Intel Quartus Prime Standard Edition features. • Added Support for IP Core Encryption topic.

5. Migrating to Intel Quartus Prime Pro Edition

The Intel Quartus Prime Pro Edition software supports migration of Intel Quartus Prime Standard Edition, Quartus Prime Lite Edition, and Quartus II software projects.

Note: The migration steps for Quartus Prime Lite Edition, Intel Quartus Prime Standard Edition, and the Quartus II software are identical. For brevity, this section refers to these design tools collectively as "other Quartus software products."

Migrating to Intel Quartus Prime Pro Edition requires the following changes to other Quartus software product projects:

1. Upgrade project assignments and constraints with equivalent Intel Quartus Prime Pro Edition assignments.
2. Upgrade all Intel FPGA IP core variations and Platform Designer (Standard) systems in your project.
3. Upgrade design RTL to standards-compliant VHDL, Verilog HDL, or SystemVerilog.

This document describes each migration step in detail.

5.1. Keep Pro Edition Project Files Separate

The Intel Quartus Prime Pro Edition software does not support project or constraint files from other Quartus software products. Do not place project files from other Quartus software products in the same directory as Intel Quartus Prime Pro Edition project files. In general, use Intel Quartus Prime Pro Edition project files and directories only for Intel Quartus Prime Pro Edition projects, and use other Quartus software product files only with those software tools.

Intel Quartus Prime Pro Edition projects do not support compilation in other Quartus software products, and vice versa. The Intel Quartus Prime Pro Edition software generates an error if the Compiler detects other Quartus software product's features in project files.

Before migrating other Quartus software product projects, click **Project ➤ Archive Project** to save a copy of your original project before making modifications for migration.

5.2. Upgrade Project Assignments and Constraints

Intel Quartus Prime Pro Edition software introduces changes to handling of project assignments and constraints that the Quartus Settings File (`.qsf`) stores. Upgrade other Quartus software product project assignments and constraints for migration to the Intel Quartus Prime Pro Edition software. Upgrade other Quartus software product assignments with **Assignments ➤ Assignment Editor**, by editing the `.qsf` file directly, or by using a Tcl script.

The following sections detail each type project assignment upgrade that migration requires.

Related Information

- Modify Entity Name Assignments on page 74
- Resolve Timing Constraint Entity Names on page 74
- Verify Generated Node Name Assignments on page 75
- Replace Logic Lock (Standard) Regions on page 75
- Modify Signal Tap Logic Analyzer Files on page 77
- Remove Unsupported Feature Assignments on page 78

5.2.1. Modify Entity Name Assignments

Intel Quartus Prime Pro Edition software supports assignments that include instance names *without* a corresponding entity name.

- "`a_entity:a|b_entity:b|c_entity:c`" (includes deprecated entity names)
- "`a|b|c`" (omits deprecated entity names)

While the current version of the Intel Quartus Prime Pro Edition software still *accepts* entity names in the `.qsf`, the Compiler *ignores* the entity name. The Compiler generates a warning message upon detection of an entity names in the `.qsf`. Whenever possible, you should remove entity names from assignments, and discontinue reliance on entity-based assignments. Future versions of the Intel Quartus Prime Pro Edition software may eliminate all support for entity-based assignments.

5.2.2. Resolve Timing Constraint Entity Names

The Intel Quartus Prime Pro Edition Timing Analyzer honors entity names in Synopsys Design Constraints (`.sdc`) files.

Use `.sdc` files from other Quartus software products without modification. However, any scripts that include custom processing of names that the `.sdc` command returns, such as `get_registers` may require modification. Your scripts must reflect that returned strings do not include entity names.

The `.sdc` commands respect wildcard patterns containing entity names. Review the Timing Analyzer reports to verify application of all constraints. The following example illustrates differences between functioning and non-functioning `.sdc` scripts:

```
# Apply a constraint to all registers named "acc" in the entity "counter".
# This constraint functions in both SE and PE, because the SDC
# command always understands wildcard patterns with entity names in them
set_false_path –to [get_registers  “counter:*|*acc”]

# This does the same thing, but first it converts all register names to
# strings, which includes entity names by default in the SE
# but excludes them by default in the PE. The regexp will therefore
# fail in PE by default.
#
# This script would also fail in the SE, and earlier
# versions of Quartus II, if entity name display had been disabled
# in the QSF.
set all_reg_strs [query_collection –list –all [get_registers *]]
foreach keeper $all_reg_strs {
   if {[regexp {counter:*|:*acc} $keeper]} {
```

```
      set_false_path –to $keeper
   }
}
```

Removal of the entity name processing from `.sdc` files may not be possible due to complex processing involving node names. Use standard `.sdc` whenever possible to replace such processing. Alternatively, add the following code to the top and bottom of your script to temporarily re-enable entity name display in the `.sdc` file:

```
# This script requires that entity names be included
# due to custom name processing
set old_mode [set_project_mode -get_mode_value always_show_entity_name]
set_project_mode -always_show_entity_name on

<... the rest of your script goes here ...>

# Restore the project mode
set_project_mode -always_show_entity_name $old_mode
```

5.2.3. Verify Generated Node Name Assignments

Intel Quartus Prime synthesis generates and automatically names internal design nodes during processing. The Intel Quartus Prime Pro Edition uses different conventions than other Quartus software products to generate node names during synthesis. When you synthesize your other Quartus software product project in Intel Quartus Prime Pro Edition, the synthesis-generated node names may change. If any scripts or constraints depend on the synthesis-generated node names, update the scripts or constraints to match the Intel Quartus Prime Pro Edition synthesis node names.

Avoid dependence on synthesis-generated names due to frequent changes in name generation. In addition, verify the names of duplicated registers and PLL clock outputs to ensure compatibility with any script or constraint.

5.2.4. Replace Logic Lock (Standard) Regions

Intel Quartus Prime Pro Edition software introduces more simplified and flexible Logic Lock constraints, compared with previous Logic Lock regions. You must replace all Logic Lock (Standard) assignments with compatible Logic Lock assignments for migration.

To convert Logic Lock (Standard) regions to Logic Lock regions:

1. Edit the `.qsf` to delete or comment out all of the following Logic Lock assignments:

```
set_global_assignment -name LL_ENABLED*
set_global_assignment -name LL_AUTO_SIZE*
set_global_assignment -name LL_STATE FLOATING*
set_global_assignment -name LL_RESERVED*
set_global_assignment -name LL_CORE_ONLY*
set_global_assignment -name LL_SECURITY_ROUTING_INTERFACE*
set_global_assignment -name LL_IGNORE_IO_BANK_SECURITY_CONSTRAINT*
set_global_assignment -name LL_PR_REGION*
set_global_assignment -name LL_ROUTING_REGION_EXPANSION_SIZE*
set_global_assignment -name LL_WIDTH*
set_global_assignment -name LL_HEIGHT
set_global_assignment -name LL_ORIGIN
set_instance_assignment -name LL_MEMBER_OF
```

2. Edit the `.qsf` or click **Tools ➤ Chip Planner** to define new Logic Lock regions. Logic Lock constraint syntax is simplified, for example:

```
set_instance_assignment -name PLACE_REGION "1 1 20 20" -to fifo1
set_instance_assignment -name RESERVE_PLACE_REGION OFF -to fifo1
set_instance_assignment -name CORE_ONLY_PLACE_REGION OFF -to fifo1
```

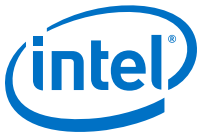
Compilation fails if synthesis finds other Quartus software product's Logic Lock assignments in an Intel Quartus Prime Pro Edition project. The following table compares other Quartus software product region constraint support with the Intel Quartus Prime Pro Edition software.

Table 15. Region Constraints Per Edition

Constraint Type	Logic Lock (Standard) Region Support Other Quartus Software Products	Logic Lock Region Support Intel Quartus Prime Pro Edition
Fixed rectangular, nonrectangular or non-contiguous regions	Full support.	Full support.
Chip Planner entry	Full support.	Full support.
Periphery element assignments	Supported in some instances.	Full support. Use “core-only” regions to exclude the periphery.
Nested (“hierarchical”) regions	Supported but separate hierarchy from the user instance tree.	Supported in same hierarchy as user instance tree.
Reserved regions	Limited support for nested or nonrectangular reserved regions. Reserved regions typically cannot cross I/O columns; use non-contiguous regions instead.	Full support for nested and nonrectangular regions. Reserved regions can cross I/O columns without affecting periphery logic if the regions are "core-only".
Routing regions	Limited support via “routing expansion.” No support with hierarchical regions.	Full support (including future support for hierarchical regions).
Floating or autosized regions	Full support.	No support.
Region names	Regions have names.	Regions are identified by the instance name of the constrained logic.
Multiple instances in the same region	Full support.	Support for non-reserved regions. Create one region per instance, and then specify the same definition for multiple instances to assign to the same area. Not supported for reserved regions.
Member exclusion	Full support.	No support for arbitrary logic. Use a core-only region to exclude periphery elements. Use non-rectangular regions to include more RAM or DSP columns as needed.

5.2.4.1. Logic Lock Region Assignment Examples

These examples show the syntax of Logic Lock region assignments in the `.qsf` file. Optionally, enter these assignments in the Assignment Editor, the Logic Lock Regions Window, or the Chip Planner.



Example 1. Assign Rectangular Logic Lock Region

Assigns a rectangular Logic Lock region to a lower right corner location of (10,10), and an upper right corner of (20,20) inclusive.

```
set_instance_assignment –name PLACE_REGION –to a|b|c "X10 Y10 X20 Y20"
```

Example 2. Assign Non-Rectangular Logic Lock Region

Assigns instance with full hierarchical path "x|y|z" to non-rectangular L-shaped Logic Lock region. The software treats each set of four numbers as a new box.

```
set_instance_assignment –name PLACE_REGION –to x|y|z "X10 Y10 X20 Y50; X20
Y10 X50 Y20"
```

Example 3. Assign Subordinate Logic Lock Instances

By default, the Intel Quartus Prime software constrains every child instance to the Logic Lock region of its parent. Any constraint to a child instance intersects with the constraint of its ancestors. For example, in the following example, all logic beneath "a|b|c|d" constrains to box (10,10), (15,15), and not (0,0), (15,15). This result occurs because the child constraint intersects with the parent constraint.

```
set_instance_assignment –name PLACE_REGION –to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment –name PLACE_REGION –to a|b|c|d "X0 Y0 X15 Y15"
```

Example 4. Assign Multiple Logic Lock Instances

By default, a Logic Lock region constraint allows logic from other instances to share the same region. These assignments place instance c and instance g in the same location. This strategy is useful if instance c and instance g are heavily interacting.

```
set_instance_assignment –name PLACE_REGION –to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment –name PLACE_REGION –to e|f|g "X10 Y10 X20 Y20"
```

Example 5. Assigned Reserved Logic Lock Regions

Optionally reserve an entire Logic Lock region for one instance and any of its subordinate instances.

```
set_instance_assignment –name PLACE_REGION –to a|b|c "X10 Y10 X20 Y20"
set_instance_assignment –name RESERVE_PLACE_REGION –to a|b|c ON

# The following assignment causes an error. The logic in e|f|g is not
# legally placeable anywhere:
# set_instance_assignment –name PLACE_REGION –to e|f|g "X10 Y10 X20 Y20"

# The following assignment does *not* cause an error, but is effectively
# constrained to the box (20,10), (30,20), since the (10,10),(20,20) box is
reserved
# for a|b|c
set_instance_assignment –name PLACE_REGION –to e|f|g "X10 Y10 X30 Y20"
```

5.2.5. Modify Signal Tap Logic Analyzer Files

Intel Quartus Prime Pro Edition introduces new methodology for entity names, settings, and assignments. These changes impact the processing of Signal Tap Logic Analyzer Files (.stp).

If you migrate a project that includes `.stp` files generated by other Quartus software products, you must make the following changes to migrate to the Intel Quartus Prime Pro Edition:

1. Remove entity names from `.stp` files. The Signal Tap Logic Analyzer allows without error, but ignores, entity names in `.stp` files. Remove entity names from `.stp` files for migration to Intel Quartus Prime Pro Edition:
   a. Click **View ➤ Utility Windows ➤ Node Finder** to locate and remove appropriate nodes. Use Node Finder options to filter on nodes.
   b. Click **Processing ➤ Start ➤ Start Analysis & Elaboration** to repopulate the database and add valid node names.
2. Remove post-fit nodes. Intel Quartus Prime Pro Edition uses a different post-fit node naming scheme than other Quartus software products.
   a. Remove post-fit tap node names originating from other Quartus software products.
   b. Click **View ➤ Utility Windows ➤ Node Finder** to locate and remove post-fit nodes. Use Node Finder options to filter on nodes.
   c. Click **Processing ➤ Start Compilation** to repopulate the database and add valid post-fit nodes.
3. Run an initial compilation in Intel Quartus Prime Pro Edition from the GUI. The Compiler automatically removes Signal Tap assignments originating other Quartus software products. Alternatively, from the command-line, run `quartus_stp` once on the project to remove outmoded assignments.

   Note: `quartus_stp` introduces no migration impact in the Intel Quartus Prime Pro Edition. Your scripts require no changes to `quartus_stp` for migration.
4. Modify `.sdc` constraints for JTAG. Intel Quartus Prime Pro Edition does not support embedded `.sdc` constraints for JTAG signals. Modify the timing template to suit the design's JTAG driver and board.

5.2.6. Remove References to .qip Files

In Intel Quartus Prime Standard Edition projects, Platform Designer (Standard) (Standard) generates `.qip` files. These files describe the parameterized IP cores to the Compiler, and appear as assignments in the project's `.qsf` file. However, in Intel Quartus Prime Pro Edition projects, the parameterized IP core description occurs in `.ip` files. Moreover, references to `.qip` files in a project's `.qsf` file cause synthesis errors during compilation.

- When migrating a project to Intel Quartus Prime Pro Edition, remove all references to `.qip` files from the `.qsf` file.

5.2.7. Remove Unsupported Feature Assignments

The Intel Quartus Prime Pro Edition software does not support some feature assignments that other Quartus software products support. Remove the following unsupported feature assignments from other Quartus software product `.qsf` files for migration to the Intel Quartus Prime Pro Edition software.

- Incremental Compilation (partitions)—The current version of the Intel Quartus Prime Pro Edition software does not support Intel Quartus Prime Standard Edition incremental compilation. Remove all incremental compilation feature assignments from other Quartus software product `.qsf` files before migration.
- Intel Quartus Prime Standard Edition Physical synthesis assignments. Intel Quartus Prime Pro Edition software does not support Intel Quartus Prime Standard Edition Physical synthesis assignments. Remove any of the following assignments from the `.qsf` file or design RTL (instance assignments) before migration.

```
PHYSICAL_SYNTHESIS_COMBO_LOGIC_FOR_AREA
    PHYSICAL_SYNTHESIS_COMBO_LOGIC
    PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION
    PHYSICAL_SYNTHESIS_REGISTER_RETIMING
    PHYSICAL_SYNTHESIS_ASYNCHRONOUS_SIGNAL_PIPELINING
    PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA
```

5.3. Upgrade IP Cores and Platform Designer (Standard) Systems

Upgrade all IP cores and Platform Designer (Standard) systems in your project for migration to the Intel Quartus Prime Pro Edition software. The Intel Quartus Prime Pro Edition software uses standards-compliant methodology for instantiation and generation of IP cores and Platform Designer systems. Most Intel FPGA IP cores and Platform Designer systems upgrade automatically in the **Upgrade IP Components** dialog box.

Other Quartus software products use a proprietary Verilog configuration scheme within the top level of IP cores and Platform Designer (Standard) systems for synthesis files. The Intel Quartus Prime Pro Edition does not support this scheme. To upgrade all IP cores and Platform Designer (Standard) systems in your project, click **Project ➤ Upgrade IP Components**.(1)

Table 16. IP Core and Platform Designer (Standard) System Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
IP and Platform Designer (Standard) system generation use a proprietary Verilog HDL configuration scheme within the top level of IP cores and Platform Designer (Standard) systems for synthesis files. This proprietary Verilog HDL configuration scheme prevents RTL entities from ambiguous instantiation errors during synthesis. However, these errors may manifest in simulation. Resolving this issue requires writing a Verilog HDL configuration to disambiguate the instantiation, delete the duplicate entity from the project, or rename one of the conflicting entities. Intel Quartus Prime Pro Edition IP strategy resolves these issues.	IP and Platform Designer system generation does not use proprietary Verilog HDL configurations. The compilation library scheme changes in the following ways: • Compiles all variants of an IP core into the same compilation library across the entire project. Intel Quartus Prime Pro Edition identically names IP cores with identical functionality and parameterization to avoid ambiguous entity instantiation errors. For example, the files for every Intel Arria 10 PCI Express* IP core variant compile into the `altera_pcie_a10_hip_151` compilation library. • Simulation and synthesis file sets for IP cores and systems instantiate entities in the same manner. • The generated RTL directory structure now matches the compilation library structure.

Note: For complete information on upgrading IP cores, refer to *Managing Intel Quartus Prime Projects*.

(1) For brevity, this section refers to Intel Quartus Prime Standard Edition, Intel Quartus Prime Lite Edition, and the Quartus II software collectively as "other Quartus software products."

5.4. Upgrade Non-Compliant Design RTL

The Intel Quartus Prime Pro Edition software introduces a new synthesis engine (`quartus_syn` executable).

The `quartus_syn` synthesis enforces stricter industry-standard HDL structures and supports the following enhancements in this release:

- Support for modules with SystemVerilog Interfaces
- Improved support for VHDL2008
- New RAM inference engine infers RAMs from GENERATE statements or array of integers
- Stricter syntax/semantics check for improved compatibility with other EDA tools

Account for these synthesis differences in existing RTL code by ensuring that your design uses standards-compliant VHDL, Verilog HDL, or SystemVerilog. The Compiler generates errors when processing non-compliant RTL. Use the guidelines in this section to modify existing RTL for compatibility with the Intel Quartus Prime Pro Edition synthesis.

Related Information

- Verify Verilog Compilation Unit on page 80
- Update Entity Auto-Discovery on page 81
- Ensure Distinct VHDL Namespace for Each Library on page 82
- Remove Unsupported Parameter Passing on page 82
- Remove Unsized Constant from WYSIWYG Instantiation on page 82
- Remove Non-Standard Pragmas on page 83
- Declare Objects Before Initial Values on page 83
- Confine SystemVerilog Features to SystemVerilog Files on page 83
- Avoid Assignment Mixing in Always Blocks on page 84
- Avoid Unconnected, Non-Existent Ports on page 84
- Avoid Illegal Parameter Ranges on page 84
- Update Verilog HDL and VHDL Type Mapping on page 85

5.4.1. Verify Verilog Compilation Unit

Intel Quartus Prime Pro Edition synthesis uses a different method to define the compilation unit. The Verilog LRM defines the concept of compilation unit as "a collection of one or more Verilog source files compiled together" forming the compilation-unit scope. Items visible only in the compilation-unit scope include macros, global declarations, and default net types. The contents of included files become part of the compilation unit of the parent file. Modules, primitives, programs, interfaces, and packages are visible in all compilation units. Ensure that your RTL accommodates these changes.

Table 17. Verilog Compilation Unit Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
Synthesis in other Quartus software products follows the Multi-file compilation unit (MFCU) method to select compilation unit files. In MFCU, all files compile in the same compilation unit. Global definitions and directives are visible in all files. However, the default net type is reset at the start of each file.	Intel Quartus Prime Pro Edition synthesis follows the Single-file compilation unit (SFCU) method to select compilation unit files. In SFCU, each file is a compilation unit, file order is irrelevant, and the macro is only defined until the end of the file.

Note: You can optionally change the MFCU mode using the following assignment:
`set_global_assignment -name VERILOG_CU_MODE MFCU`

5.4.1.1. Verilog HDL Configuration Instantiation

Intel Quartus Prime Pro Edition synthesis requires instantiation of the Verilog HDL configuration, and not the module. In other Quartus software products, synthesis automatically finds any Verilog HDL configuration relating to a module that you instantiate. The Verilog HDL configuration then instantiates the design.

If your top-level entity is a Verilog HDL configuration, set the Verilog HDL configuration, rather than the module, as the top-level entity.

Table 18. Verilog HDL Configuration Instantiation

Other Quartus Software Products	Intel Quartus Prime Pro Edition
From the Example RTL, synthesis automatically finds the `mid_config` Verilog HDL configuration relating to the instantiated module.	From the Example RTL, synthesis does not find the `mid_config` Verilog HDL configuration. You must instantiate the Verilog HDL configuration directly.

Example RTL:
`config mid_config;`
`design good_lib.mid;`
`instance mid.sub_inst use good_lib.sub;`
`endconfig`
 	
`module test (input a1, output b);`
`mid_config mid_inst ( .a1(a1), .b(b));`
`// in other Quartus products preceding line would have been:`
`//mid mid_inst ( .a1(a1), .b(b));`
`endmodule`
 	
`module mid (input a1, output b);`
`sub sub_inst (.a1(a1), .b(b));`
`endmodule`	

5.4.2. Update Entity Auto-Discovery

All editions of the Intel Quartus Prime and Quartus II software search your project directory for undefined entities. For example, if you instantiate entity "sub" in your design without specifying "sub" as a design file in the Quartus Settings File (`.qsf`), synthesis searches for `sub.v`, `sub.vhd`, and so on. However, Intel Quartus Prime Pro Edition performs auto-discovery at a different stage in the flow. Ensure that your RTL code accommodates these auto-discovery changes.

Table 19. Entity Auto-Discovery Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
Always automatically searches your project directory and search path for undefined entities.	Always automatically searches your project directory and search path for undefined entities. Intel Quartus Prime Pro Edition synthesis performs auto-discovery earlier in the flow than other Quartus software products. This results in discovery of more syntax errors. Optionally disable auto-discovery with the following `.qsf` assignment: `set_global_assignment -name AUTO_DISCOVER_AND_SORT OFF`

5.4.3. Ensure Distinct VHDL Namespace for Each Library

Intel Quartus Prime Pro Edition synthesis requires that VHDL namespaces are distinct for each library. The stricter library binding requirement complies with VHDL language specifications and results in deterministic behavior. This benefits team-based projects by avoiding unintentional name collisions. Confirm that your RTL respects this change.

Table 20. VHDL Namespace Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
For the Example RTL, the analyzer searches all libraries in an unspecified order until the analyzer finds package `utilities_pack` and uses items from that package. If another library, for example `projectLib` also contains `utilities_pack`, the analyzer may use this library instead of `myLib.utilites_pack` if found before the analyzer searches `myLib`.	For the Example RTL, the analyzer uses the specific `utilities_pack` in `myLib`. If `utilities_pack` does not exist in library `myLib`, the analyzer generates an error.
Example RTL: `library myLib; use myLib.utilities_pack.all;`	

5.4.4. Remove Unsupported Parameter Passing

Intel Quartus Prime Pro Edition synthesis does not support parameter passing using `set_parameter` in the `.qsf`. Synthesis in other Quartus software products supports passing parameters with this method. Except for the top-level of the design where permitted, ensure that your RTL does not depend on this type of parameter passing.

Table 21. SystemVerilog Feature Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
From the Example RTL, synthesis overwrites the value of parameter SIZE in the instance of `my_ram` instantiated from entity `mid-level`.	From the Example RTL, synthesis generates a syntax error for detection of parameter passing assignments in the `.qsf`. Specify parameters in the RTL. The following example shows the supported top-level parameter passing format. This example applies only to the top-level and sets a value of 4 to parameter `N`: `set_parameter -name N 4`
Example RTL: `set_parameter –entity mid_level –to my_ram –name SIZE 16`	

5.4.5. Remove Unsized Constant from WYSIWYG Instantiation

Intel Quartus Prime Pro Edition synthesis does not allow use of an unsized constant for WYSIWYG instantiation. Synthesis in other Quartus software products allows use of SystemVerilog (`.sv`) unsized constants when instantiating a WYSIWYG in a `.v` file.

Intel Quartus Prime Pro Edition synthesis allows use of unsized constants in `.sv` files for uses other than WYSIWYG instantiation. Ensure that your RTL code does not use unsized constants for WYSIWYG instantiation. For example, specify a sized literal, such as 2'b11, rather than '1.

5.4.6. Remove Non-Standard Pragmas

Intel Quartus Prime Pro Edition synthesis does not support the `vhdl(verilog)_input_version` pragma or the `library` pragma. Synthesis in other Quartus software products supports these pragmas. Remove any use of the pragmas from RTL for Intel Quartus Prime Pro Edition migration. Use the following guidelines to implement the pragma functionality in Intel Quartus Prime Pro Edition:

- `vhdl(verilog)_input_version` Pragma—allows change to the input version in the middle of an input file. For example, to change VHDL 1993 to VHDL 2008. For Intel Quartus Prime Pro Edition migration, specify the input version for each file in the `.qsf`.
- `library` Pragma—allows changes to the VHDL library into which files compile. For Intel Quartus Prime Pro Edition migration, specify the compilation library in the `.qsf`.

5.4.7. Declare Objects Before Initial Values

Intel Quartus Prime Pro Edition synthesis requires declaration of objects before initial value. Ensure that your RTL declares objects before initial value. Other Quartus software products allow declaration of initial value prior to declaration of the object.

Table 22. Object Declaration Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
From the Example RTL, synthesis initializes the output `p_prog_io1` with the value of `p_progio1_reg`, even though the register declaration occurs in Line 2.	From the Example RTL, synthesis generates a syntax error when you specify initial values before declaring the register.

Example RTL:
`1 output p_prog_io1 = p_prog_io1_reg;`
`2 reg p_prog_io1_reg;`	

5.4.8. Confine SystemVerilog Features to SystemVerilog Files

Intel Quartus Prime Pro Edition synthesis does not allow SystemVerilog features in Verilog HDL files. Other Quartus software products allow use of a subset of SystemVerilog (`.sv`) features in Verilog HDL (`.v`) design files. To avoid syntax errors in Intel Quartus Prime Pro Edition, allow only SystemVerilog features in Verilog HDL files.

To use SystemVerilog features in your existing Verilog HDL files, rename your Verilog HDL (`.v`) files as SystemVerilog (`.sv`) files. Alternatively, you can set the file type in the .qsf, as shown in the following example:

```
set_global_assignment -name SYSTEMVERILOG_FILE <file>.v
```

Table 23. SystemVerilog Feature Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
From the Example RTL, synthesis interprets `$clog2` in a `.v` file, even though the Verilog LRM does not define the `$clog2` feature. Other Quartus software products allow other SystemVerilog features in `.v` files.	From the Example RTL, synthesis generates a syntax error for detection of any non-Verilog HDL construct in `.v` files. Intel Quartus Prime Pro Edition synthesis honors SystemVerilog features only in `.sv` files.

Example RTL:
`localparam num_mem_locations = 1050;`
`wire mem_addr [$clog2(num_mem_locations)-1 : 0];`	

5.4.9. Avoid Assignment Mixing in Always Blocks

Intel Quartus Prime Pro Edition synthesis does not allow mixed use of blocking and non-blocking assignments within ALWAYS blocks. Other Quartus software products allow mixed use of blocking and non-blocking assignments within ALWAYS blocks. To avoid syntax errors, ensure that ALWAYS block assignments are of the same type for Intel Quartus Prime Pro Edition migration.

Table 24. ALWAYS Block Assignment Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
Synthesis honors the mixed blocking and non-blocking assignments, although the Verilog Language Specification no longer supports this construct.	Synthesis generates a syntax error for detection of mixed blocking and non-blocking assignments within an ALWAYS block.

5.4.10. Avoid Unconnected, Non-Existent Ports

Intel Quartus Prime Pro Edition synthesis requires that a port exists in the module prior to instantiation and naming. Other Quartus software products allow you to instantiate and name an unconnected port that does not exist in the module. Modify your RTL to match this requirement.

To avoid syntax errors, remove all unconnected and non-existent ports for Intel Quartus Prime Pro Edition migration.

Table 25. Unconnected, Non-Existent Port Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
Synthesis allows you to instantiate and name unconnected or non-existent ports that do not exist on the module.	Synthesis generates a syntax error for detection of mixed blocking and non-blocking assignments within an ALWAYS block.

5.4.11. Avoid Illegal Parameter Ranges

Intel Quartus Prime Pro Edition synthesis generates an error for detection of constant numeric (integer or floating point) parameter values that exceed the language specification. Other Quartus software products allow constant numeric (integer or floating point) values for parameters that exceed the language specifications. To avoid syntax errors, ensure that constant numeric (integer or floating point) values for parameters conform to the language specifications.

5.4.12. Update Verilog HDL and VHDL Type Mapping

Intel Quartus Prime Pro Edition synthesis requires that you use `0` for "`false`" and `1` for "`true`" in Verilog HDL files (`.v`). Other Quartus software products map "true" and "false" strings in Verilog HDL to TRUE and FALSE Boolean values in VHDL. Intel Quartus Prime Pro Edition synthesis generates an error for detection of non-Verilog HDL constructs in `.v` files. To avoid syntax errors, ensure that your RTL accommodates these standards.

5.5. Migrating to Intel Quartus Prime Pro Edition Revision History

This chapter has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	Initial release in Intel Quartus Prime Standard Edition User Guide.

A. Intel Quartus Prime Standard Edition User Guides

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

Related Information

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

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