



Intel® Quartus® Prime Pro Edition User Guide

Getting Started

Updated for Intel® Quartus® Prime Design Suite: **22.4**

Answers to Top FAQs:

- Q What do I need for FPGA design?**
A [FPGA Basic Design Prerequisites](#) on page 6
- Q What do I need to download to use Quartus?**
A [Intel FPGA Design Software for Download](#) on page 6
- Q Which Quartus version should I use?**
A [Selecting a Software Edition](#) on page 7
- Q How do I setup a project?**
A [Creating a New Project](#) on page 10
- Q Do you have an example design to start with?**
A [Start a Project from a Design Example](#) on page 11
- Q Does Quartus work with my other tools?**
A [Integrating Other EDA Tools](#) on page 35
- Q How do I add my IP?**
A [Adding Your IP to IP Catalog](#) on page 82
- Q How do I migrate an old project?**
A [Migrate to Intel Quartus Prime Pro Edition](#) on page 117
- Q Do you have basic tool training?**
A [Intel FPGA Technical Training Curriculum](#)

Contents

1. Introduction to Intel® Quartus® Prime Pro Edition.....	5
1.1. FPGA Basic Design Prerequisites.....	6
1.2. Selecting an Intel Quartus Prime Software Edition.....	7
1.3. Introduction to Intel Quartus Prime Pro Edition Revision History.....	8
2. Managing Intel Quartus Prime Projects.....	10
2.1. Creating a New FPGA Design Project.....	10
2.1.1. Using the Board-Aware Flow.....	10
2.2. Viewing Basic Project Information.....	20
2.2.1. Using the Compilation Dashboard.....	21
2.2.2. Viewing Project Reports.....	22
2.2.3. Viewing Project Messages.....	23
2.3. Intel Quartus Prime Project Contents.....	27
2.3.1. Project File Best Practices.....	27
2.4. Managing Project Settings.....	28
2.4.1. Optimizing Project Settings.....	29
2.5. Managing Logic Design Files.....	33
2.5.1. Including Design Libraries.....	33
2.5.2. Creating a Project Copy.....	34
2.6. Managing Timing Constraints.....	34
2.7. Integrating Other EDA Tools.....	35
2.8. Exporting Compilation Results.....	36
2.8.1. Exporting a Version-Compatible Compilation Database	36
2.8.2. Importing a Version-Compatible Compilation Database	38
2.8.3. Creating a Design Partition.....	39
2.8.4. Exporting a Design Partition.....	41
2.8.5. Reusing a Design Partition.....	43
2.8.6. Viewing Quartus Database File Information.....	43
2.8.7. Clearing Compilation Results.....	45
2.9. Migrating Projects Across Operating Systems.....	46
2.9.1. Migrating Design Files and Libraries.....	46
2.9.2. Design Library Migration Guidelines.....	47
2.10. Archiving Projects.....	47
2.10.1. Manually Adding Files To Archives.....	48
2.10.2. Archiving Projects for Service Requests.....	48
2.10.3. Archiving Projects for External Revision Control.....	49
2.10.4. Creating Database-Only Archives.....	50
2.11. Command-Line Interface.....	51
2.11.1. Project Revision Commands.....	52
2.11.2. Project Archive Commands.....	52
2.11.3. Project Database Commands.....	53
2.12. Managing Projects Revision History.....	54
3. Design Planning.....	58
3.1. Design Planning.....	58
3.2. Create a Design Specification and Test Plan.....	58
3.3. Plan for the Target Device or Board.....	58
3.3.1. Device Migration Planning.....	59

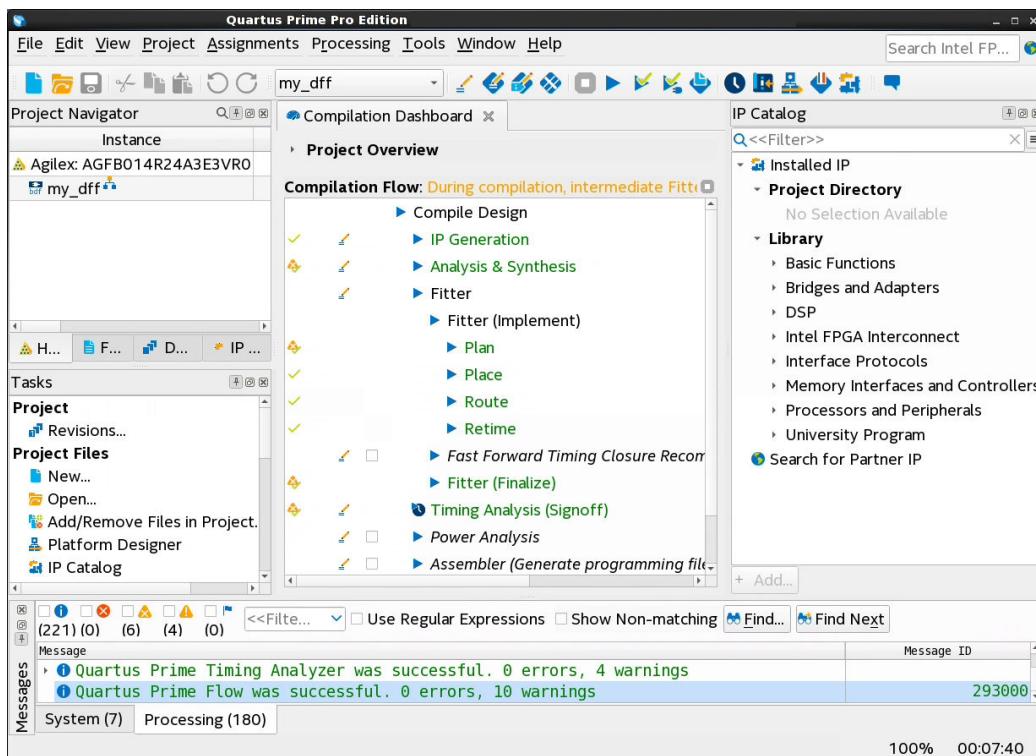
3.4. Plan for Intellectual Property Cores.....	60
3.5. Plan for Standard Interfaces.....	60
3.6. Plan for Device Programming.....	61
3.7. Plan for Device Power Consumption.....	62
3.8. Plan for Interface I/O Pins.....	66
3.9. Plan for other EDA Tools.....	68
3.9.1. Third-Party Synthesis Tools.....	68
3.9.2. Third-Party Simulation Tools.....	68
3.10. Plan for On-Chip Debugging Tools.....	69
3.11. Plan HDL Coding Styles.....	70
3.11.1. Design Recommendations.....	70
3.11.2. Recommended HDL Coding Styles.....	70
3.11.3. Managing Metastability.....	70
3.12. Plan for Hierarchical and Team-Based Designs.....	71
3.12.1. Flat Compilation without Design Partitions.....	71
3.13. Design Planning Revision History.....	72
4. Introduction to Intel FPGA IP Cores.....	75
4.1. IP Catalog and Parameter Editor.....	76
4.1.1. The Parameter Editor.....	77
4.2. Installing and Licensing Intel FPGA IP Cores.....	77
4.2.1. Intel FPGA IP Evaluation Mode.....	78
4.3. IP General Settings.....	81
4.4. Adding IP to IP Catalog.....	82
4.5. Best Practices for Intel FPGA IP.....	83
4.6. Specifying the IP Core Parameters and Options (Intel Quartus Prime Pro Edition).....	83
4.6.1. Applying Preset Parameters for Specific Applications.....	85
4.6.2. Customizing IP Presets.....	87
4.7. IP Core Generation Output (Intel Quartus Prime Pro Edition).....	90
4.8. Scripting IP Core Generation.....	92
4.9. Modifying an IP Variation.....	93
4.10. Upgrading IP Cores.....	93
4.10.1. Upgrading IP Cores at Command-Line.....	96
4.10.2. Migrating IP Cores to a Different Device.....	97
4.10.3. Troubleshooting IP or Platform Designer System Upgrade.....	98
4.11. Simulating Intel FPGA IP Cores.....	99
4.11.1. Generating IP Simulation Files.....	99
4.11.2. Scripting IP Simulation.....	101
4.12. Generating Simulation Files for Platform Designer Systems and IP Variants.....	110
4.13. Synthesizing IP Cores in Other EDA Tools.....	112
4.14. Instantiating IP Cores in HDL.....	112
4.14.1. Example Top-Level Verilog HDL Module.....	113
4.14.2. Example Top-Level VHDL Module.....	113
4.15. Support for the IEEE 1735 Encryption Standard.....	113
4.16. Introduction to Intel FPGA IP Cores Revision History.....	115
5. Migrating to Intel Quartus Prime Pro Edition.....	117
5.1. Keep Pro Edition Project Files Separate.....	117
5.2. Upgrade Project Assignments and Constraints.....	117
5.2.1. Modify Entity Name Assignments.....	118
5.2.2. Resolve Timing Constraint Entity Names.....	118

5.2.3. Verify Generated Node Name Assignments.....	119
5.2.4. Replace Logic Lock (Standard) Regions.....	119
5.2.5. Modify Signal Tap Logic Analyzer Files.....	121
5.2.6. Remove References to .qip Files.....	122
5.2.7. Remove Unsupported Feature Assignments.....	122
5.3. Upgrade IP Cores and Platform Designer Systems.....	123
5.4. Upgrade Non-Compliant Design RTL.....	124
5.4.1. Verify Verilog Compilation Unit	124
5.4.2. Update Entity Auto-Discovery.....	125
5.4.3. Ensure Distinct VHDL Namespace for Each Library.....	126
5.4.4. Remove Unsupported Parameter Passing.....	126
5.4.5. Remove Unsized Constant from WYSIWYG Instantiation.....	126
5.4.6. Remove Non-Standard Pragmas.....	127
5.4.7. Declare Objects Before Initial Values.....	127
5.4.8. Confine SystemVerilog Features to SystemVerilog Files.....	127
5.4.9. Avoid Assignment Mixing in Always Blocks.....	128
5.4.10. Avoid Unconnected, Non-Existent Ports.....	128
5.4.11. Avoid Illegal Parameter Ranges.....	128
5.4.12. Update Verilog HDL and VHDL Type Mapping.....	129
5.5. Migrating to Intel Quartus Prime Pro Edition Revision History.....	129
6. Document Archives.....	130
A. Intel Quartus Prime Pro Edition User Guides.....	131

1. Introduction to Intel® Quartus® Prime Pro Edition

This user guide describes basic concepts and operation of the Intel® Quartus® Prime Pro Edition software. This software provides a complete design environment for FPGA designs targeting the Intel Agilex™, Intel Stratix® 10, Intel Arria® 10, and Intel Cyclone® 10 GX devices.⁽¹⁾

Figure 1. Intel Quartus Prime Pro Edition Software GUI



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.

⁽¹⁾ A field-programmable gate array (FPGA) is a specialized integrated circuit that you can customize and reconfigure multiple times. To learn about and select a target Intel FPGA device family, refer to <https://www.intel.com/content/www/us/en/products/details/fpga.html>.

1.1. FPGA Basic Design Prerequisites

Using the Intel Quartus Prime software to create a basic FPGA design requires the following prerequisite knowledge and software installation and licensing:

Prerequisite Knowledge

- Basic knowledge of digital logic design.
- Basic knowledge of how to describe a hardware design using VHDL, Verilog HDL, SystemVerilog, or EDA schematic tools.

Note: You can accelerate design creation and success by starting your design project from a pre-verified design example that targets an Intel FPGA development board, as [Creating a New Project from a Design Example](#) on page 11 describes.

Prerequisite Software and Licensing

To use the Intel Quartus Prime software for FPGA design, your system must first meet the minimum Windows or Linux system requirements for installation. Next, you download and install the software components. You must purchase a license for any applicable software prior to production use.

The Intel [FPGA Software Download Center](#) webpage allows you to download Intel FPGA software and IP cores. To download software, select and visit the desired software landing page from the [FPGA Software Download Center](#).

Table 1. Intel FPGA Design Software for Download and Installation

Software	Description	Required?
Intel Quartus Prime Software	You must install an edition of the Intel Quartus Prime software to compile your design and generate files to program the target FPGA. The Intel Quartus Prime installation also includes the Intel FPGA IP Library and Nios II EDS. The Intel Quartus Prime Pro Edition software 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.	Yes
Device support Files (.qdz)	You must install support for one or more Intel FPGA device families as part of the Intel Quartus Prime software installation when prompted. Only select the support files for the device families that you plan to target.	Yes
Intel Quartus Prime Help Files	Optional HTML help system that provides descriptions of the Intel Quartus Prime software GUI.	No
Questa* Intel FPGA Edition Simulator	Intel provides this simulator which is a version of the Questa* Advanced simulator targeted for simulating Intel Quartus Prime designs. The Questa Intel FPGA Edition simulator supports the Intel FPGA gate-level simulation libraries, and includes behavioral simulation, HDL test benches, and Tcl scripting support.	No
DSP Builder for Intel® FPGAs	Digital signal processing (DSP) design tool that enables Hardware Description Language (HDL) generation of DSP algorithms directly from the MathWorks Simulink* environment onto Intel FPGAs.	No

Note: Refer to [Intel FPGA Software Installation and Licensing](#) for step-by-step installation and licensing instructions for all Intel FPGA software.

Related Information

- Intel FPGA Software Installation and Licensing
- FPGAs for Dummies eBook

1.2. Selecting an Intel Quartus Prime Software Edition

Depending on your target FPGA device and desired software features, you can choose either the Intel Quartus Prime Edition software or the Intel Quartus Prime Standard Edition software for your Intel FPGA design.

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

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 Device Support		✓
Intel Agilex™ Device Support		✓
Partial Reconfiguration		✓
Block-Based (Hierarchical) Design Flows		✓
OpenCL support		✓
Incremental Fitter Optimization		✓
Interface Planner (formerly BluePrint)		✓

The following features are only available in the Intel Quartus Prime Pro Edition software:

- Hyper-Aware Design Flow—use Hyper-Retiming to reach the highest performance in Intel Agilex and Intel Stratix 10 devices.
- Advanced synthesis—integrates new, stricter language parser supporting all major IEEE RTL languages, with enhanced algorithms, and parallel synthesis capabilities, and support for SystemVerilog 2009.
- Hierarchical project structure—preserve individual post-synthesis, post-placement, and post-place and route results for design instances. Optimizes without impacting other partition placement or routing.
- Incremental Fitter Optimizations—run and optimize Fitter stages incrementally. Each Fitter stage generates detailed reports.
- Faster, more accurate I/O placement—plan interface I/O in Interface Planner.
- Platform Designer (Pro)—builds on the system design and custom IP integration capabilities of Platform Designer (Standard). Platform Designer (Pro) introduces hierarchical isolation between system interconnect and IP components.
- Block-Based Design Flows—preserve and reuse design blocks at various stages of compilation.

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 or RTL to schematic conversion

Related Information

[Migrating to Intel Quartus Prime Pro Edition on page 117](#)

1.3. Introduction to Intel Quartus Prime Pro Edition Revision History

Document Version	Intel Quartus Prime Version	Changes
2022.06.20	22.2	<ul style="list-style-type: none">• Added new <i>Top FAQs</i> navigation to document cover.• Revised <i>Introduction</i> to add FPGA definition and device selection footnote.• Added new <i>FPGA Basic Design Prerequisites</i> topic.• Added new <i>Experiment with a Design Example</i> topic.
2020.09.28	20.3	<ul style="list-style-type: none">• Updated GUI screenshot in Introduction.
2019.09.30	19.3	<ul style="list-style-type: none">• Added compilation support for Intel Agilex devices.
2018.09.24	18.1	<ul style="list-style-type: none">• Added screenshot of Intel Quartus Prime Pro Edition GUI.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.05.07	18.0	Initial release as separate chapter of <i>Getting Started User Guide</i> . Separated <i>Migrating to Intel Quartus Prime Pro Edition</i> as independent chapter in user guide.
2017.11.06	17.1	<ul style="list-style-type: none"> Described Intel Quartus Prime tool name updates for Platform Designer (Qsys), Interface Planner (BluePrint), Timing Analyzer (TimeQuest), Eye Viewer (EyeQ), and Advanced Link Analyzer (Advanced Link Analyzer). Added Verilog HDL Macro example. Updated for latest Intel branding conventions.
2017.05.08	17.0	<ul style="list-style-type: none"> Removed statement about limitations for safe state machines. The Compiler supports safe state machines. State machine inference is enabled by default. Added reference to Block-Based Design Flows. Removed procedure on manual dynamic synthesis report generation. The Compiler automatically generates dynamic synthesis reports when enabled.
2016.10.31	16.1	<ul style="list-style-type: none"> Implemented Intel rebranding. Added reference to Partial Reconfiguration support. Added to list of Intel Quartus Prime Standard Edition features unsupported by Intel Quartus Prime Pro Edition. Added topic on Safe State Machine encoding. Described unsupported Intel Quartus Prime Standard Edition physical synthesis options. Removed deprecated Per-Stage Compilation (Beta) Compilation Flow. Changed title from "Remove Filling Vectors" to "Remove Unsized Constant".
2016.05.03	16.0	<ul style="list-style-type: none"> Removed software beta status and revised feature set. Added topic on Safe State Machine encoding. Added Generating Dynamic Synthesis Reports. Corrected statement about Verilog Compilation Unit. Corrected typo in Modify Entity Name Assignments. Added description of Fitter Plan, Place and Route stages, reporting, and optimization. Added Per-Stage Compilation (Beta) Compilation Flow. Added Platform Designer information. Added OpenCL and Signal Tap with routing preservation as unique Pro Edition features. Clarified limitations for multiple Logic Lock instances in the same region.
2015.11.02	15.1	<ul style="list-style-type: none"> First version of document.

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 viewing and exporting the design compilation results.

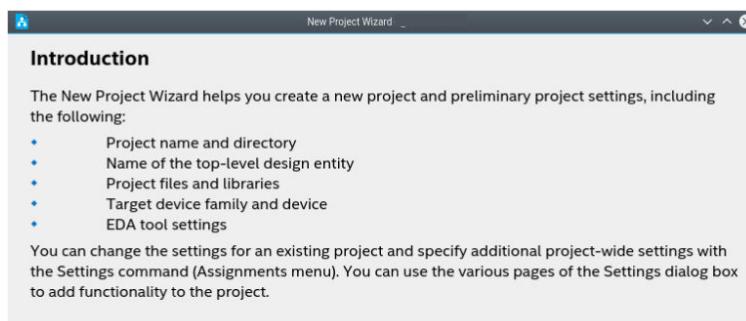
After you create or open a project, the GUI displays integrated information and controls for the open project.

2.1. Creating a New FPGA Design Project

The Intel Quartus Prime software makes it easy for you to quickly setup a new FPGA design project. Click **File > New Project Wizard** to quickly setup and open a new project tailored for your application. The wizard guides you through specifying various options for new project setup.

The wizard includes access to helpful project templates and design examples that allow you to preconfigure project settings for specific applications, FPGA devices, and target boards. Alternatively, you can create a blank design project and specify all design files and settings.

Figure 3. **New Project Wizard - Introduction**



2.1.1. Using the Board-Aware Flow

The Intel Quartus Prime Pro Edition software allows you to create a system that targets a specific development board, rather than only targeting a specific FPGA device. When you target a specific development board, the Intel Quartus Prime software is *aware* of the target board (board-aware) which accelerates the process of appropriately configuring, connecting, and validating IP for the target board.

What is the Intel Quartus Prime Software Board-Aware Flow?

In the board-aware flow, you can optionally start your project from a pre-verified design example (rather than an empty project) and target a specific Intel FPGA development board. You can also create appropriate IP presets to target the specific board. The Intel Quartus Prime Platform Designer system integration tool is also board-aware, allowing you to automatically set pin assignments and export appropriate system interfaces for the target board.

The board-aware flow simplifies the application of appropriate parameters and pin assignments for the instantiated IP in your project, thereby reducing the chance of configuration errors. You can also save and reuse your preferred and verified board and IP configurations for use in other projects that target the same IP or board.

The board-aware flow helps to ensure the proper hand-off, consistency, and reuse of configuration options across multiple projects, developers, and boards.

Note: To define new boards and IP preset files in Platform Designer, refer to *Intel Quartus Prime Pro Edition User Guide: Platform Designer* and *AN 988: Using the Board-Aware Flow in the Intel Quartus Prime Pro Edition Software*.

Related Information

- [Creating a New Project from a Design Example](#) on page 11
- [Specifying a Target Board for the Project](#) on page 19
- [Applying Preset Parameters for Specific Applications](#) on page 85
- [Intel Quartus Prime Pro Edition User Guide: Platform Designer](#)
- [AN 988: Using the Board-Aware Flow in the Intel Quartus Prime Pro Edition Software](#)

2.1.1.1. Creating a New Project from a Design Example

The Intel Quartus Prime software provides access to installed and online platform- and board-specific design examples that you can use as a starting point for your own design. You can accelerate your design progress by starting from a pre-validated design example that installs with the Intel Quartus Prime software or is available online.

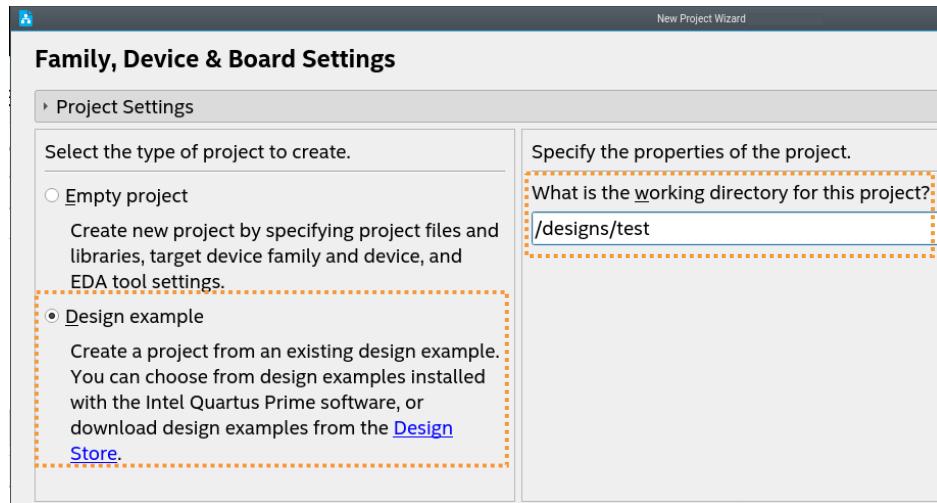
This technique can be especially helpful if you are new to FPGA design or EDA design tools. The design example can help you to quickly analyze a validated design on a board and appropriately configure it in various ways to match your users' needs. Alternatively, you can start with an **Empty Project** for which you specify all settings and design files.

- **Pre-installed design examples**—you can immediately access the design examples that install along with the Intel Quartus Prime software installation at: <quartus>\acds\quartus\common\board_desigs.
- **Online design examples**—you can access design examples hosted online, which includes designs from the [Intel FPGA Design Store](#).
- **Downloaded design examples**—you can access your previously downloaded design examples, or any design example that you store in a local drive, under downloaded reference designs.

To create a new Intel Quartus Prime project that is based on a design example, follow these steps:

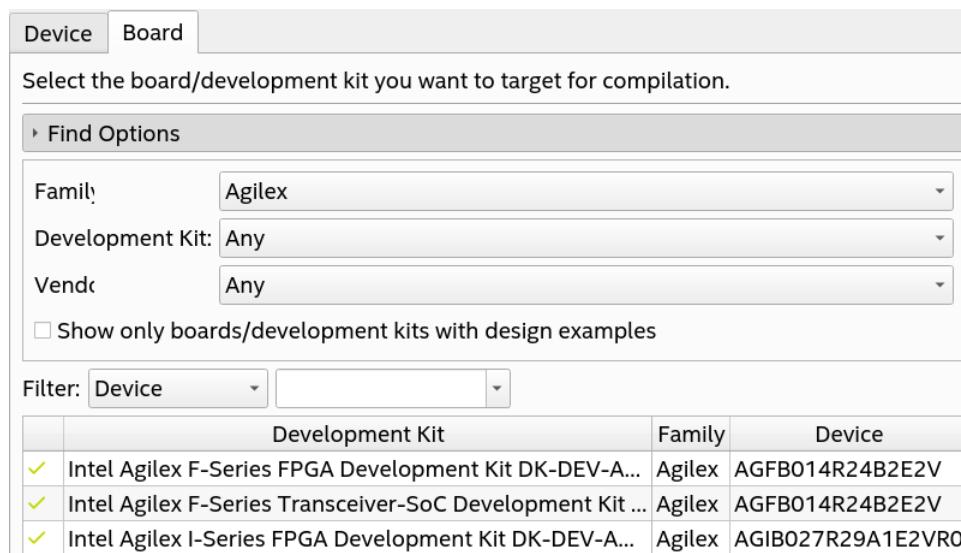
1. In the Intel Quartus Prime software, click **File > New Project Wizard**. Click **Next** to view the **Family, Device & Board Settings** wizard page.
2. Under the **Select the type of project to create**, select **Design Example** and click **Next**. The **Family, Device & Board Settings** page appears, allowing you to find and select the design example from which to base your project.

Figure 4. Family, Device & Board Settings Page of New Project Wizard



3. Under **What is the working directory for this project?**, specify the directory to store your project files and click **Next**.
4. Under **Find Options**, select the **Family**, **Development Kit**, and **Vendor** design example you want to use. Refer to **Family, Device & Board Settings** on page 13.

Figure 5. Board Tab in New Project Wizard



The search results display the design examples that meet your search criteria.

5. Select the design example that you want in the search results and click **Next**. If the design example is licensed by Intel FPGA, a **Software License Agreement** page appears that prompts you to accept the license agreement before you can proceed.
6. Click **Next** to proceed to the **Summary** page.
7. Click **Finish** to deploy the selected design example in the Intel Quartus Prime software. When a design example downloads, the design's .par downloads to the download path that you define in **More Settings**, but the design itself extracts to the project working directory that you specify.

Also refer to [Accessing Online Design Examples](#) on page 15 and [Accessing Downloaded Design Examples](#) on page 18.

Related Information

[Intel FPGA Design Examples](#)

2.1.1.1.1. Family, Device & Board Settings

The following options are available in the **Family, Device & Board Settings** page of the **New Project Wizard**. Specify these options to locate and deploy a validated design example targeting a specific board as a starting point for your FPGA design project. Some options are only available from **File > Open Example Project**

Table 2. Family, Device & Board Settings Page Options

Option	Description
Select the type of project to create	<ul style="list-style-type: none">• Empty—create a new empty FPGA design project to which you add all design files, settings, and constraints.• Design example—create a new project from an existing design example. You can access installed or online available design examples.
What is the working directory for this project?	Specifies the directory where you want to extract and deploy the design example.
Find Options	Allows you to filter design example search results by one of the following facets: <ul style="list-style-type: none">• Load from > Pre-installed design examples—specifies that search includes examples installed with the Intel Quartus Prime software.• Load from > User downloaded design examples—search includes design examples that you download or your own design examples that you store in a local repository.• Load from > Online design examples—search includes design examples hosted online, including examples from the Intel FPGA Design Store.• Family—search only includes design examples for the device families that you specify. You can specify multiple values.• Quartus Prime version—search only includes design examples that support the Intel Quartus Prime software version that you specify. You can specify multiple pipe separated values.• Development kit—search only includes design examples that support the Intel FPGA development kit that you specify. You can specify multiple pipe separated values.
More settings button	Opens the Options panel that allows you to configure the Internet Connectivity and Design Examples connection and download settings, as Design Examples Options on page 17 describes.
Reset button	Resets the Find Options to default settings.

continued...

Option	Description
Legend panel	Displays the meaning of design example status icons in the search results. Design example status is validated (checkmark icon), unvalidated (question mark icon), or unsupported for the current Intel Quartus Prime software version (x icon).
Filter text box	Specifies a text string to further filter design example search results according to any text string you specify.
Search results list	Displays the design examples, status, and location that match your search filters.
Details panel	Displays a detailed description and diagram of the selected design example.
Design Store button	Opens the Design Store website in your default web browser from which you can download available Intel FPGA validated design examples.

2.1.1.1.2. Accessing Pre-Installed Design Examples

The Intel Quartus Prime software installation includes design examples for your immediate use.

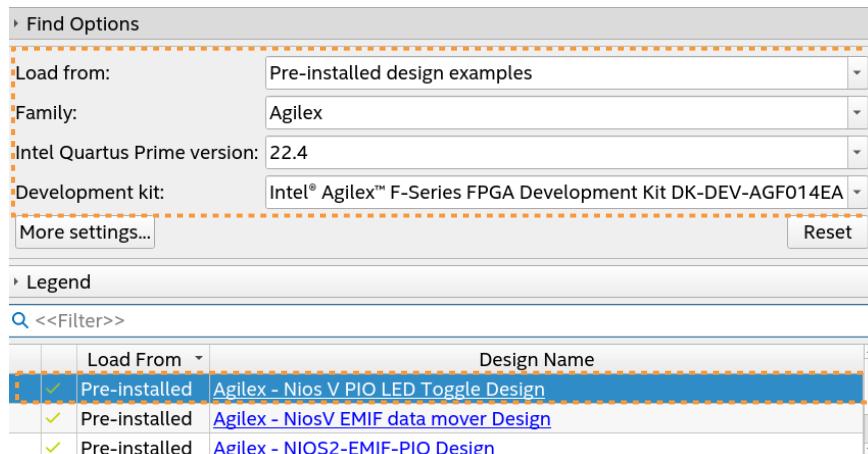
You can access the pre-installed design examples while using the Intel Quartus Prime software using the following methods:

- Click **File > New Project Wizard > Family, Device & Board Settings** page.
- Click **Open Example Project** on the **Home** tab (**Help > Home**).
- Click **File > Open Example Project**.

To create a new project based on pre-installed design examples, follow these steps Intel Quartus Prime Pro Edition software:

1. Click **File > Open Example Project**. The **Design Example** page of the **New Project Wizard** opens.
2. For **What is the working directory for this project?**, specify the directory location to store your project files.

Figure 6. Find Options Locate Pre-Installed Design Example



3. Under **Find Options**, specify the following settings to filter the list of design examples for the target device and board. Also refer to [Family, Device & Board Settings](#) on page 13.

- a. In **Load from**, select the **Pre-Installed design examples**, repository.
 - b. In **Family**, select your target FPGA device family.
 - c. In **Intel Quartus Prime version**, select the software version.
 - d. In **Development kit**, select the target kit or board.
4. Under **Design name**, select the design example to base your project on.
 5. Click **Next**, and then click **Finish**. The design extracts to the working directory and opens in the Intel Quartus Prime software.

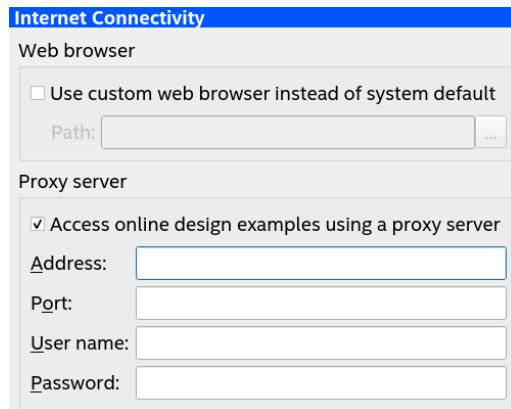
2.1.1.1.3. Accessing Online Design Examples

You can create a new project based on a design example that you access from an online repository. To use this method, you may need to specify a proxy server for access and the download path.

To create a new project in the Intel Quartus Prime software based on online design examples, follow these steps:

1. Click **File > Open Example Project**. The **Design Example** page of the **New Project Wizard** opens.
2. For **What is the working directory for this project?**, specify the directory location to store your project files.
3. Click the **More Settings** button. The **Options** dialog box opens with the **Internet Connectivity** tab open by default.

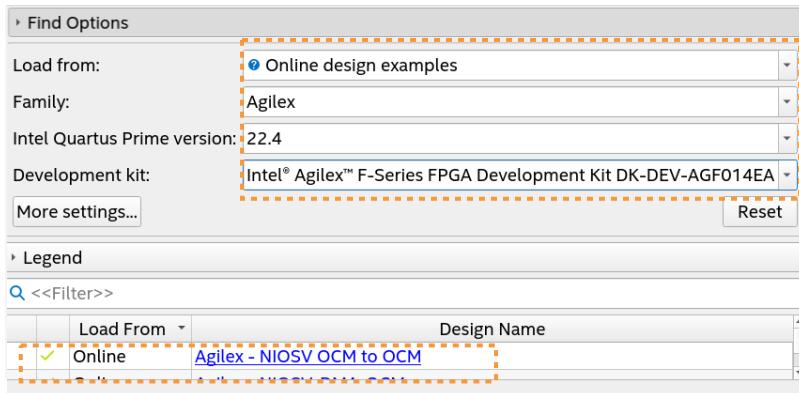
Figure 7. Intel Quartus Prime Software Internet Connectivity Settings



4. If your internet connection requires a proxy server (using VPN), turn on the **Access online design examples using a proxy server** option, and then specify your proxy **Address**, **Port**, **User name**, and **Password**. If your internet connection does not require a proxy server, skip this step.
5. On the **Design Example Search Locations** tab, specify the **Download path** for download of the design example .par file.
6. Click **OK**.
7. Under **Find Options**, specify the following settings. Also refer to [Family, Device & Board Settings](#) on page 13.

- a. In **Load from**, select **Downloaded design examples**.
- b. In **Family**, **Intel Quartus Prime version**, and **Development kit** fields select the values to match your target design and board.
8. In the design example list, select the design that you want to deploy.

Figure 8. Online Agilex - I/O PLL Reconfiguration Design



9. Click **Next**, and then click **Finish**. The design extracts to the working directory and opens in the Intel Quartus Prime software.

Internet Connectivity Options

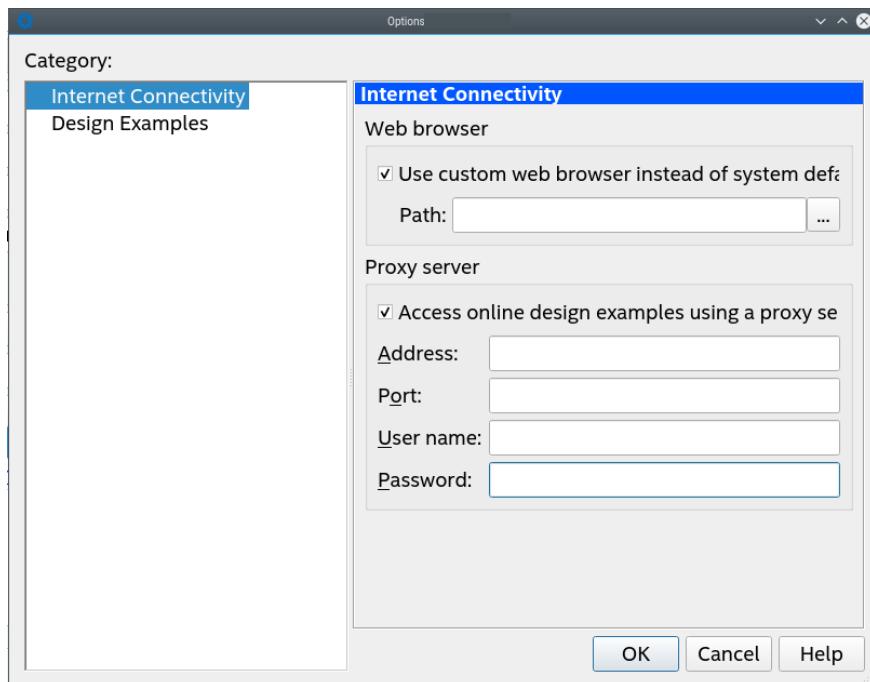
You can specify the following internet connectivity options that determine how the Intel Quartus Prime software connects to the internet for various functions, such as accessing Help and design examples, with either of the following:

- Click **Tools > Options > Internet Connectivity**
- Or
- Click **More Settings** on the **Design Example** page of the **New Project Wizard** (**File > New Project Wizard**).

The following options are available on the **Internet Connectivity** options page.

Table 3. Internet Connectivity Options

Option	Description
Web browser	Specifies the web browser that deploys when the Intel Quartus Prime software accesses the internet, including the Intel FPGA Design Store web page. Enable Use custom web browser to specify the path to your preferred supported web browser.
Proxy server	Specify options if connecting to the internet through a proxy server. To access online design examples specify the appropriate option: <ul style="list-style-type: none"> • Access online design examples using a proxy server—turn on this option if you are connected to the internet through a VPN. Turn off this option if you are not connected to the internet through a VPN (such as connection through a private network).

Figure 9. Internet Connectivity Page

Related Information

[Design Example Options Page](#) on page 17

Design Examples Options

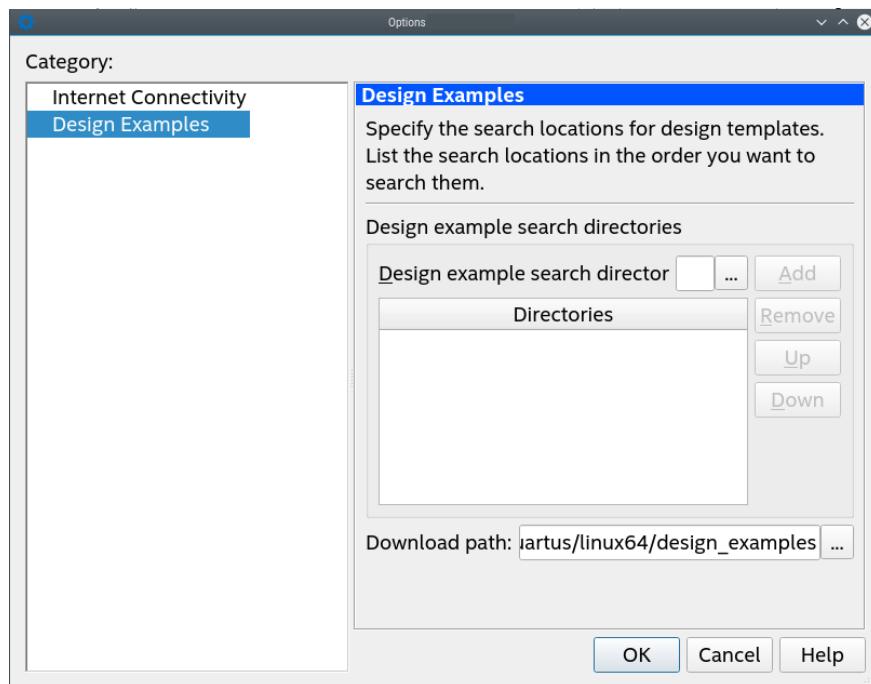
You can click the following to specify options that determine how the Intel Quartus Prime software accesses available design examples.

- Click **File > New Project Wizard** and then click the **More Settings** button on the **Design Example** page of the **New Project Wizard**.

Table 4. Design Examples Options

Option	Description
Design Example search directory	Specifies the local directories that the Intel Quartus Prime software searches for design examples. This setting determines which directories you include in search when using the New Project Wizard to start a project from an existing design example. Click Add , Remove , Up , or Down to change the search order and contents in the Directories list.
Directories	Lists the various directories that you include in the design example search path for the New Project Wizard .
Download path	Specifies the path for download of online design examples.

Figure 10. Design Examples Page



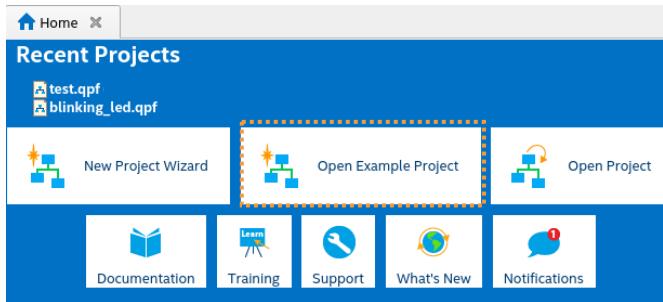
2.1.1.1.4. Accessing Downloaded Design Examples

You can create a new project from a design example that you have previously downloaded. Download a design example .par file from an online repository (such as the Intel FPGA Design Store) into your working directory. Designs that you create yourself and store in a local drive also appear as downloaded examples.

To create a new project based on downloaded design examples, follow these steps:

1. Download a design example, as [Accessing Online Design Examples](#) on page 15 describes.

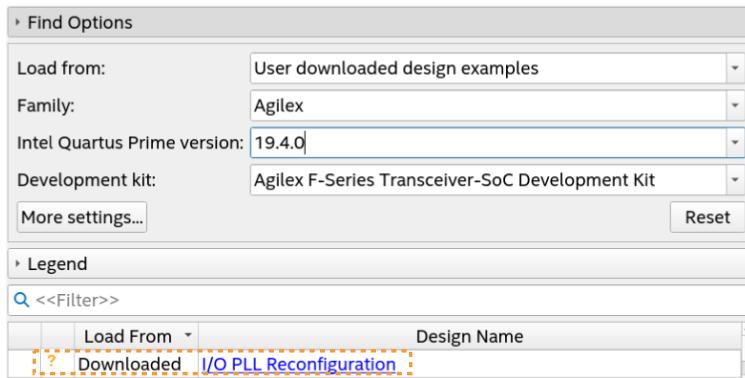
Figure 11. Open Example Project Icon on Home Page



2. Click the **Open Example Project** icon on the Intel Quartus Prime Pro Edition **Home** page. The **Design Example** page of the **New Project Wizard** opens.
3. For **What is the working directory for this project?**, specify the directory location to store your project files.

4. Under **Find Options**, specify the following settings. Refer to [Family, Device & Board Settings](#) on page 13.
 - a. In **Load from**, select **Downloaded design examples**.
 - b. In **Family**, **Intel Quartus Prime version**, and **Development kit** fields select the values to match your target design and board.
5. In the design example list, select the design that you want to deploy.

Figure 12. Downloaded Agilex I/O PLL Reconfiguration Design



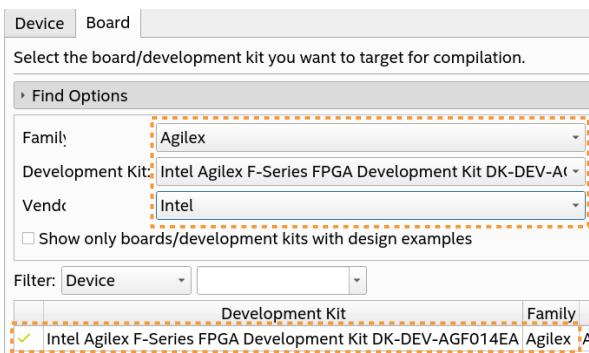
6. Click **Next**, and then click **Finish**. The design extracts to the working directory and opens in the Intel Quartus Prime software.

2.1.1.2. Specifying a Target Board for the Project

You can specify the target for a new project in the New Project Wizard, or you can specify a target board for an existing project by clicking **Assignments > Device**. To specify a target board for an existing project, follow these steps:

1. Click **Assignments > Device**. The **Device** dialog box appears.
2. Click the **Board** tab. The **Board** tab allows you to target a specific FPGA device board, rather than just a specific FPGA device.

Figure 13. Board Tab Settings

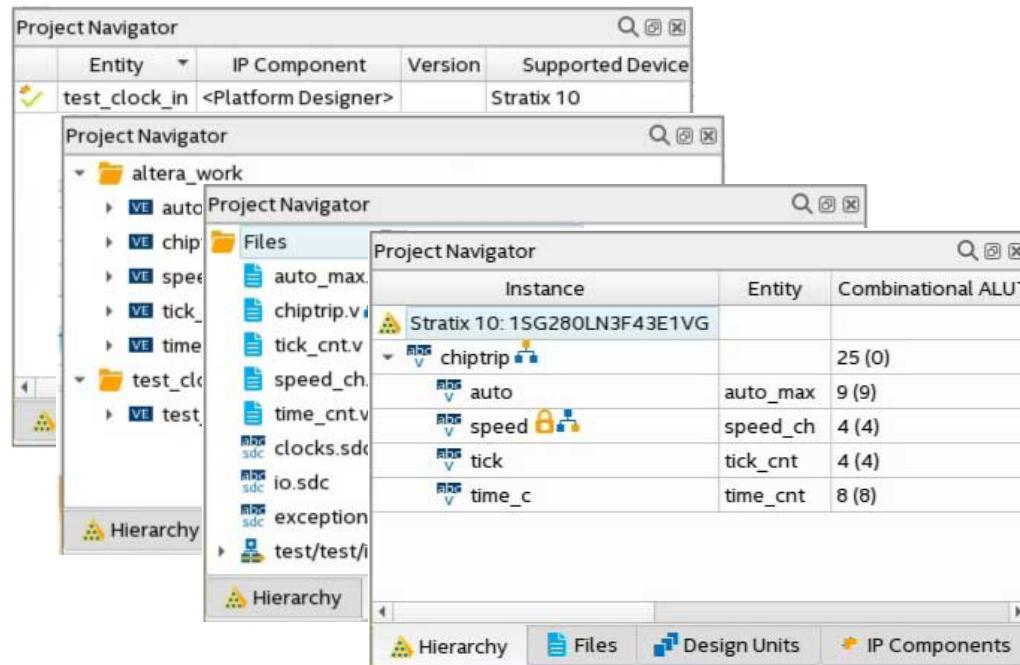


3. In the **Family**, **Intel Quartus Prime version**, and **Development kit** fields, select the values to match your target design and board.
4. Click the desired board in the list. The board details appear in the right pane.
5. Click **OK**. Your project now targets the specified board and device.

2.2. Viewing Basic Project Information

View basic information about your project in the Project Navigator, the **Tasks** pane, Compilation Dashboard, Report panel, and **Messages** window.

Figure 14. Project Navigator Hierarchy, Files, Design Units, and IP Components Tabs



The Project Navigator

The **Project Navigator** (**View > Project Navigator**) displays the elements of your project, such as the design files, IP components, and your project hierarchy (after elaboration). Right-click in the **Project Navigator** to locate the elements of your project. Project information appears on the **Files**, **Hierarchy**, **Design Units**, and **IP Components** tabs.

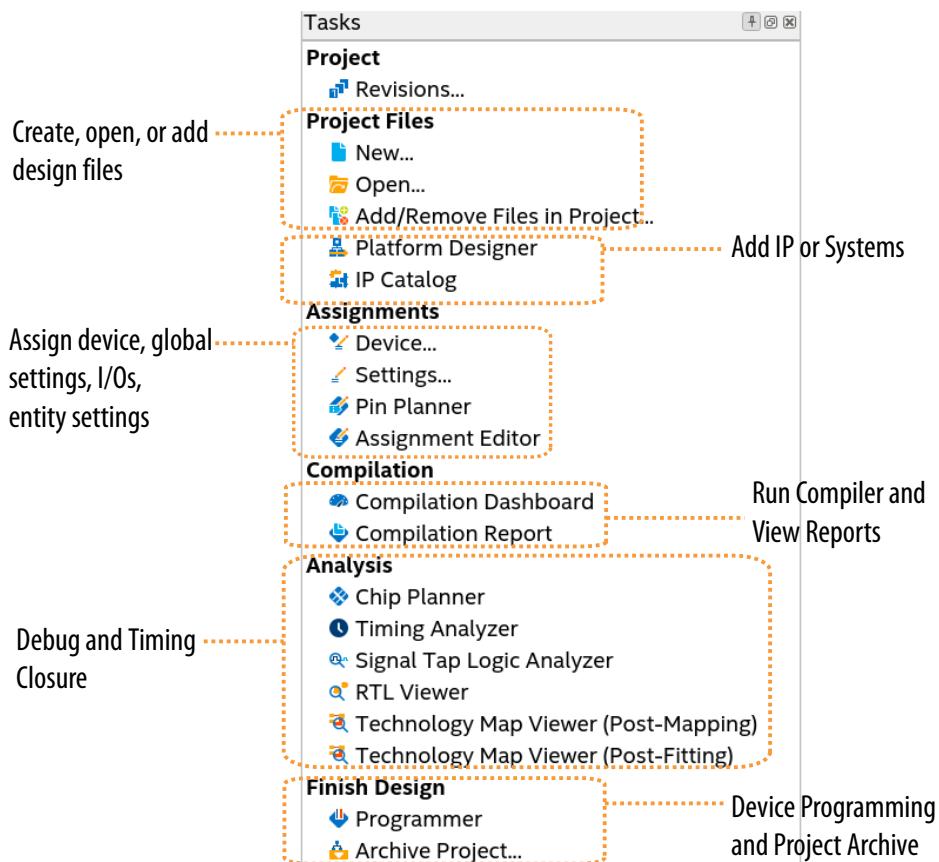
Table 5. Project Navigator Tabs

Project Navigator Tab	Description
Files	Lists all design files in the current project. Right-click design files in this tab to run these commands: <ul style="list-style-type: none"> Open the file Remove the file from project View file Properties
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 components, and third-party IP. Click Launch IP Upgrade Tool from this tab to upgrade outdated IP components.

Project Tasks Pane

The **Tasks** pane (**View > Tasks**) launches common project tasks, such as creating design files, adding IP, running compilation, and device programming.

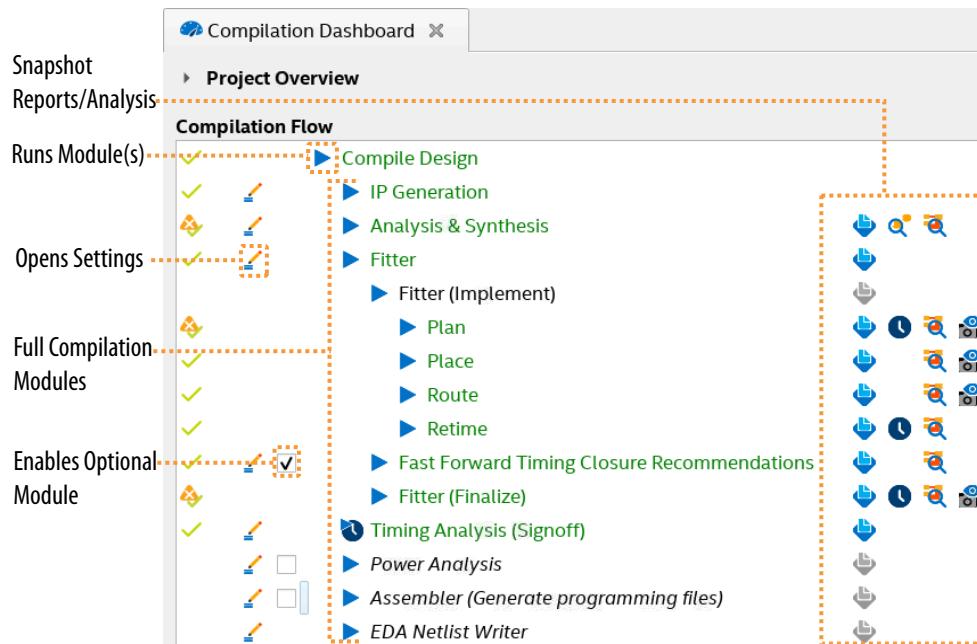
Figure 15. Tasks Pane



2.2.1. Using the Compilation Dashboard

The Compilation Dashboard provides immediate access to settings, controls, and reporting for each stage of the compilation flow.

The Compilation Dashboard appears by default when you open a project, or you can click **Compilation Dashboard** in the Tasks window to re-open it.

Figure 16. Compilation Dashboard


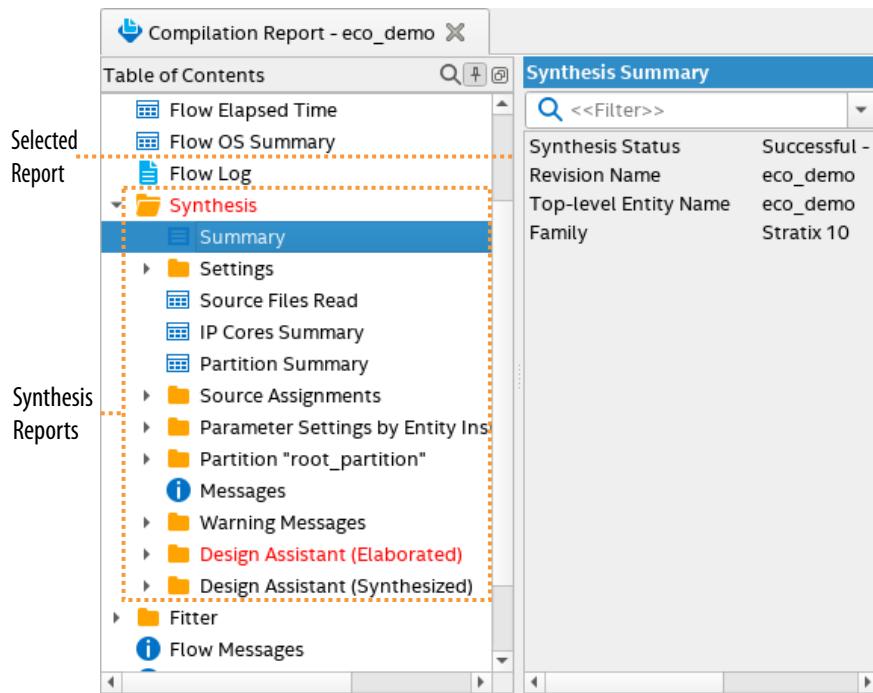
- Click the **Pencil** icon to edit settings for that stage of the compilation flow.
- Click any Compiler stage to run one or more Compiler stage. You can click a Compiler stage to resume an interrupted compilation flow provided no compilation settings have changed from the initial start of the compilation flow.
- Click the **Report**, **RTL Viewer**, **Technology Map Viewer**, **Timing Analyzer**, or **Snapshot Viewer** icons for analysis of stage results.

As the Compiler progresses through the flow, the dashboard updates the status of each module, and enables icons that you can click for reports and analysis. The dashboard is also updated if you run your compilation flow from a command line with the `quartus_sh --flow` command.

2.2.2. Viewing Project Reports

The Compilation Report panel updates dynamically to display detailed reports during project processing. To access Compilation Reports, click (**Processing** > **Compilation Report**).

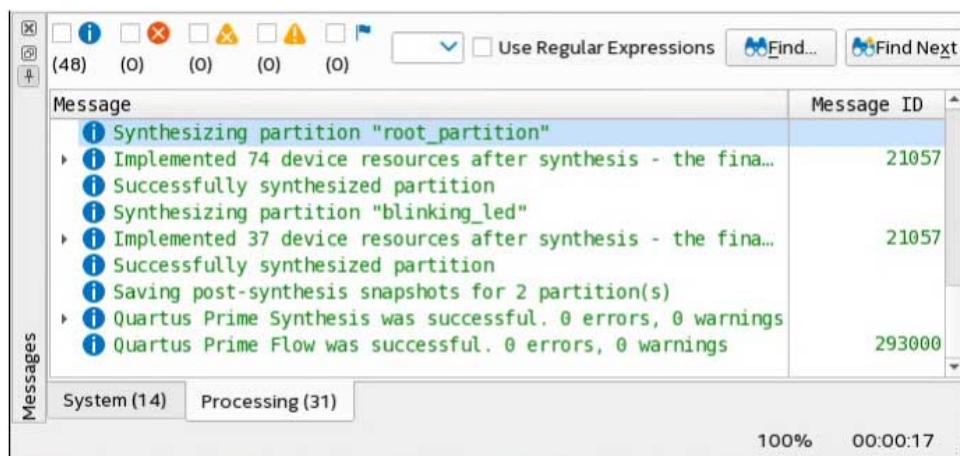
Review the detailed information in these the compilation reports to determine correct implementation. Right-click report data to locate and edit the source in project files.

Figure 17. Compilation Report

2.2.3. Viewing Project Messages

The **Messages** window (**View > 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
- **Find**—locates specific messages

Figure 18. Messages Window

2.2.3.1. Viewing Synthesis Warning Messages

Warning messages may contain hierarchies. In **Compilation Report > Synthesis > Messages** window, you can view hierarchical warning messages up to any level including the parent and child messages. For each message, you can view its source, file location, line number, and message ID by selecting appropriate column under **Message Column** (right-click on a message in the **Message** panel as shown in the following image and click **Message Column**).

Figure 19. **Synthesis Warning Messages (Two levels)**

Message	Source	File	Line	Message ID
① Synthesizing partition "root_partition"	Synthesis			286031
① Timing-Driven Synthesis is running on partition "root_partition"	Synthesis			21057
④ Implemented 48 device resources after synthesis - the final re:	Synthesis			
① Successfully synthesized partition	Synthesis			
① Synthesizing partition "blinking_led_top"	Synthesis			
④ Output pins are stuck at VCC or GND	Synthesis			13024
④ Pin "u_blinking_led_top value_top[0]" is stuck at VCC	Design Software	/nfs/site/...	33	13410
④ Pin "u_blinking_led_top value_top[1]" is stuck at VCC	Design Software	/nfs/site/...	33	13410
④ Pin "u_blinking_led_top value_top[2]" is stuck at VCC	Design Software	/nfs/site/...	33	13410
④ Pin "u_blinking_led_top value_top[3]" is stuck at VCC	Design Software	/nfs/site/...	33	13410
④ Implemented 6 device resources after synthesis - the final re:	Synthesis			21057
① Successfully synthesized partition	Synthesis			
① Saving post-synthesis snapshots for 2 partition(s)	Synthesis			
① Quartus Prime Synthesis was successful. 0 errors, 6 warnings	Synthesis			

System (14) Processing (28)

Figure 20. **Example of Synthesis Warning Messages With Three Levels**



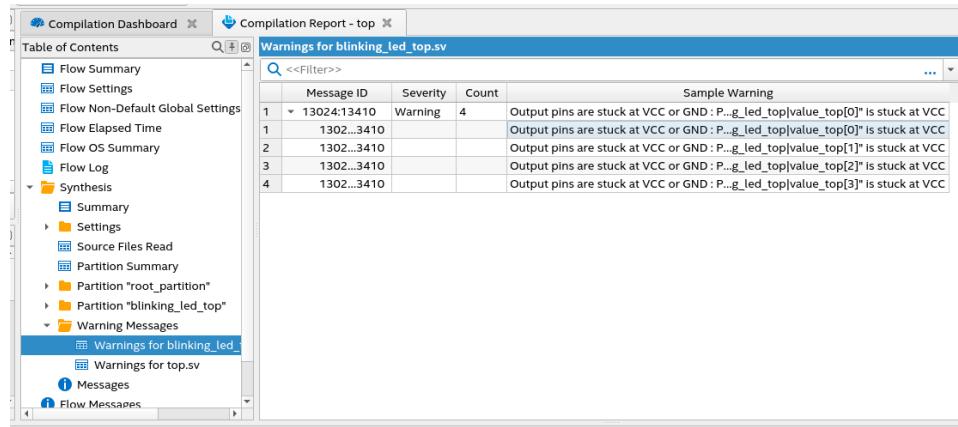
In **Compilation Report > Synthesis > Warning Messages**, you can view a comprehensive list of synthesis warning messages for each source file included in your design. You can view all child warning messages hidden within a parent warning message by expanding the collapsible rows. To view the location of each warning, perform these steps:

1. Right-click on the message.
2. Select the **Locate Node** option.
3. Select the desired tool to view the node.

Note:

In the source file-specific warning messages window, messages are hierarchical in nature and display up to three levels. If the warning messages go deeper than three levels, use the **Message (View > Messages)** window to view them.

In the source file-specific warning messages window, hierarchical messages are displayed with message IDs and sample warning messages that are a combination of the parent and child messages.

Figure 21. Synthesis Warning Messages for Each Source File

2.2.3.2. Suppressing Message Display

You can suppress display of unimportant messages from the Messages window, so that you can focus on the messages that are important to you. To suppress one or more messages from displaying in the Messages window, right-click the message, and then click any of the following commands:

- **Suppress Message**—suppresses all messages that match the exact text you specify.
- **Suppress Messages with Matching ID**—suppresses all messages that match the message ID number you specify, ignoring variables.
- **Suppress Messages with Matching Keyword**—suppresses all messages that match the keyword or hierarchy you specify.
- **Message Suppression Manager**—allows you to create and edit message suppression rules. You can define message suppression rules by message text, message ID number, or keyword.

Note:

- You cannot suppress error or Intel legal agreement messages.
- Suppressing a message also suppresses any submessages.
- A root message does not display if you suppress all of the root message's submessages.
- Message suppression is project revision-specific. Derivative project revisions inherit any suppression.
- You cannot edit messages or suppression rules during compilation.
- Messages are written to `stdout` when you use command-line executables.

Figure 22. Message Suppression Manager

Suppressing Messages by Design Entity

You can optionally suppress messages by design entity without modifying HDL. Entity-based message suppression can be helpful to eliminate insignificant warnings for specific IP components or design entities that may be obscuring other more important warnings.

To suppress messages by design entity, add the following line to the project .qsf, or to the .qip file for stand-alone IP components:

```
set_global_assignment -name MESSAGE_DISABLE -entity <name>
```

2.2.3.3. Promoting Critical Warnings to Errors

You can promote critical warnings to errors so that the compilation flow halts on receiving the critical warnings as it does with an errors. All critical warnings are supported.

You can only promote the message IDs on open projects.

1. In the **Message** dialog box, right-click on the critical warning you want to promote to an error.
2. Click **Message Promotion > Promote Critical Message ID to Error**
The software now treats the critical warning as an error.
3. To clear all promotions, click **Message Promotion > Clear All Message Promotions**
4. Alternatively, manually promote or demote a critical warning in the .qsf. For example:

```
set_global_assignment -name PROMOTE_WARNING_TO_ERROR 12677
```

2.3. 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 6. 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 Design	Quartus Database File (.qdb)
Partition database	Partition compilation results	Project > Export Design Partition	Partition Database File (.qdb)
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 (.ip)
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)

2.3.1. 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 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.

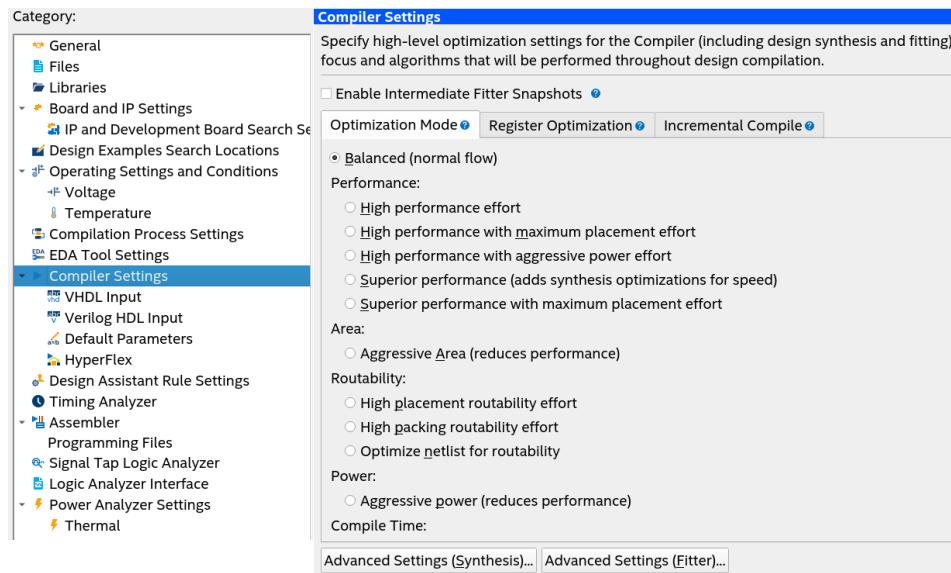
2.4. Managing Project Settings

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

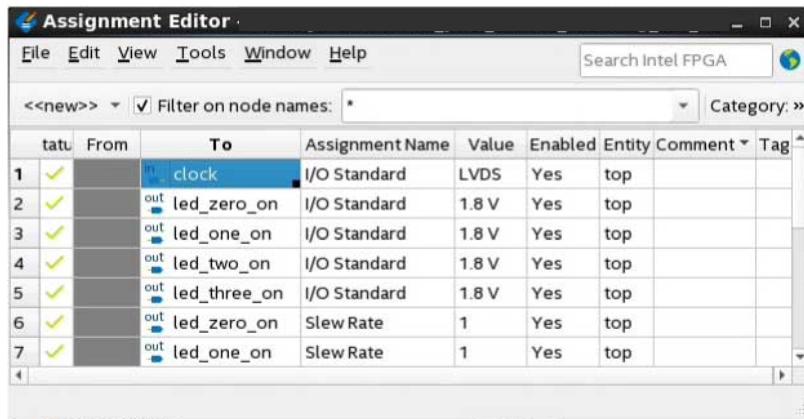
To access global project settings, click **Assignments > Settings**, or click **Settings** on the **Tasks** pane.

Figure 23. 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 24. Assignment Editor

2.4.1. Optimizing Project Settings

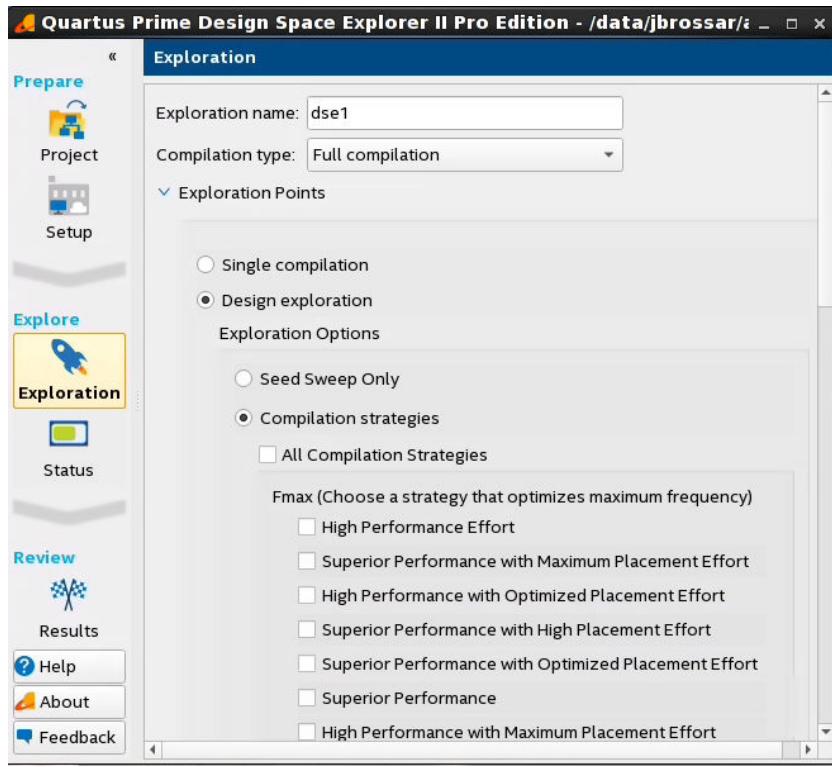
Optimize project settings to meet your design goals.

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.

2.4.1.1. 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 25. Design Space Explorer II


2.4.1.2. Optimize Settings with Project Revisions

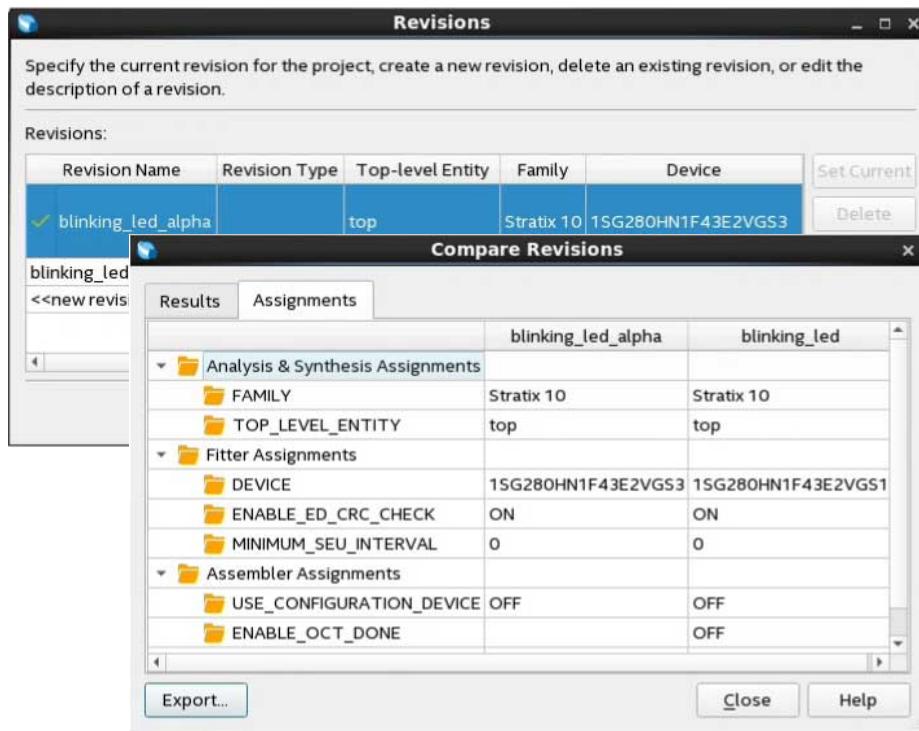
You can save multiple, named project revisions within your Intel Quartus Prime project (**Project > Revisions**). Each project revision captures a unique set of project settings and constraints, while using the same set of logic design files.

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 26. Comparing Project Revisions

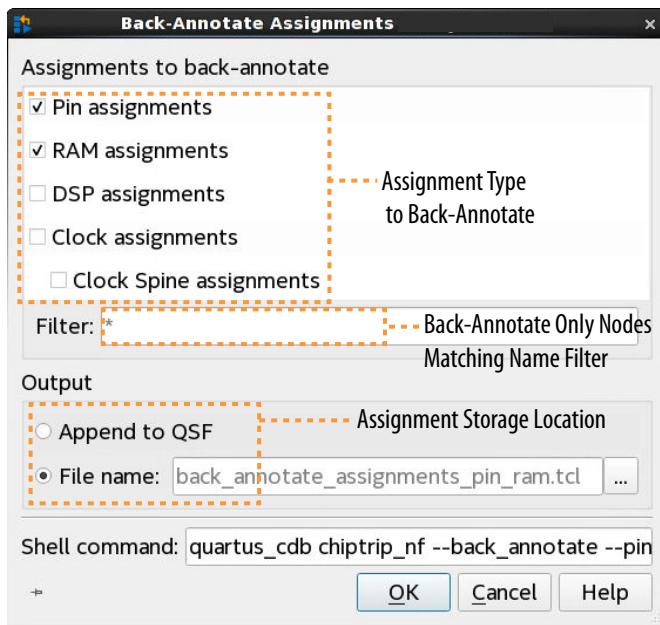
Related Information

[Project Revision Commands](#) on page 52

2.4.1.3. Back-Annotate Optimized 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 resource assignments to preserve that same implementation in subsequent compilations. Back-annotation can simplify timing closure by allowing you to lock down placement of your optimized results.

Locking down placement of large blocks related to Clocks, RAMs, and DSPs can produce higher f_{MAX} with less noise. Large blocks like RAMs and DSPs have heavier connectivity than regular LABs, complicating movement during placement. When a seed produces good results from suitable RAM and DSP placement, you can capture that placement with back-annotation. Subsequent compiles can then benefit from the high quality RAM and DSP placement from the good seed.

Figure 27. Back-Annotate Assignments Dialog Box


To back-annotate (copy) the device resource assignments from the last compilation to the project .qsf (or to a Tcl file) for use in the next compilation:

1. Run a full compilation, or run the Fitter through at least the **Place** stage.
2. Click **Assignments > Back-Annotate Assignments**.
3. Under **Assignments to back-annotate**, specify whether you want to preserve **Pin assignments**, **RAM assignments**, **DSP assignments**, **Clock assignments**, and **Clock Spine assignments** in the back-annotation.
4. In **Filter**, specify a text string (including wildcards) if you want to filter back-annotated assignments by entity name.
5. Under **Output**, specify whether to save the back-annotated assignments to the .qsf or to a Tcl file. A default Tcl file name displays.

Alternatively, you can run back-annotation with the following quartus_cdb executable. The **Shell command** field displays the shell command constructed by the options that you specify in the GUI.

```
quartus_cdb chiptrip_nf --back_annotate --pin --ram --dsp --clocks \
--spines --file "<file>.tcl"
```

Note:

Check available arguments by running quartus_cdb <project> --back_annotate --help.

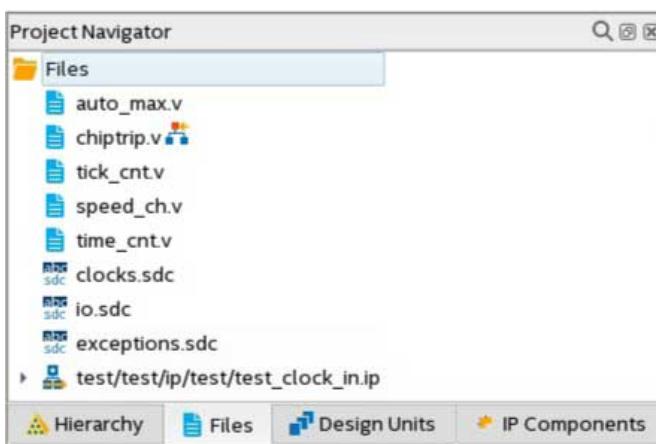
2.5. Managing Logic Design Files

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). 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 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 28. 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**

2.5.1. 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 47

2.5.2. Creating a Project Copy

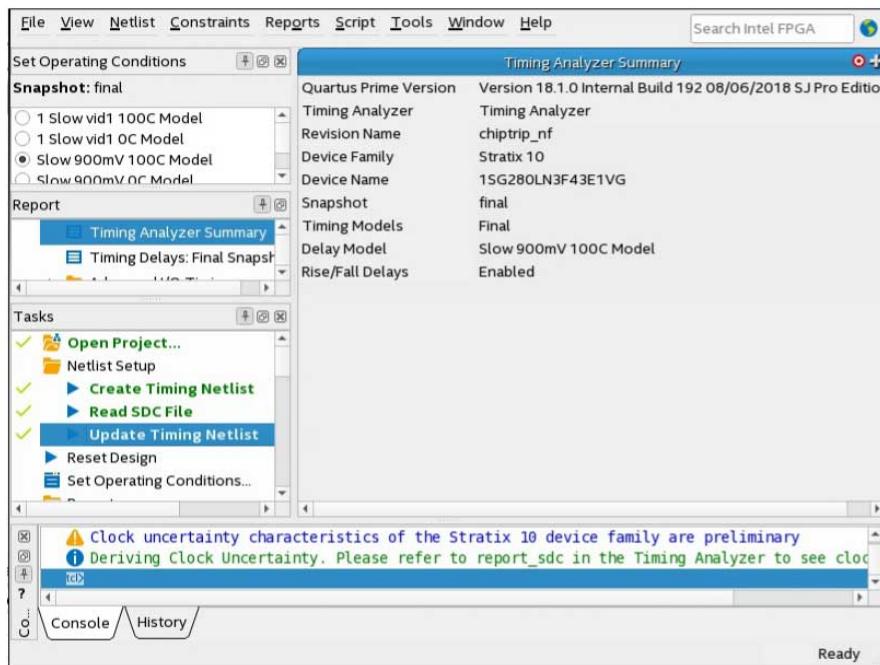
Click **Project > Copy Project** to create a separate copy of your project, rather than just a revision within the same project.

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.

2.6. 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.

Figure 29. Timing Analyzer



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.

2.7. Integrating Other EDA Tools

You can optionally integrate supported EDA synthesis, netlist partitioning, simulation, and signal integrity verification tools into the Intel Quartus Prime design flow.

The Intel Quartus Prime software supports input netlist files from supported EDA synthesis tools. The Compiler's EDA Netlist Writer module (quartus_eda) can automatically generate output files for processing in other EDA tools. The EDA Netlist Writer runs optionally as part of a full compilation, or you can run EDA Netlist Writer separately from the GUI or at the command line. The following functions are available to simplify EDA tool integration:

Table 7. EDA Tool Integration Functions

EDA Integration Task	EDA Integration Function
Specify settings for generation of output files for processing in other EDA tools.	Click Assignments > Settings > EDA Tool Settings to specify options for supported tools.
Generate output files for processing in other EDA tools.	Click Processing > Start > Start EDA Netlist Writer (or run quartus_eda) to generate files.
Compile RTL and gate-level simulation model libraries for your device, supported EDA simulators, and design language.	Click Tools > Launch Simulation Library Compiler to compile simulation libraries easily.
Generate EDA tool-specific setup scripts to compile, elaborate, and simulate Intel FPGA IP models and simulation model library files.	Specify options for Simulation file output when generating Intel FPGA IP with IP parameter editor.
Generate files that allow supported EDA tools to perform netlist modifications, such as adding new modules, partitioning the netlist, and changing module connectivity.	Use the quartus_eda -resynthesis command to generate a Verilog Quartus Mapping File (.vqm) that contains a node-level (or atom) representation of the netlist in standard structural Verilog RTL.
Include files generated by other EDA design entry or synthesis tools in your project as synthesized design files.	Click Project > Add/Remove Files In Project to add supported Design File files from other EDA tools.

Related Information

- [Intel Quartus Prime Pro Edition User Guide: Third-party Simulation](#)
- [Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis](#)
- [Intel Quartus Prime Pro Edition User Guide: PCB Design Tools](#)

2.8. Exporting Compilation Results

The Intel Quartus Prime Compiler writes the results to a set of database files. You can run a command to export the compilation results database as a single Quartus Database File (.qdb).

After running design compilation, the exported .qdb file contains the data to reproduce similar compilation 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. Although you cannot directly read the contents of the .qdb file after export, you can view attributes of the database file in the Quartus Database File Viewer.

Table 8. Exporting Compilation Results

To Export Compilation Results For	Method	Description
Complete Design	Click Project > Export Design	Saves compilation results for the current project revision in a version-compatible Quartus database file (.qdb) that you can import to another project or migrate to a later version of the Intel Quartus Prime software. You can export the results for the synthesized or final compilation snapshot. <i>Note:</i> Not supported for Intel Agilex devices.
Design Partition	Click Project > Export Design Partition	Saves compilation results for a design partition as a Partition Database File (.qdb) that you can import to another project using the same version of the Intel Quartus Prime software. You can export the results for the synthesized or final compilation snapshot.

Related Information

[Creating Database-Only Archives](#) on page 50

2.8.1. Exporting a Version-Compatible Compilation Database

You can export a project compilation database to a format that ensures version-compatibility with a later version of the Intel Quartus Prime software. The Intel Quartus Prime Pro Edition software version supports export of version-compatible databases for the following software versions and devices:

Table 9. Version-Compatible Compilation Database Support

The first table column indicates the first version to support version-compatible compilation database export for the specified devices.

Note:

- Database import supports two major versions back. For example, a database that you export from version 19.3, you can then import using version 19.3, 20.1, and 20.3. However, you cannot import version 19.3 to 21.1.
- You can export from any version that follows a supported version, if the version still supports the devices.

First Version with 'Export Design' Support	Intel Stratix 10 and Intel Agilex Devices	Intel Arria 10 and Intel Cyclone 10 GX Devices
18.0	No Support.	Supports all devices.
18.1	<ul style="list-style-type: none"> • 1SG250L • 1SG280H_S2 • 1SG280L • 1SG280L_S3 • 1SX250L • 1SX280L • 1SX280L_S3 	Supports all devices.
19.1	<ul style="list-style-type: none"> • 1SM16BH • 1SM21BH • 1SM16CH • 1SM21CH • 1SM21KH • 1SM16KH • 1SM21LH • 1SM16LH 	Supports all devices.
19.3	<ul style="list-style-type: none"> • 1SG10MH_U1 • 1SG10MH_U2 • 1ST250E • 1ST280E • 1SM16E • 1SM21E • 1ST165E • 1ST210E • 1SG166H • 1SG211H 	Supports all devices.
20.1	<ul style="list-style-type: none"> • 1SD280P • 1ST040E • 1ST085E • 1ST110E 	Supports all devices.
20.3	<ul style="list-style-type: none"> • 1SD21BP • 1SG040H • 1SX040H 	Supports all devices.
20.4	<ul style="list-style-type: none"> • 1SN21BH • 1SN21CE 	Supports all devices.

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 Design**. Select the **synthesized** or **final Snapshot**.

Figure 30. Export Design Dialog Box



4. Specify a name for the **Quartus Database File** to contain the exported results, and click **OK**.
5. To include the exported design's settings and constraint files, copy the **.qsf** and **.sdc** files to the import project directory.

2.8.2. 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 36 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. Click **Project > Import Design** and specify the **Quartus Database File**. To remove previous results, turn on **Overwrite existing project's databases**

Figure 31. Import Design Dialog Box

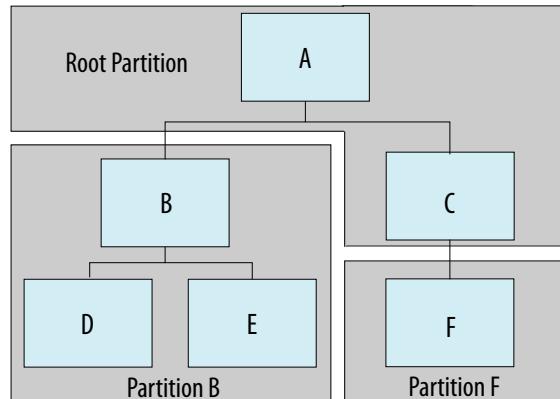


4. Click **OK**. When you compile the imported design, run only Compiler stages that occur after the stage the **.qdb** preserves, rather than running a full compilation. For example, if you import a version-compatible database that contains the synthesized snapshot, start compilation with the Fitter (**Processing > Start > Start Fitter**). If you import a version-compatible database that contains the final snapshot, start compilation with Timing Analysis (Signoff) (**Processing > Start > Start Timing Analysis (Signoff)**).

2.8.3. Creating a Design Partition

A design partition is a logical, named, hierarchical boundary that you can assign to an instance in your design. Defining a design partition allows you to optimize and lock down the compilation results for individual blocks. You can then optionally export the compilation results of a design partition for reuse in another context, such as reuse in another project.

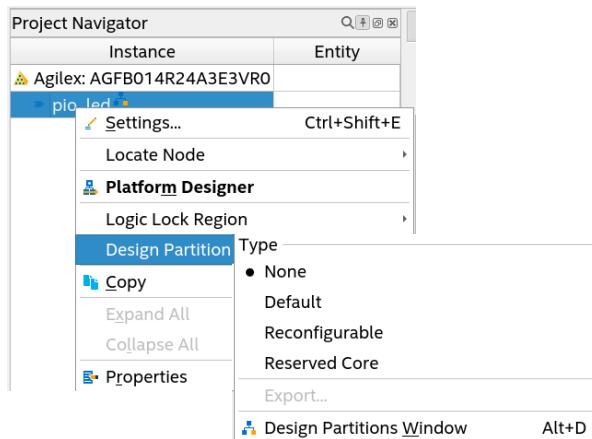
Figure 32. Design Partitions in Design Hierarchy



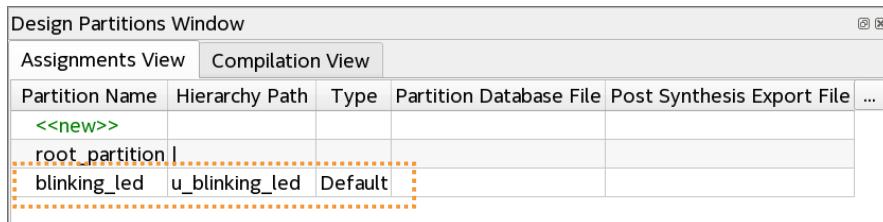
Follow these steps to create and modify design partitions:

1. In the Intel Quartus Prime software, open the project that you want to partition.
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. In the Project Navigator, right-click an instance in the **Hierarchy** tab, click **Design Partition > Set as Design Partition**.

Figure 33. Creating a Design Partition from the Project Hierarchy



4. To view and edit all design partitions in the project, click **Assignments > Design Partitions Window**.

Figure 34. Design Partitions Window


- Specify the properties of the design partition in the Design Partitions Window. The following settings are available:

Table 10. Design Partition Settings

Option	Description
Partition Name	Specifies the partition name. Each partition name must be unique and consist of only alphanumeric characters. The Intel Quartus Prime software automatically creates a top-level () "root_partition" for each project revision.
Hierarchy Path	Specifies the hierarchy path of the entity instance that you assign to the partition. You specify this value in the Create New Partition dialog box. The root partition hierarchy path is .
Type	Double-click to specify one of the following partition types that control how the Compiler processes and implements the partition: <ul style="list-style-type: none"> Default—Identifies a standard partition. The Compiler processes the partition using the associated design source files. Reconfigurable—Identifies a reconfigurable partition in a partial reconfiguration flow. Specify the Reconfigurable type to preserve synthesis results, while allowing refit of the partition in the PR flow. Reserved Core—Identifies a partition in a block-based design flow that is reserved for core development by a Consumer reusing the device periphery.
Preservation Level	Specifies one of the following preservation levels for the partition: <ul style="list-style-type: none"> Not Set—specifies no preservation level. The partition compiles from source files. synthesized—the partition compiles using the synthesized snapshot. final—the partition compiles using the final snapshot. With Preservation Level of synthesized or final , changes to the source code do not appear in the synthesis.
Empty	Specifies an empty partition that the Compiler skips. This setting is incompatible with the Reserved Core and Partition Database File settings for the same partition. The Preservation Level must be Not Set . An empty partition cannot have any child partitions.
Partition Database File	Specifies a Partition Database File (.qdb) that the Compiler uses during compilation of the partition. You export the .qdb for the stage of compilation that you want to reuse (synthesized or final). Assign the .qdb to a partition to reuse those results in another context.
Entity Re-binding	<ul style="list-style-type: none"> PR Flow—specifies the entity that replaces the default persona in each implementation revision. Root Partition Reuse Flow—specifies the entity that replaces the reserved core logic in the consumer project.
Color	Specifies the color-coding of the partition in the Chip Planner and Design Partition Planner displays.
Post Synthesis Export File	Automatically exports post-synthesis compilation results for the partition to the specified .qdb file each time Analysis & Synthesis runs. You can automatically export any design partition that does not have a preserved parent partition, including the root_partition.
Post Final Export File	Automatically exports post-final compilation results for the partition to the specified .qdb file each time the final stage of the Fitter runs. You can automatically export any design partition that does not have a preserved parent partition, including the root_partition.

2.8.4. Exporting a Design Partition

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

When you compile a design containing design partitions, the Compiler can preserve a synthesis or final snapshot of results for each partition. You can export the synthesized or final compilation results for individual design partitions with the **Export Design Partition** dialog box.

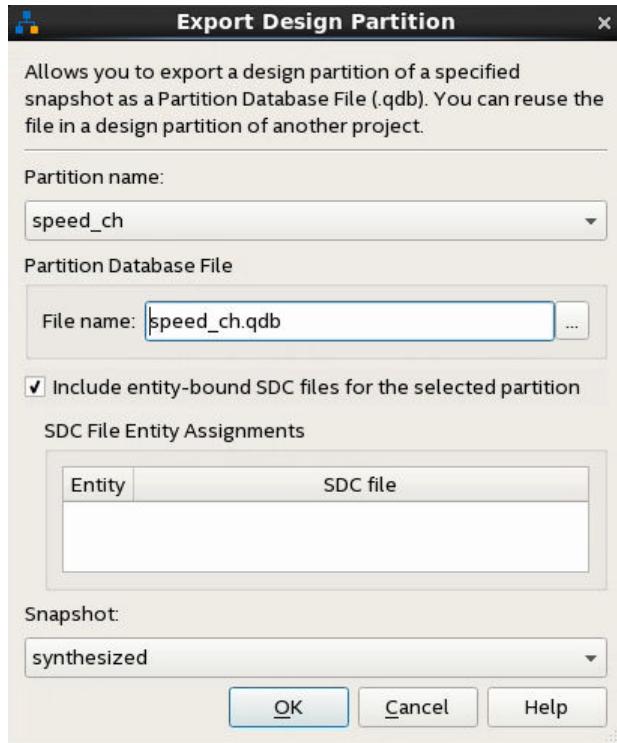
If the partition includes any entity-bound .sdc files, you can include those constraints in the .qdb. In addition, you can automate export of one or more partitions in the Design Partitions Window.

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. [Creating a Design Partition](#) on page 39 describes this process.
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 35. Export Design Partition Dialog Box



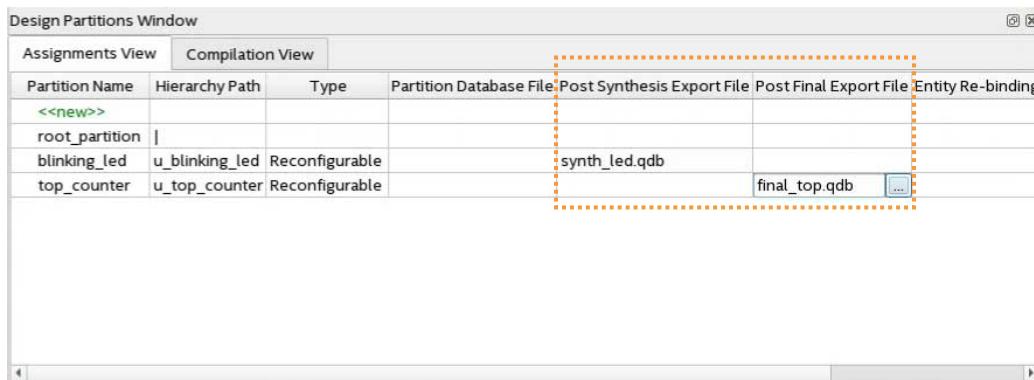
- Select the **Partition name** and the compilation **Snapshot** for export.
 - To include any entity-bound .sdc files in the exported .qdb, turn on **Include entity-bound SDC files for the selected partition**.
4. Click **OK**. The compilation results for the design partition exports to the file that you specify.

Automated Design Partition Export

Follow these steps to automatically export one or more design partitions following each compilation:

1. Open a project containing one or more design partitions. [Creating a Design Partition](#) on page 39 describes this process.
2. To open the Design Partitions Window, click **Assignments > Design Partitions Window**.
3. To automatically export a partition with synthesis results after each time you run synthesis, specify the a .qdb export path and file name for the **Post Synthesis Export File** option for that partition. If you specify only a file name without a path, the file exports to the output_files directory after compilation.
4. To automatically export a partition with final snapshot results each time you run the Fitter, specify a .qdb file name for the **Post Final Export File** option for that partition. If you specify only a file name without a path, the file exports to the output_files directory after compilation.

Figure 36. Specifying Export File in Design Partitions Window



.qsf Equivalent Assignment:

```
set_instance_assignment -name EXPORT_PARTITION_SNAPSHOT_<FINAL|SYNTHESIZED> \
    <hpath> -to <file_name>.qdb
```

Related Information

- [Intel Quartus Prime Pro Edition User Guide: Block-Based Design](#)
- [Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration](#)

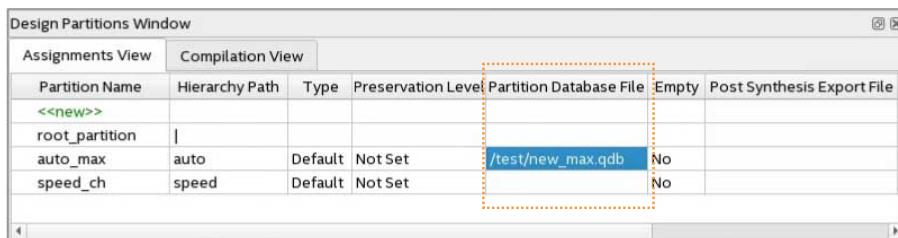
2.8.5. Reusing a Design Partition

You can reuse the compilation results of a design partition exported from another Intel Quartus Prime project. Reuse of a design partition allows you to share a synthesized or final design block with another designer. Refer to *Intel Quartus Prime Pro Edition User Guide: Block-Based Design* for more information about reuse of design partitions.

To reuse an exported design partition in another project, you assign the exported partition .qdb to an appropriately configured design partition in the target project via the Design Partition Window:

1. Export a design partition with the appropriate snapshot, as [Exporting a Design Partition](#) on page 41 describes.
2. Open the target Intel Quartus Prime project that you want to reuse the exported partition.
3. Click **Processing > Start > Start Analysis & Elaboration**.
4. Click **Assignments > Design Partitions Window**, and then create a design partition to contain the logic and compilation results of the exported .qdb.
5. Click the **Partition Database File** option for the new partition and select the exported .qdb file.

Figure 37. Partition Database File Setting in Design Partitions Window



Partition Name	Hierarchy Path	Type	Preservation Level	Partition Database File	Empty	Post Synthesis Export File
<<new>>						
root_partition						
auto_max	auto	Default	Not Set	/test/new_max.qdb	No	
speed_ch	speed	Default	Not Set		No	

6. Specify any other properties of the design partition in the Design Partitions Window. The Compiler uses the partition's assigned .qdb as the source.

2.8.6. Viewing Quartus Database File Information

Although you cannot directly read a .qdb file, you can view helpful attributes about the file to quickly identify its contents and suitability for use.

The Intel Quartus Prime software automatically stores metadata about the project of origin when you export a Quartus Database File (.qdb). You can then use the Quartus Database File Viewer to display the attributes of any of these .qdb files.

Figure 38. Quartus Database File Viewer

Project Summary:	
Attribute	Value
Project Information	
Contents	Partition
Date	Thu Aug 16 10:37:40 2018
Device	1SG280HN1F43E2VGS1
Entity	blinking_led
Family	Stratix 10
Partition Name	blinking_led
Revision Name	blinking_led
Revision Type	Unspecified
Snapshot	synthesized
Version	18.1.0
Version-Compatible	No
Resource Utilization	
Average fan-out	0.16
Combinational ALUT usage for logic	0
Dedicated logic registers	2
Estimate of Logic utilization (ALMs needed)	1
I/O pins	35
Maximum fan-out	2
Maximum fan-out node	u_blinking_led counter[23]
Total DSP Blocks	0
Total fan-out	6

Follow these steps to view the attributes of a .qdb file:

1. In the Intel Quartus Prime software, click **File > Open**, select **Design Files for Files of Type**, and select a .qdb file.
2. Click **Open**. The Quartus Database File Viewer displays project and resource utilization attributes of the .qdb.

Alternatively, run the following command-line equivalent:

```
quartus_cdb --extract_metadata --file <archive_name.qdb> \
--type quartus --dir <extraction_directory> \
[--overwrite]
```

2.8.6.1. QDB File Attribute Types

The Quartus Database Viewer can display the following attributes of a .qdb file:

Table 11. QDB File Attributes

QDB Attribute Types	Attribute	Example
Project Information	Contents	Partition
	Date	Thu Jan 23 10:56:23 2018
	Device	10AX016C3U19E2LG
	Entity (if Partition)	Counter
	Family	Arria 10
	Partition Name	root_partition

continued...

	Revision Name	Top
	Revision Type	PR_BASE
	Snapshot	synthesized
	Version	18.1.0 Pro Edition
	Version-Compatible	Yes
Resource Utilization (exported for partition QDB only)	For synthesized snapshot partition lists data from the Synthesis Resource Usage Summary report.	Average fan-out:16 Dedicated logic registers:14 Estimate of Logic utilization:1 I/O pins:35 Maximum fan-out:2 Maximum fan-out node:counter[23] Total DSP Blocks:0 Total fan-out:6 ...
	For the final snapshot partition, lists data from the Fitter Partition Statistics report.	Average fan-out:.16 Combinational ALUTs: 16 I/O Registers M20Ks ...

2.8.7. 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 39. Clean Project Dialog Box Cleans the Project Database



2.9. Migrating Projects Across Operating Systems

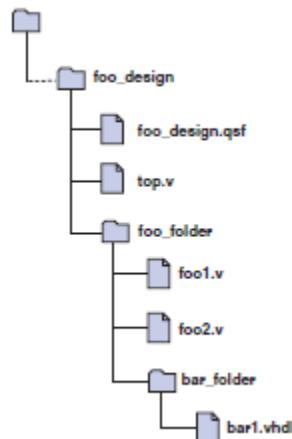
Consider the following cross-platform issues when moving your project from one operating system to another (for example, from Windows* to Linux*).

2.9.1. 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 40. All Inclusive Project Directory Structure

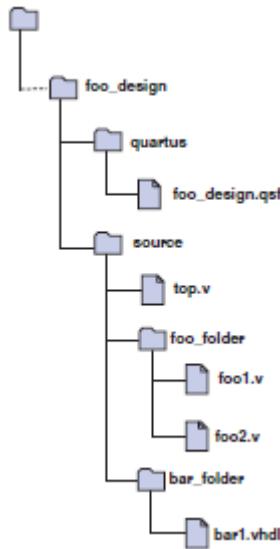


2.9.1.1. Use Relative Paths

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 41. Intel Quartus Prime Project Directory Separate from Design Files



2.9.2. 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:\`

2.10. 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 RTL 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.

Related Information

[Project Archive Commands](#) on page 52

2.10.1. Manually Adding Files To Archives

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

Note: If preserving a custom component as part of an Intel Quartus Prime Archive (.qar), you must first explicitly add the component _hw.tcl file to the project to ensure that the .qar includes the component. Click **Project > Add/Remove Files in Project** to add files to your project.

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**.

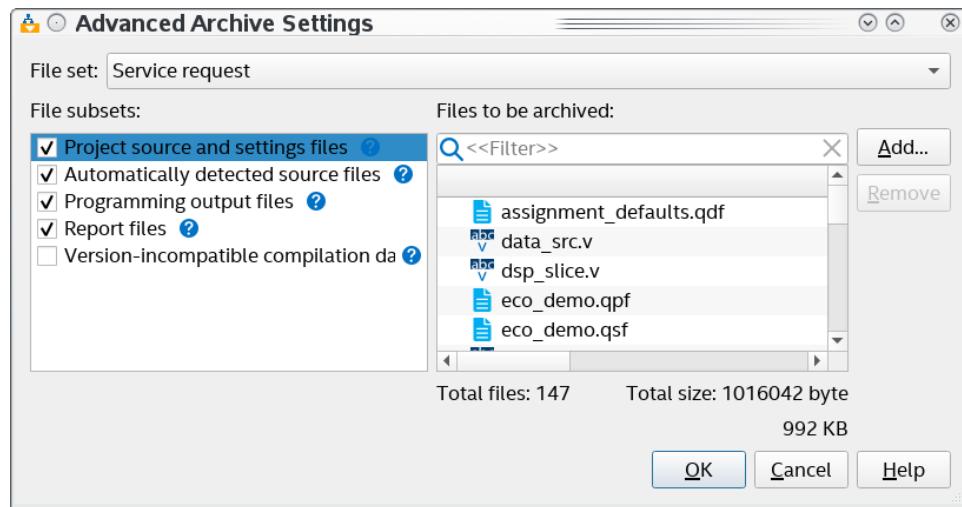
2.10.2. Archiving Projects for Service Requests

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)
4. Click **OK**, and then click **Archive**.

Figure 42. Archiving Project for Service Request



2.10.3. Archiving Projects for External Revision Control

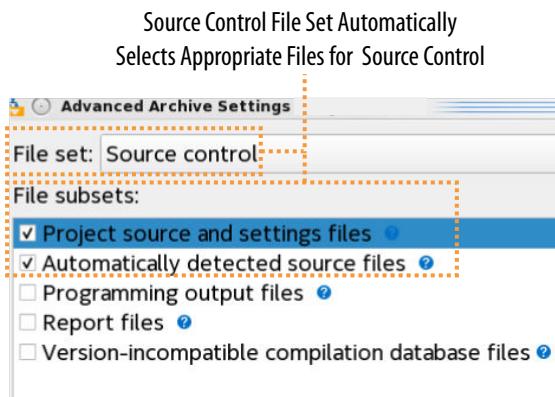
Your project may involve different team members with distributed responsibilities, such as sub-module design, device and system integration, simulation, and timing closure. In such cases, it may be useful to track and protect file revisions in an external revision control system.

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.

2.10.3.1. Project Files to Include In External Revision Control

When archiving Intel Quartus Prime projects for external source control, The **Source control** setting in **Advanced Archive Settings** dialog box is preset to include all appropriate file types for source control automatically.

Figure 43. Advanced Archive Settings Dialog Box



Include the following file types when archiving projects for external revision control:

Table 12. Project Files to Include In External Revision Control (Included Automatically with 'Source Control' Archive Setting)

File Type	Description
Intel Quartus Prime project setting and assignment files	<ul style="list-style-type: none"> Intel Quartus Prime Project Files (.qpf) Intel Quartus Prime Settings Files (.qsf) Intel Quartus Prime Pin Planner File (.ppf)
Timing constraint files	Synopsys Design Constraint Files (.sdc)
Design files	<ul style="list-style-type: none"> Verilog HDL Design Files (.v) SystemVerilog Design Files (.sv) VHDL Design Files (.vhd) Block Diagram/Schematic Design Files (.bdf) Block Symbol Files (.bsf) Verilog Quartus Mapping Files (.vqm) Platform Designer System Files (.qsys) State Machine Editor Files (.smf) Tcl Script Design Files (.tcl)
System and IP files	<ul style="list-style-type: none"> IP variation file (.ip) Verilog IP design files (.v) SystemVerilog IP design files (.sv) VHDL IP design files (.sv) VHDL Component Declaration Files (.cmp) Intel Quartus Prime IP file (.qip) Intel Quartus Prime Simulation IP File (.sip) Platform Designer System Files (.qsys) Platform Designer connection and parameterization files (.sopcinfo) IP upgrade status files (.csv) IP synthesis parameters files (.qgsynthc) IP simulation parameters files (.qgsimc) Platform Designer system exported as (.tcl).
EDA tool integration files	<ul style="list-style-type: none"> Verilog HDL Output Files (.vo) VHDL Output Files (.vho) VHDL simulation model files (.vhd) Verilog HDL simulation model files (.v) Simulation library files (cds.lib, hdl.var) Simulation setup scripts (_setup.sh, .tcl, .spd, .txt)

2.10.4. Creating Database-Only Archives

If your project contains sensitive RTL that you do not want to share with Intel support, you can create a *database-only* archive. A database-only archive contains only the minimum files required to run timing analysis, fitter, assembler, and GUI design-inspection tools like Chip Planner.

A database-only archive includes only the project Quartus databases and any additional files required for compilation. It does not include RTL files.

You can review the complete list of files included in the archive in a report generated when you create a database-only archive.

Security Note: A database-only archive does not guarantee protection for sharing your design without sharing your RTL. The RTL Netlist Viewer, Technology Map Viewer, and other views, along with the EDA Netlist Writer, are still available for projects exported using this feature.

With some effort, the original content can be reverse engineered.

To disable these features, encrypt your design. For details, see [Support for the IEEE 1735 Encryption Standard](#) on page 113.

Before creating a database-only archive, your project must have completed one of the following compilation stages:

- **Synthesis**

Archives that are created after running Synthesis can be used to run the fitter and then complete timing analysis, run the assembler, and use GUI-based design-inspection tools.

- **Finalized**

Archives that are created after completing a full compilation flow for your project can be used to complete timing analysis, run the assembler, and use GUI design-inspection tools.

To create a database-only archive, run the following command:

```
quartus_sh --archive_database -project <project_file> [-use_final_db]
```

Specify the `-use_final_db` option to create a database-only archive based on the finalized snapshot of your project. Otherwise, a database-only archive based on the synthesized snapshot is created.

The command generates two files: a `.qar` file that contains the database-only archive, and a `.contents.txt` file that lists files that are included in the `.qar` file.

You can get command syntax details by running the following command:

```
quartus_sh --help=archive_database
```

Related Information

[Support for the IEEE 1735 Encryption Standard](#) on page 113

2.11. 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>.`

2.11.1. Project Revision Commands

create_revision Command

create_revision defines the properties of a new project revision.

```
create_revision <name> -based_on <revision_name> -set_current -new_rev_type \
<rev_type> -root_partition_qdb_file <root qdb>
```

Table 13. 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.
-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 30

2.11.2. Project Archive Commands

project_archive Command

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

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

Table 14. project_archive Command Options

Options	Description
-all_revisions	Includes all revisions of the current project in the archive.
-common_directory /<name>	Preserves original project directory structure in specified subdirectory.

continued...

Options	Description
-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.

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 47

2.11.3. Project Database Commands

export_database Command

export_design exports the specified project database to the .qdb file you specify.

These commands require the quartus_cdb executable.

```
quartus_cdb <revision name> --export_design --file <file name>.qdb \
--snapshot <synthesized/final>
```

import_database Command

import_design imports the specified project database to the .qdb file you specify.

```
quartus_cdb <revision name> --import_design --file <file name>.qdb
```

export_block Command

export_block exports the specified partition database to the .qdb file you specify.

```
quartus_cdb -r <project name> -c <revision name> --export_block \
<partition name> --snapshot <name> --file <file name>.qdb
```

2.11.3.1. quartus_cdb Executables to Manage Version-Compatible Databases

The command-line arguments to the quartus_cdb executable in the Quartus Prime Pro software are export_design and import_design. The exported version-compatible design files are archived in a file (with a .qdb extension). This differs from the Intel Quartus Prime Standard Edition software, which writes all files to a directory.

In the Intel Quartus Prime Standard Edition software, the flow exports both post-map and post-fit databases. In the Intel Quartus Prime Pro Edition software, the export command requires the snapshot argument to indicate the target snapshot to export. If the specified snapshot has not been compiled, the flow exits with an error. In ACDS 16.0, export is limited to "synthesized" and "final" snapshots.

```
quartus_cdb <project_name> [-c <revision_name>] --export_design
--snapshot <snapshot_name> --file <filename>.qdb
```

The import command takes the exported *.qdb file and the project to which you want to import the design.

```
quartus_cdb <project_name> [-c <revision_name>] --import_design
--file <archive>.qdb [--overwrite] [--timing_analysis_mode]
```

The --timing_analysis_mode option is only available for Intel Arria 10 designs. The option disables legality checks for certain configuration rules that may have changed from prior versions of the Intel Quartus Prime software. Use this option only if you cannot successfully import your design without it. After you have imported a design in timing analysis mode, you cannot use it to generate programming files.

2.12. Managing Projects Revision History

Document Version	Intel Quartus Prime Version	Changes
2022.12.12	22.4	<ul style="list-style-type: none"> Revised <i>Creating a New FPGA Design Project</i> for board-aware features. Added <i>Using the Board-Aware Flow</i> topic. Added <i>Creating a New Project from a Design Example</i> topic. Added <i>Family, Device & Board Settings</i> topic. Added <i>Accessing Pre-Installed Design Examples</i> topic. Added <i>Accessing Online Design Examples</i> topic. Added <i>Accessing Downloaded Design Examples</i> topic. Added <i>Internet Connectivity Options</i> topic. Added <i>Design Examples Options</i> topic. Added <i>Specifying a Target Board for the Project</i> topic.
2022.03.28	22.1	<ul style="list-style-type: none"> Removed references to obsolete Advisors from <i>Optimizing Project Settings</i> topic. Added <i>Viewing Synthesis Warning Messages</i> topic. Removed the topic <i>Automated Problem Reports</i>.
2021.06.21	21.2	<ul style="list-style-type: none"> Added <i>Version-Compatible Compilation Database Support</i> table. Added "Promoting Critical Warnings to Errors" topic.
2021.03.29	21.1	<ul style="list-style-type: none"> Added "Creating Database-Only Archives" topic. Added "Promoting Critical Warnings to Errors" topic
2020.09.28	20.3	<ul style="list-style-type: none"> Updated "Back-Annotate Optimized Assignments" for support of pins, clocks, RAMs, and DSPs.
2020.05.01	20.1	<ul style="list-style-type: none"> Added note about .qar file requirements to "Design Guidelines for Component Instances" topic.
2019.09.30	19.3	<ul style="list-style-type: none"> Added "Disabling Automated Problem Reports" topic. Added "Suppressing Messages" topic.
2018.09.24	18.1	<ul style="list-style-type: none"> Subdivided "Exporting, Archiving, and Migrating Projects" into separate sections. Described migration of full chip database in "Exporting a Version-Compatible Compilation Database" topic. Described automated .qdb partition export in "Exporting a Design Partition" topic. Added "Viewing Quartus Database File Information" topic. Added "Specifying the Target Device or Board" topic.

continued...

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"> Divided "Introduction to Intel FPGA IP Cores" into separate chapter. Moved "IP Core Best Practices" topic to <i>Introduction to Intel FPGA IP Cores</i> chapter. Moved "Factors Affecting Compilation Results" topic to <i>Design Compilation: Intel Quartus Prime Pro Edition User Guide</i>.
2018.05.07	18.0.0	<ul style="list-style-type: none"> Initial release as chapter of <i>Getting Started User Guide</i>. Revised "Exporting a Design Partition" topic to add Include entity-bound SDC files for the selected partition option, to add prerequisite steps, and to remove import step covered in separate topic. Changed title of "Managing Team-Based Designs" to "Exporting, Archiving, and Migrating Projects" and updated content. Changed title of "Migrating Compilation Results Across Software Versions" to "Exporting the Compilation Database" and updated content. Changed title of "Exporting the Results Database" to "Exporting a Version-Compatible Design Compilation Database" and updated content. Changed title of "Importing the Results Database" to "Importing a Version-Compatible Design Compilation Database" and updated content. Changed title of "Cleaning the Project Database" to "Cleaning the Project Compilation Database." Updated screenshots of IP Catalog and Parameter Editor for latest IP names.

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> Revised product branding for Intel standards. Revised topics on Intel FPGA IP Evaluation Mode (formerly OpenCore). Removed -compatible attribute from <code>export_design</code> command content. Updated figure: IP Upgrade Alert in Project Navigator. Updated IP Core Upgrade Status table with new icons, and added row for IP Component Outdated status.
2017.05.08	17.0.0	<ul style="list-style-type: none"> Added Project Tasks pane and update New Project Wizard. Updated Compilation Dashboard image to show concurrent analysis. Removed Smart Compilation option from Settings dialog box screenshot. Updated IP Catalog screenshots for latest GUIs. Added topic on Back-Annotate Assignments command. Added Exporting a Design Partition topic. Removed mentions to deprecated Incremental Compilation. Added reference to Block-Level Design Flows.
2016.10.31	16.1.0	<ul style="list-style-type: none"> Added references to compilation stages and snapshots. Removed support for comparing revisions. 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. Added Support for IP Core Encryption topic. Rebranding for Intel

continued...

Date	Version	Changes
2016.05.03	16.0.0	<ul style="list-style-type: none"> Removed statements about serial equivalence when using multiple processors. Added the "Preserving Compilation Results" section. Added the "Migrating Results Across Quartus Prime Software" section and its subsections for information about importing and exporting compilation results between different versions of Quartus Prime. Added the "Project Database Commands" section and its subsections.
2016.02.09	15.1.1	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Added Generating Version-Independent IP Simulation Scripts topic. Added example IP simulation script templates for supported simulators. Added Incorporating IP Simulation Scripts in Top-Level Scripts topic. Added Troubleshooting IP Upgrade topic. Updated IP Catalog and parameter editor descriptions for GUI changes. Updated IP upgrade and migration steps for latest GUI changes. Updated Generating IP Cores process for GUI changes. Updated Files Generated for IP Cores and Qsys system description. Removed references to devices and features not supported in version 15.1. Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
2015.05.04	15.0.0	<ul style="list-style-type: none"> 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. 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Replaced MegaWizard Plug-In Manager information with IP Catalog. Added standard information about upgrading IP cores. Added standard installation and licensing information. Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor.
November 2013	13.1.0	<ul style="list-style-type: none"> Conversion to DITA format
May 2013	13.0.0	<ul style="list-style-type: none"> Overhaul for improved usability and updated information.
<i>continued...</i>		

Date	Version	Changes
June 2012	12.0.0	<ul style="list-style-type: none">Removed survey link.Updated information about VERILOG_INCLUDE_FILE.
November 2011	10.1.1	Template update.
December 2010	10.1.0	<ul style="list-style-type: none">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.

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 or Board

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.

Alternatively, you can create a system that targets a specific development board, rather than only targeting a specific FPGA device. When you target a specific development board, the Intel Quartus Prime software is *aware* of the target board (board-aware) which accelerates the process of appropriately configuring, connecting, and validating IP for the target board. Refer to [Using the Board-Aware Flow](#) on page 10 for details.

Device and Board 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 documentation for detailed device characteristics. View a summary of each device's resources by selecting a device in the **Device** dialog box (**Assignments** ➤ **Device**)
- 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

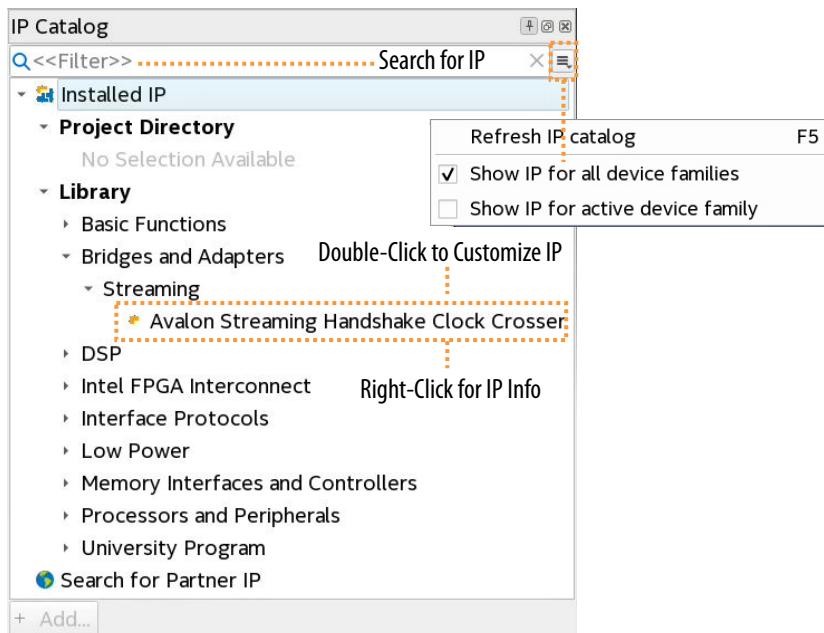
Determine whether you want to migrate your design to another device density to allow flexibility when your design nears completion. You may want to target a smaller (and less expensive) device and then move to a larger device if necessary to meet your design requirements. Other designers may prototype 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 44. 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 75 for general information on using Intel FPGA IP cores.

Related Information

- [Introduction to Intel FPGA IP Cores](#) on page 75
- [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.

You can use the Intel Quartus Prime Interface Planner to help you accurately plan constraints for design implementation. Use Interface Planner to prototype interface implementations and rapidly define a legal device floorplan.

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—allow a component to use an address-mapped read or write protocol that connects master components to slave components.
- Avalon streaming—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.

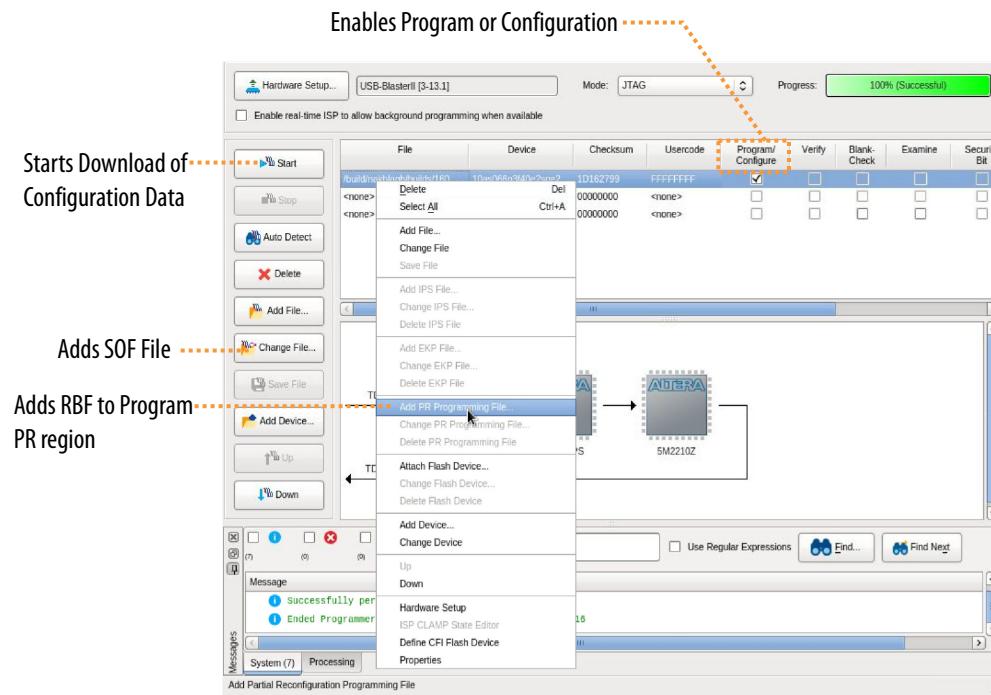
Related Information

[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 45. 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.

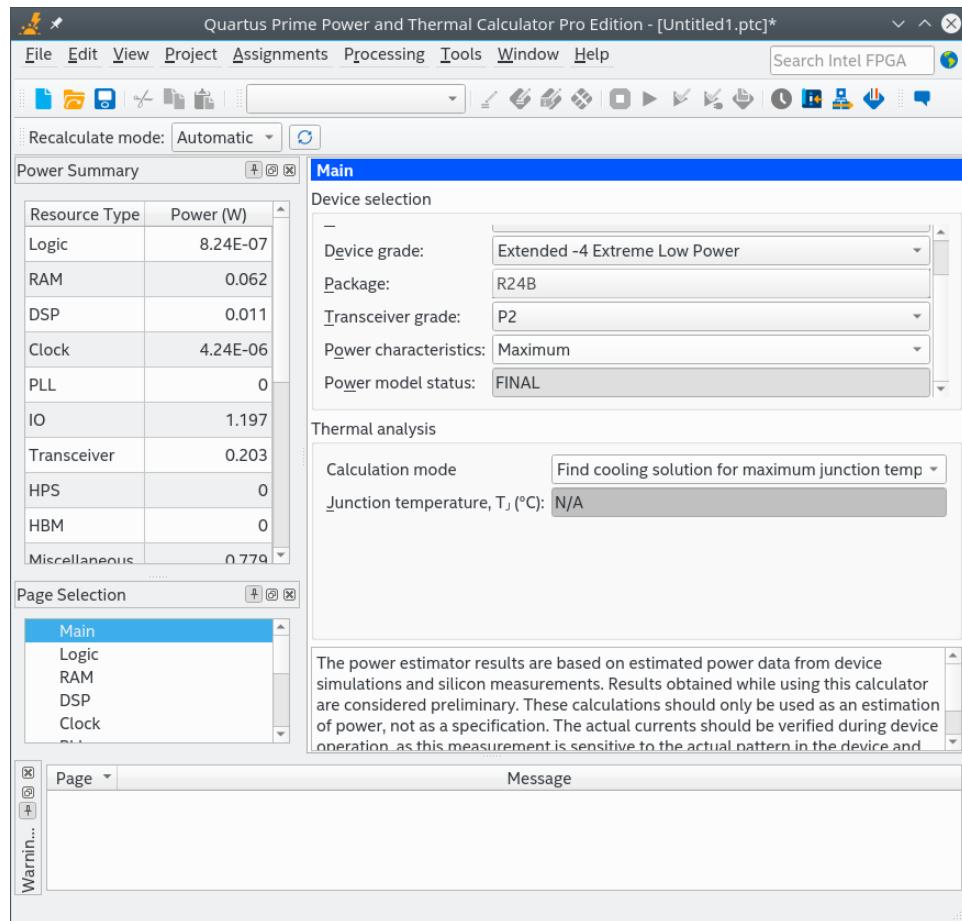
Power and Thermal Calculator

The Power and Thermal Calculator (PTC) is a standalone tool that allows you to estimate power utilization for your design for Intel Stratix 10 and Intel Agilex device families.

Note: PTC does not support older devices, such as the Intel Arria 10 and Intel Cyclone 10 families, for which you must use the corresponding Early Power Estimator.

Estimating power consumption early in the design cycle allows you to plan power budgets and avoid unexpected results when designing the PCB.

Figure 46. Power and Thermal Calculator



You can enter data in PTC manually or through the output from an Intel Quartus Prime software compile.

- To manually enter data into PTC, enter the device resources, operating frequency, toggle rates, IP usage, 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 PTC manually.
- If you have an existing design or a partially completed design, you can use the Intel Quartus Prime software to generate a .qptc file, which you can then import into PTC.

For additional information about PTC, refer to the [Intel FPGA Power and Thermal Calculator User Guide](#).

Early Power Estimator (EPE) Spreadsheet

For devices other than Intel Stratix 10 and Intel Agilex, 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 47. Early Power Estimator (EPE) Spreadsheet

[Visit the Online Power Management Resource Center](#)

Early Power Estimator

Cyclone® 10 GX

Version 18.0.1, Build 06.28

Input Parameters		Thermal Power (W)	
Family	Cyclone 10 GX	Logic	0.000
Device	10CX085	RAM	0.000
Device Grade	Extended -5	DSP	0.000
Package	F672	Clock	0.000
Transceiver Grade	N/A	PLL	0.000
Power Characteristics	Typical	I/O	0.000
V _{CC} Voltage (mV)	900	XCVR	0.000
Power Model Status	FINAL	HPS	0.000
Junction Temp, T _J	Auto Compute	P _{STATIC}	0.718
Ambient Temp, T _A (°C)	25	TOTAL (W)	0.718
Cooling Solution	SmartVID Power Savings		
θ _{JA} Junction-Ambient	Intel recommends using Intel® Empirion® Power Solutions with Intel® FPGAs		
Board Thermal Model	Typical	Thermal Analysis	
θ _{JB} Junction-Board	2.9	Junction Temp, T _J (°C)	26.0
Board Temp, T _B (°C)	25	Maximum Allowed T _A (°C)	96.2

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.

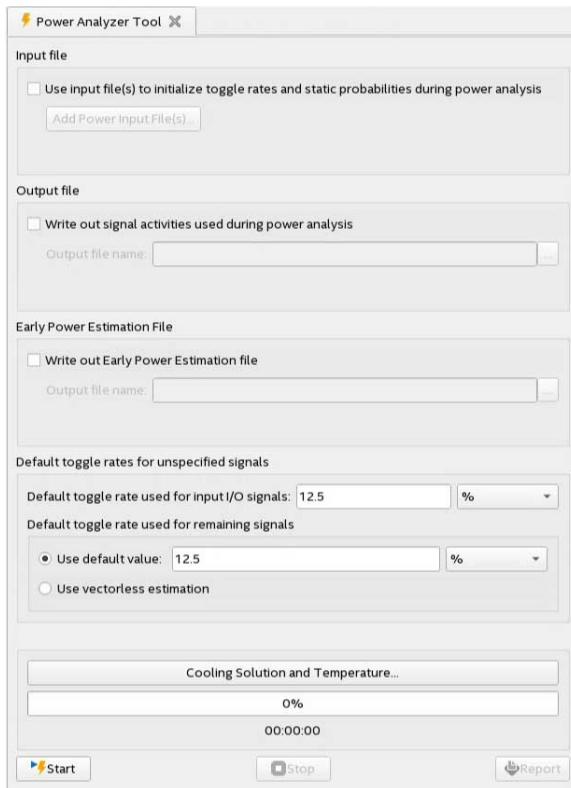
If you have an existing design or a partially completed design, you can use the Intel Quartus Prime software to generate the Early Power Estimator File (.txt, .csv) to assist you in completing the EPE spreadsheet.

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.

Figure 48. Intel Quartus Prime Power Analyzer



Related Information

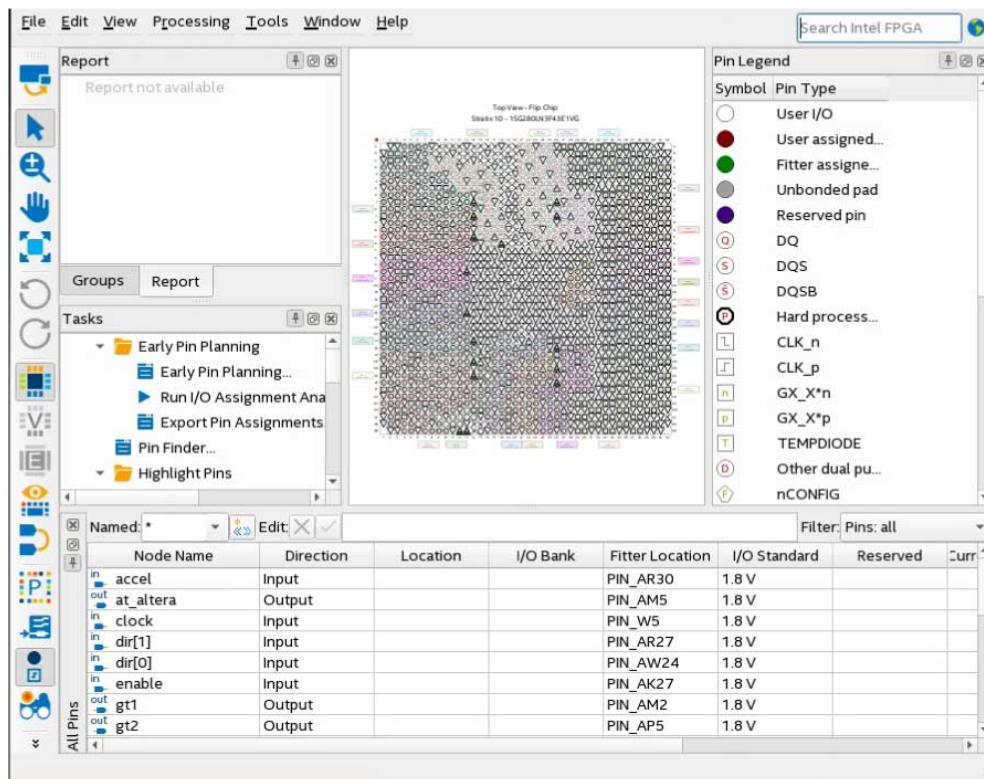
[Intel Quartus Prime Edition User Guide: Power Analysis and Optimization](#)

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.

Figure 49. Pin Planner



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.

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 Pro Edition User Guide: Design Optimization](#)
For more information about I/O assignment and analysis.

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.

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

The Intel Quartus Prime software does not include a native simulator, but provides support for RTL- and gate-level design simulation in specific, supported EDA simulators.

In addition, Intel provides the Questa - Intel FPGA Edition simulator along with the Intel Quartus Prime software. The Intel Quartus Prime software includes precompiled libraries for the Questa - Intel FPGA Edition simulator.

Alternatively you can purchase a full licenses for other supported third-party simulators 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.

Related Information

["Simulator Support," Intel Quartus Prime Pro Edition User Guide: Third-Party Simulation](#)

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), 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.

Table 15. 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 Pro 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.

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 Pro 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 Pro 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 use block-based design to preserve portions of previous placement and routing in subsequent compilations. Block based design practices can reduce your total compilation time in a partitioned design if you make small changes to other portions of your design.

3.13. Design Planning Revision History

Document Version	Intel Quartus Prime Version	Changes
2022.12.12	22.4	<ul style="list-style-type: none"> Updated <i>Plan for the Target Device or Board</i> topic for board-aware features.
2022.06.20	22.2	<ul style="list-style-type: none"> Removed obsolete <i>Simultaneous Switching Noise Analysis</i> topic from this basic discussion.
2022.03.28	22.1	<ul style="list-style-type: none"> Added information about Power and Thermal Calculator in <i>Plan for Device Power Consumption</i>.
2021.10.04	21.3	<ul style="list-style-type: none"> Added support for Questa Intel FPGA Edition simulator. Removed support for ModelSim - Intel FPGA Edition simulator.
2018.09.24	18.1.0	<ul style="list-style-type: none"> Moved information about specifying the target board to "Specifying the Target Device or Board" in <i>Managing Projects</i> 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 <i>Managing Projects</i> 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, <i>Third-Party Simulation User Guide</i> Retitled "Planning for Device Programming or Configuration" to "Plan for Device Programming" 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." Retitled <i>Design Planning with the Intel Quartus Prime Software</i> to <i>Design Planning</i>
2018.05.07	18.0	Initial release as separate chapter of <i>Getting Started User Guide</i> .

Date	Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"> Changed instances of OpenCore Plus to Intel FPGA IP Evaluation Mode. Changed instances of Qsys to Platform Designer (Standard)
2017.05.08	17.0.0	<ul style="list-style-type: none"> Removed mentions to Integrated Synthesis.
2016.10.31	16.1.0	<ul style="list-style-type: none"> Implemented Intel rebranding.
2016.05.03	16.0.0	Added information about Development Kit selection.
2015.11.02	15.1.0	<ul style="list-style-type: none"> Added references to Interface Planning chapter. Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
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.

continued...

Date	Version	Changes
November, 2012	12.1.0	Update for changes to early pin planning feature
June 2012	12.0.0	Editorial update.
November 2011	11.0.1	Template update.
May 2011	11.0.0	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

continued...

Date	Version	Changes
		<ul style="list-style-type: none">• Updated information on "Planning for On-Chip Debugging Options" on page 1-10• 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	<ul style="list-style-type: none">• No change to content
November 2008	8.1.0	<ul style="list-style-type: none">• Changed to 8-1/2 x 11 page size. No change to content.
May 2008	8.0.0	<ul style="list-style-type: none">• 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

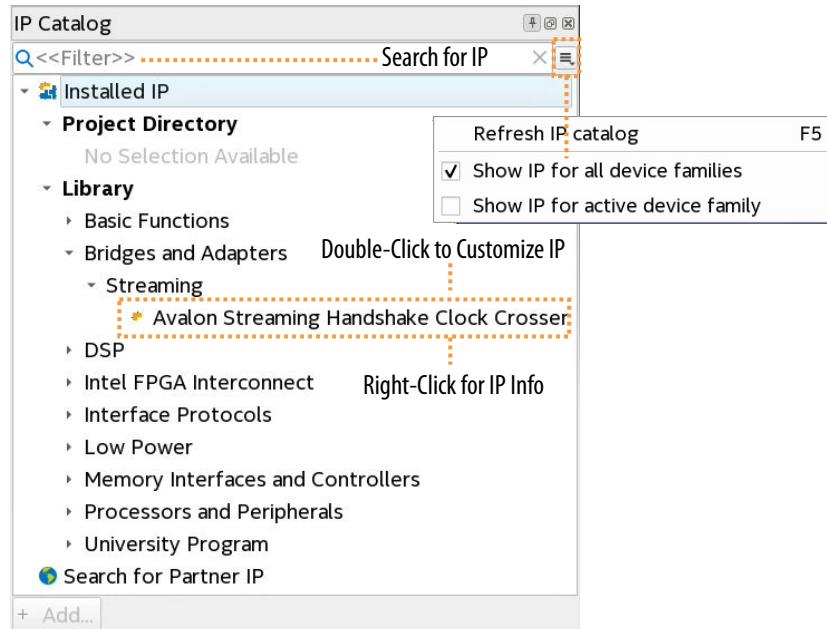
Intel and strategic IP partners offer a broad portfolio of configurable IP cores optimized for Intel FPGA devices.

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	Interface protocols
Bridges and adapters	Low power functions
DSP functions	Memory interfaces and controllers
Intel FPGA interconnect	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 50. Intel FPGA IP Catalog



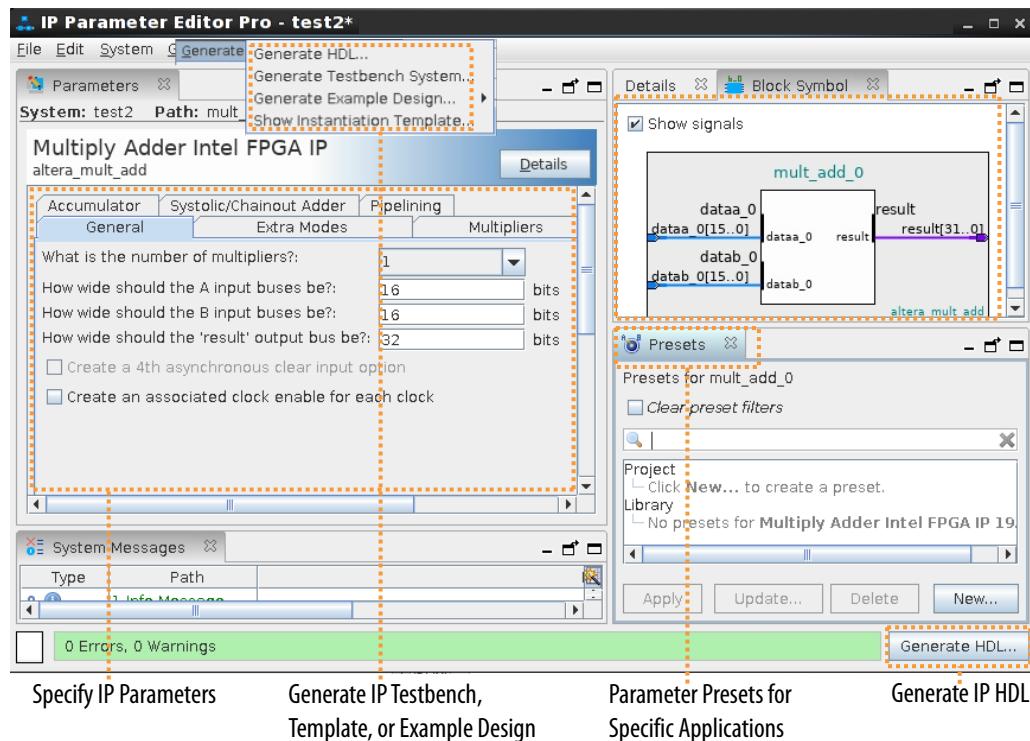
4.1. IP Catalog and Parameter Editor

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 prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Intel Quartus Prime IP file (.ip) for an IP variation in Intel Quartus Prime Pro Edition projects. This file represents the IP variation in the project, and stores parameterization information.⁽²⁾

Figure 51. Example IP Parameter Editor



⁽²⁾ The parameter editor generates a top-level Quartus IP file (.qip) for an IP variation in Intel Quartus Prime Standard Edition projects.

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).
- Click **Validate System Integrity** to validate a system's generic components against companion files. (Platform Designer systems only)
- Click **Sync All System Info** to validate a system's generic components against companion files. (Platform Designer systems only)

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* for information on use of IP in Platform Designer and Platform Designer, respectively.

Related Information

[Creating a System with Platform Designer](#)

4.2. 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 52. IP Core Installation Path

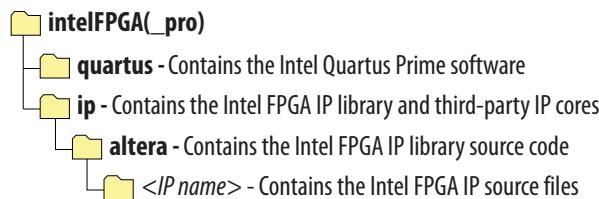


Table 16. IP Core Installation Locations

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.

4.2.1. 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.

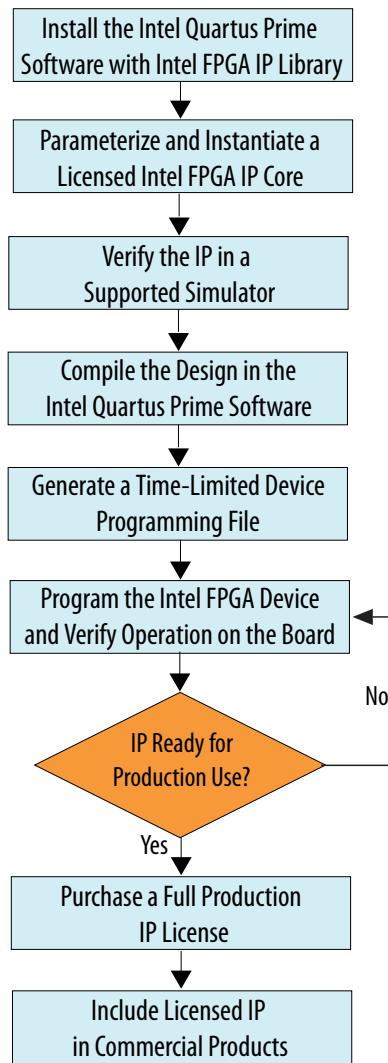
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **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 53. 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 [Intel FPGA Self-Service Licensing Center](#).

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 FPGA Licensing Support Center
- Introduction to Intel FPGA Software Installation and Licensing

4.2.1.1. Intel FPGA IP Versioning

Intel FPGA IP versions match the Intel Quartus Prime Design Suite software versions until v19.1. Starting in Intel Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

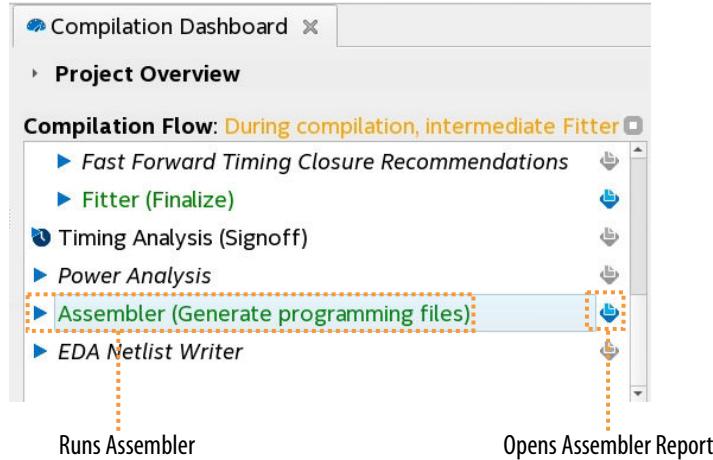
4.2.1.2. Checking the IP License Status

You can check the license status of all IP in an Intel Quartus Prime project by viewing the Assembler report.

To generate and view the Assembler report in the GUI:

1. Click **Assembler** on the Compilation Dashboard.
2. When the Assembler (and any prerequisite stages of compilation) complete, click the **Report** icon for the Assembler in the Compilation Dashboard.

Figure 54. Assembler Report Icon in Compilation Dashboard



3. Click the **Encrypted IP Cores Summary** report.

Figure 55. Encrypted IP Cores Summary Report

Assembler Encrypted IP Cores Summary			
Show: <input type="button" value="Visible"/> <input type="button" value="Hide"/> <input type="button" value="Search"/> <<Filter>>			
	Vendor	IP Core Name	License Type
1	Intel FPGA	Signal Tap (6AF7 BCE1)	Licensed
2	Intel FPGA	Signal Tap (6AF7 BCEC)	Licensed

To generate and view the Assembler report at the command line:

- Type the following command:

```
quartus_asm <project name> -c <project revision>
```

- View the output report in /output_files/<project_name>.asm.rpt.

```
+-----+
; Assembler Encrypted IP Cores Summary ;
+-----+
; Vendor ; IP Core Name ; License Type ;
+-----+
; Intel ; PCIe SRIOV with 4-PFs and 2K-VFs (6AF7 00FB) ; Unlicensed ;
; Intel ; Signal Tap (6AF7 BCE1) ; Licensed ;
; Intel ; Signal Tap (6AF7 BCEC) ; Licensed ;
+-----+
```

4.3. IP General Settings

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

Table 17. Location of IP Core General Settings in the Intel Quartus Prime Software

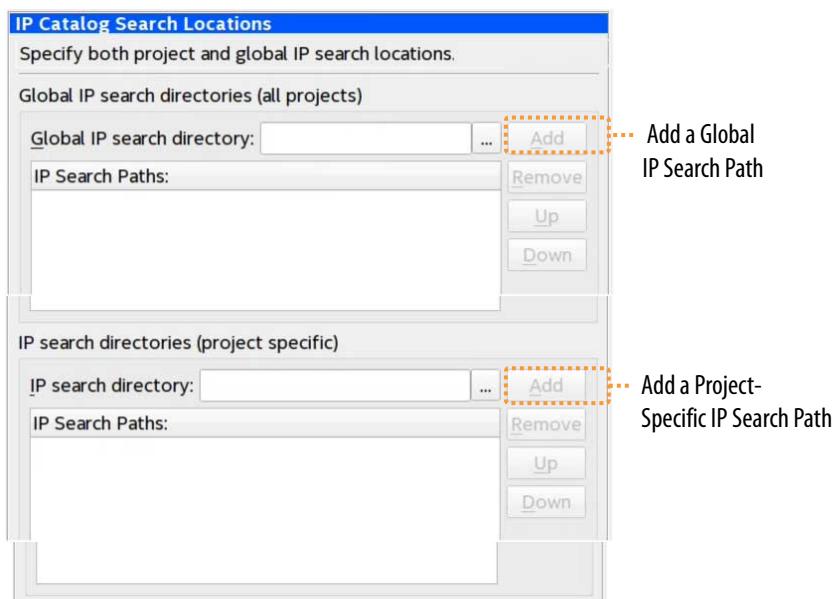
Setting	Description	Location
Maximum Platform Designer memory usage size	Increase if you experience slow processing for large systems, or for out of memory errors.	Tools > Options > IP Settings Or Tasks pane > Settings > IP Settings
IP generation HDL preference	The parameter editor generates the HDL you specify for IP variations.	
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.	
Generate IP simulation model when generating IP	Enables automatic generation of simulation models every time you generate the IP.	
Use available processors for parallel generation of Quartus project IPs	Directs Platform Designer to generate IPs in parallel, using the number of processors that you specify in the Compilation Process Settings pane of the Intel Quartus Prime project settings.	
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 Or Tasks pane > Settings > IP Catalog Search Locations

4.4. Adding IP to IP Catalog

The IP Catalog automatically displays Intel FPGA IP and other IP components that have a corresponding `_hw.tcl` or `.ipx` file located in the project directory, in the default Intel Quartus Prime installation directory, or in the IP search path. You can optionally add your own custom or third-party IP component to IP Catalog by adding the component's `_hw.tcl` or `.ipx` file to the IP search path.

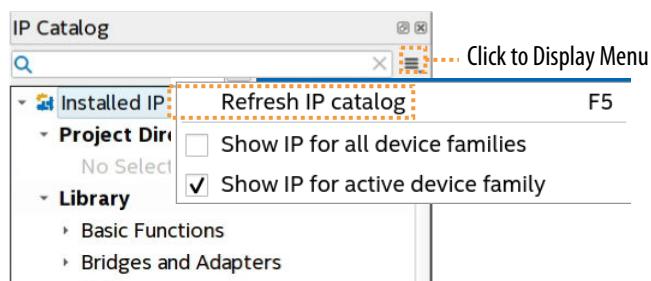
Follow these steps to add custom or third-party IP to the IP Catalog:

Figure 56. Specifying IP Search Locations



1. In the Intel Quartus Prime software, click **Tools > Options > IP Search Path** to open the **IP Search Path Options** dialog box.
2. Click **Add** or **Remove** to add/remove a location that contains IP.
3. To refresh the IP Catalog, click **Refresh IP Catalog** in the Intel Quartus Prime Platform Designer, or click **File > Refresh System** in Platform Designer.

Figure 57. Refreshing IP Catalog



4.5. 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-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 its 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 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 99 describes.

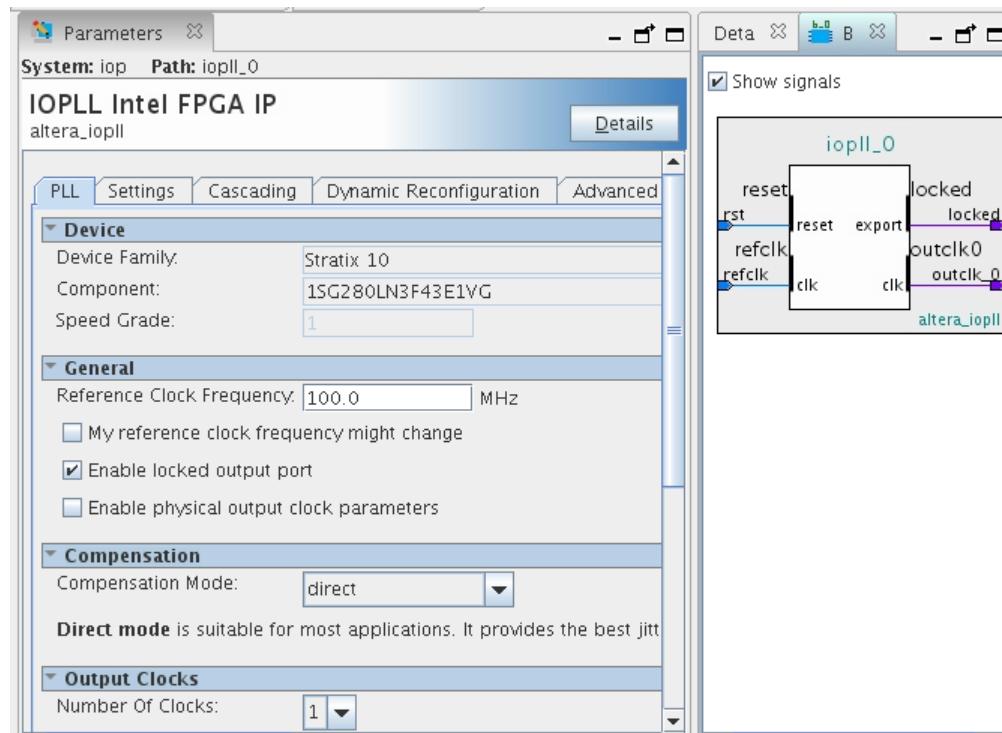
4.6. Specifying the IP Core Parameters and Options (Intel Quartus Prime Pro Edition)

Quickly configure Intel FPGA IP cores in the Intel Quartus Prime parameter editor. Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the IP core. The parameter editor generates the IP variation synthesis and optional simulation files, and adds the .ip file representing the variation to your project automatically.

Follow these steps to locate, instantiate, and customize an IP core in the parameter editor:

1. Create or open an Intel Quartus Prime project (.qpf) to contain the instantiated IP variation.
2. In the IP Catalog (**Tools > IP Catalog**), locate and double-click the name of the IP core to customize. To locate a specific component, type some or all of the component's name in the IP Catalog search box. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. Do not include spaces in IP variation names or paths. The parameter editor saves the IP variation settings in a file named `<your_ip>.ip`. Click **OK**. The parameter editor appears.

Figure 58. IP Parameter Editor (Intel Quartus Prime Pro Edition)



4. Set the parameter values in the parameter editor and view the block diagram for the component. The **Parameterization Messages** tab at the bottom displays any errors in IP parameters:
 - Optionally, select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
 - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
 - Specify options for processing the IP core files in other EDA tools.

Note: Refer to your IP core user guide for information about specific IP core parameters.
5. Click **Generate HDL**. The **Generation** dialog box appears.

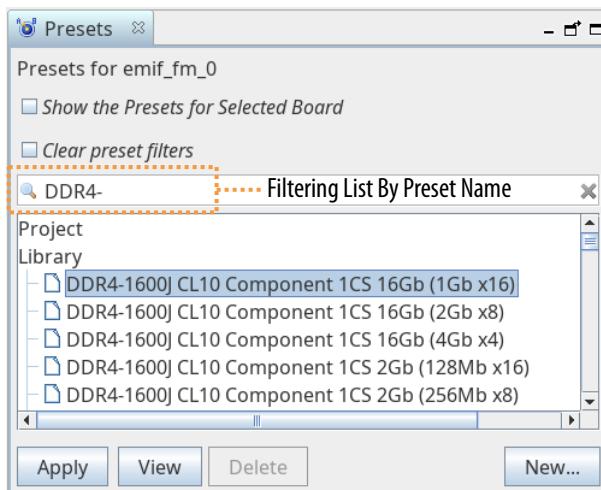
6. Specify output file generation options, and then click **Generate**. The synthesis and simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate > Generate Testbench System**. Specify testbench generation options, and then click **Generate**.
8. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate > Show Instantiation Template**.
9. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Note: Some IP cores generate different HDL implementations according to the IP core parameters. The underlying RTL of these IP cores contains a unique hash code that prevents module name collisions between different variations of the IP core. This unique code remains consistent, given the same IP settings and software version during IP generation. This unique code can change if you edit the IP core's parameters or upgrade the IP core version. To avoid dependency on these unique codes in your simulation environment, refer to *Generating a Combined Simulator Setup Script*.

4.6.1. Applying Preset Parameters for Specific Applications

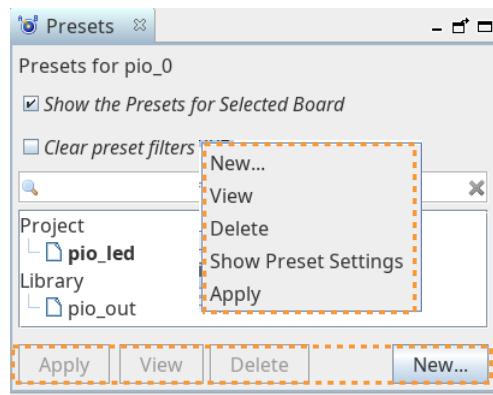
The **Preset** tab displays the names of available preset settings for an IP component. A preset is a specific collection of parameter settings that are appropriate for a specific protocol, application, or board. Double-click the preset name (or click **Apply**) to instantly apply the parameter values defined in the preset to the current IP instance.

Figure 59. Selecting Preset Parameters



4.6.1.1. Viewing, Applying, and Deleting IP Presets

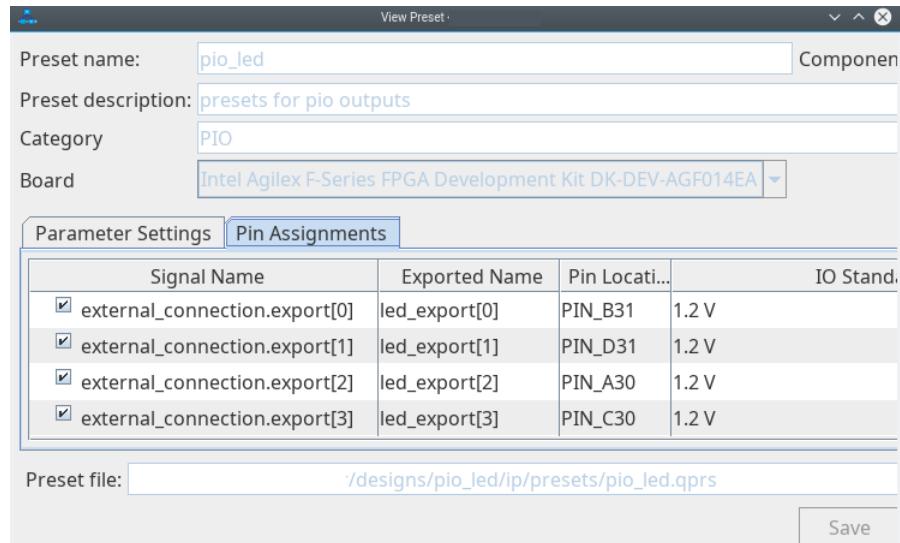
You can view the properties of a preset, apply a preset, or delete any existing preset in the **Presets** tab.

Figure 60. View, Apply, and Delete Presets in Presets Tab


Note: Right-click a preset to access the same **View**, **Apply**, and **Delete** preset functions in the context menu.

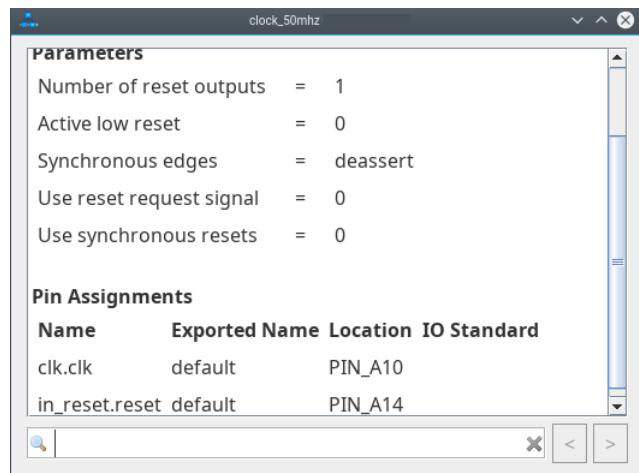
Viewing Presets

Click the **View** button to show the preset properties in the read-only **Update Preset** dialog box.

Figure 61. View Button Opens View Preset Dialog Box


Right-click a preset and click **Show Preset Settings** to view a searchable report of the preset settings.

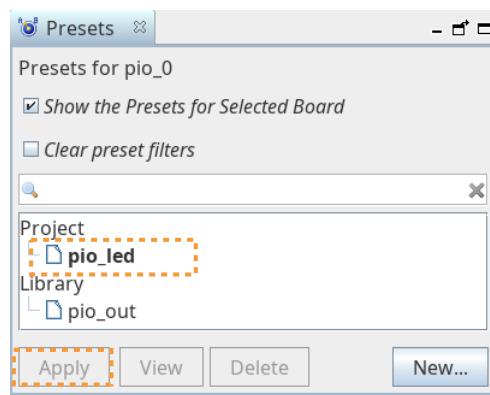
Figure 62. Show Preset Settings Searchable Report



Applying Presets to IP Instances

Click the **Apply** button (or double-click) to apply the IP preset to the currently selected IP. Applied presets appear in bold text.

Figure 63. Applied Presets Appear in Bold Text



Deleting Presets from the System

Click the **Delete** button to delete the current preset from the Platform Designer system.

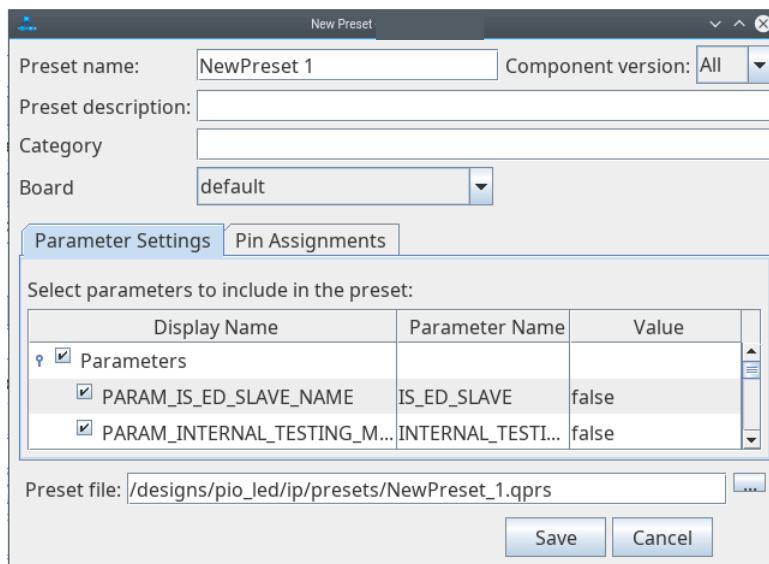
4.6.2. Customizing IP Presets

You can optionally define and save a custom set of parameter settings as an IP preset, and then apply the preset whenever you add an instance of the IP component to any system.

Follow these steps to save a custom IP preset:

1. In IP Catalog, double-click any component to launch the parameter editor.
2. To search for a specific preset to base initial settings, type a partial preset name in the search box.
3. In the **Presets** tab, click **New** to specify the **Preset name** and **Preset description**.
4. Under **Select parameters to include in the preset**, enable or disable the parameters you want to include in the preset.
5. Specify the path for the **Preset file** that preserves the collection of parameter settings. The location of the new .qprs preset file is added to the IP search path automatically.

Figure 64. Create New Preset



6. Click **Save**.
7. To apply the preset to an IP component, click **Apply**. Preset parameter values that match the current parameter settings appear in bold.

4.6.2.1. Defining Preset Pin Assignments

You can define pin assignments that are included as part of an IP preset. When you apply the IP preset to an IP instance, the pin assignments export during the IP or system's HDL generation.

You define preset pin assignments in the **Pin Assignments** tab of the **New Preset** dialog box, or in a Pin Assignments File (.tcl) that you create.

4.6.2.1.1. Defining Preset Pin Assignments in Pin Assignments Tab

The **Pin Assignments** tab allows you to specify the **Exported Name** of the signals, to select the appropriate **Pin Location**, and to select the appropriate **IO Standard** for the target board.

By default, the **Exported Name** name takes the form of:

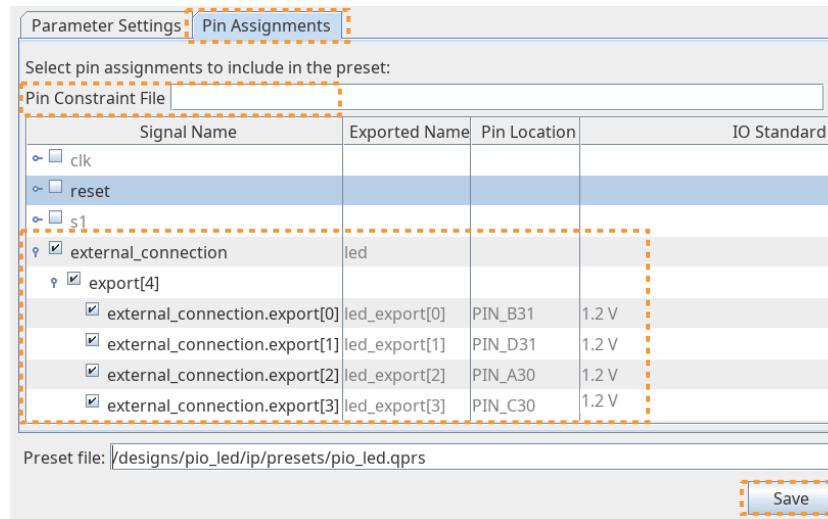
```
module_name + interface_name + pin_role
```

For example:

```
pio0_external_connection_export[0]
```

You can change the **Exported Name** by double-clicking on the **Exported Name** for the interface and typing a new name. All of the signals of the interface then update automatically to reflect the name you specify.

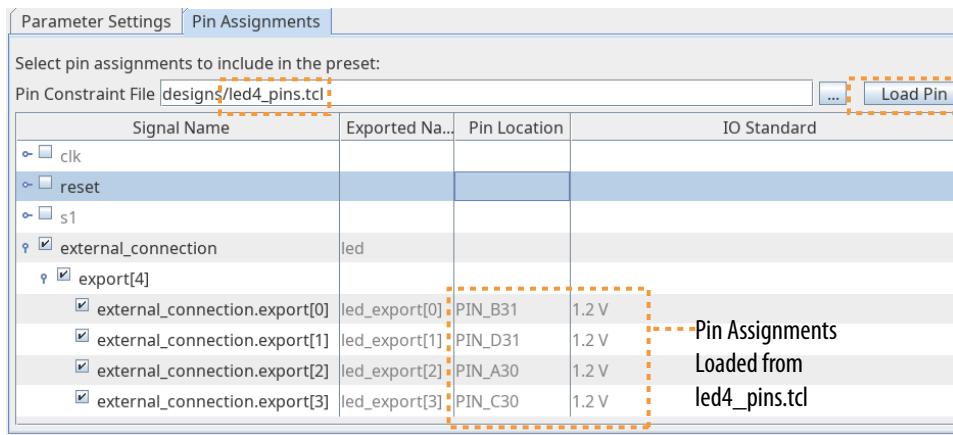
Figure 65. Pin Assignments Tab



For example, typing `led` for the **external_connection** interface updates the signals of the interface to `led_export[n]`. The **external_connection** is the interface name, and **external_connection_export(0)** is the signal name.

4.6.2.1.2. Defining Preset Pin Assignments in a Pin File

Alternatively, you can specify the pin assignments in a Pin Constraints File (.tcl), which can be more efficient for projects with many ports. You specify this .tcl file as the **Pin Constraint File** on the **Pin Assignments** tab, and then click **Load Pin**. The **Pin Location** and **IO Standard** update per the loaded pin assignments.

Figure 66. Loading Pin Assignments from Tcl File


The following shows the contents of an example Pin Constraints File (.tcl):

```
set_instance_assignment -to "led_export[0]" -name IO_STANDARD "1.2 V"
set_location_assignment -to "led_export[0]" "PIN_B31"
set_instance_assignment -to "led_export[1]" -name IO_STANDARD "1.2 V"
set_location_assignment -to "led_export[1]" "PIN_D31"
set_instance_assignment -to "led_export[2]" -name IO_STANDARD "1.2 V"
set_location_assignment -to "led_export[2]" "PIN_A30"
set_instance_assignment -to "led_export[3]" -name IO_STANDARD "1.2 V"
set_location_assignment -to "led_export[3]" "PIN_C30"
```

4.7. IP Core Generation Output (Intel Quartus Prime Pro Edition)

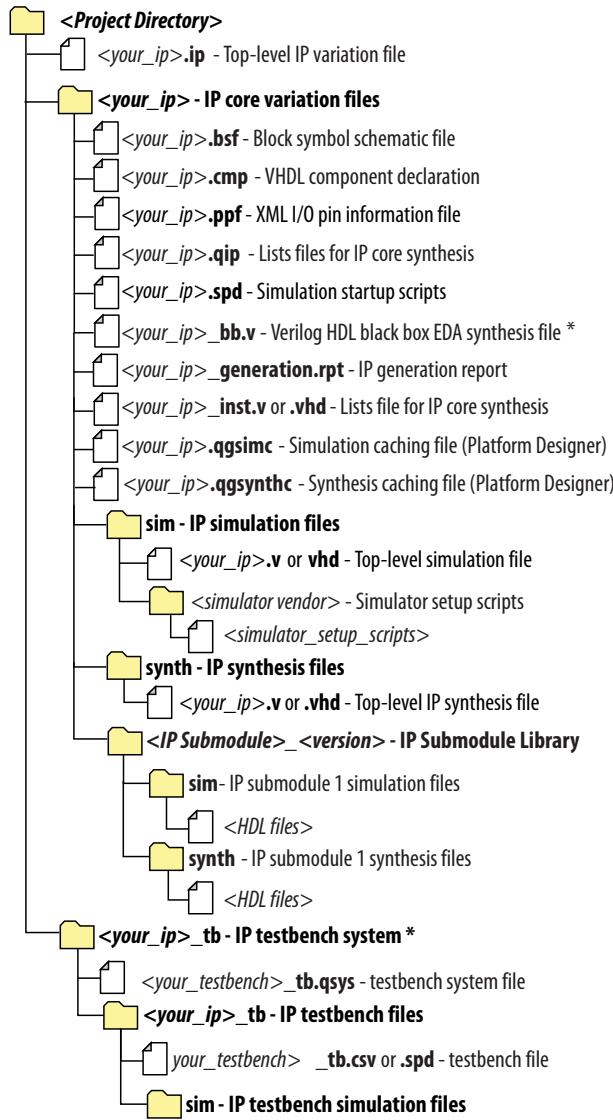
The Intel Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Platform Designer system.

Table 18. Output Files of Intel FPGA IP Generation

File Name	Description
<your_ip>.ip	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file.
<your_ip>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.
<your_ip>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<your_ip>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<your_ip>.csv	Contains information about the upgrade status of the IP component.
<your_ip>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).

continued...

File Name	Description
<your_ip>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<your_ip>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<your_ip>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<your_ip>_inst.v or _inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<your_ip>.regmap	If the IP contains register information, the Intel Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of host and agent interfaces. This file complements the .socinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<your_ip>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Intel Quartus Prime software stores the .svd files for agent interface visible to the System Console hosts in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system agents, Platform Designer accesses the registers by name.
<your_ip>.v <your_ip>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a msim_setup.tcl script to set up and run a simulation with a supported Siemens EDA simulator, such as the ModelSim simulator.
aldec/	Contains a Riviera-PRO* script rivierapro_setup.tcl to setup and run a simulation.
/synopsys/vcs /synopsys/vcsmx	Contains a shell script vcs_setup.sh to set up and run a VCS* simulation. Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a VCS MX simulation.
/xcelium	Contains an Xcelium* Parallel simulator shell script xcelium_setup.sh and other setup files to set up and run a simulation.
/submodules	Contains HDL files for the IP core submodule.
<IP submodule>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

Figure 67. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)


4.8. Scripting IP Core Generation

Use the `qsys-script` and `qsys-generate` utilities to define and generate an IP core variation outside of the Intel Quartus Prime GUI.

To parameterize and generate an IP core at command-line, follow these steps:

1. Run `qsys-script` to start a Tcl script that instantiates the IP and sets parameters:

```
qsys-script --script=<script_file>.tcl
```

2. Run qsys-generate to generate the IP core variation:

```
qsys-generate <IP variation file>.qsys
```

4.9. 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.

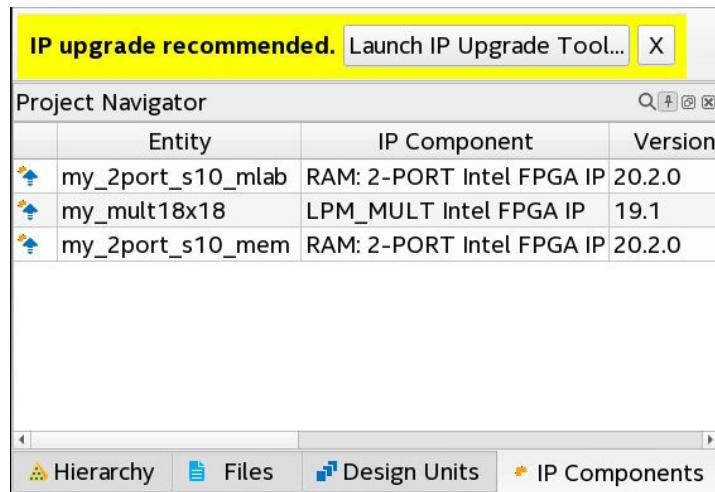
Table 19. Modifying an IP Variation

Menu Command	Action
File > Open	Select the top-level HDL (.v, or .vhdl) IP variation file to launch the parameter editor and modify the IP variation. Regenerate the IP variation to implement your changes.
View > 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.

4.10. 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 68. 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.

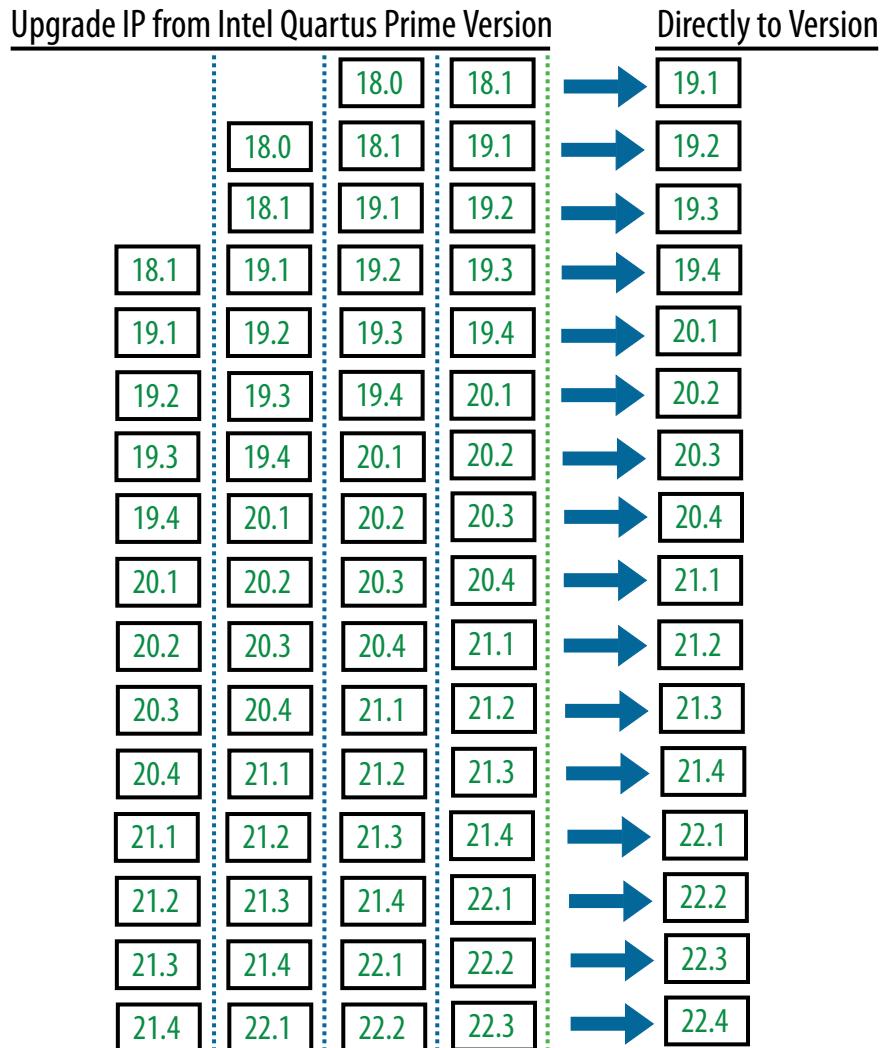
Note: Upgrading IP cores may append a unique identifier to the original IP core entity names, without similarly modifying the IP instance name. There is no requirement to update these entity references in any supporting Intel Quartus Prime file, such as the Intel Quartus Prime Settings File (.qsf), Synopsys* 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 20. 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.
IP Component Outdated 	Indicates that your IP variation uses an outdated version of the IP core.
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, such as the Intel Quartus Prime Standard Edition, generated this IP. Replace this IP component with a compatible component in the current edition.

Note: Beginning with the Intel Quartus Prime Pro Edition software version 19.1, IP upgrade supports migration of IP released within one year of the Intel Quartus Prime Pro Edition software version, as the following chart defines:

Figure 69. Intel Quartus Prime Pro Edition IP Version Upgrade Paths

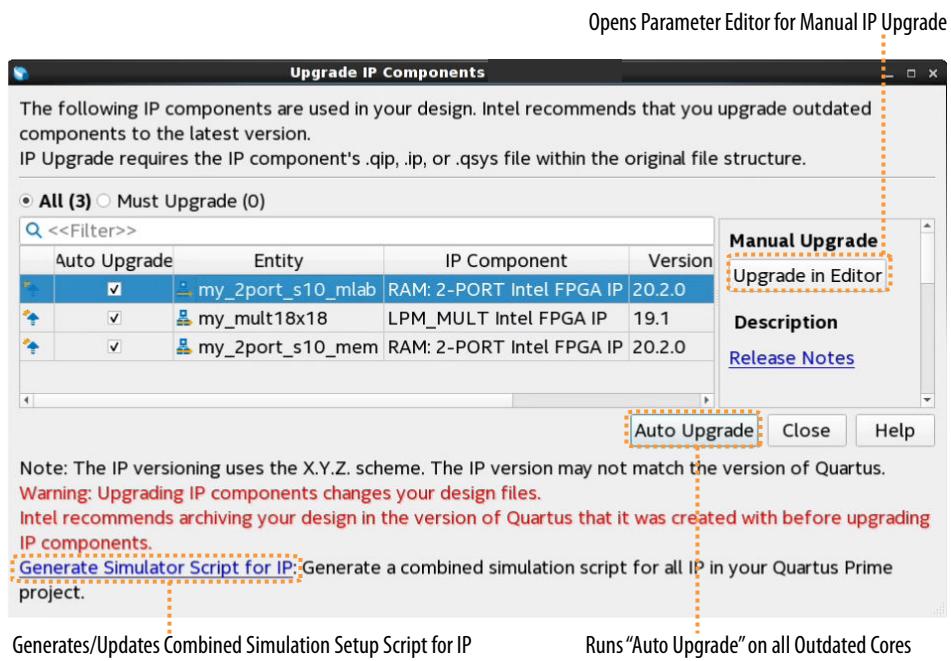


Follow these steps to upgrade IP cores:

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 **Auto Upgrade**. 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 70. Upgrading IP Cores (Intel Quartus Prime Pro Edition Example)



Generates/Updates Combined Simulation Setup Script for IP

Runs "Auto Upgrade" on all Outdated 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](#)

4.10.1. 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, .vhdl > \
<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>.<hdl1>" <quartus_project>
```

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

4.10.2. 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.

- To display the IP cores that require migration, click **Project > Upgrade IP Components**. The **Description** field provides migration instructions and version differences.
- 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.
- 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.
- 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.
- 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.
- 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](#)

4.10.3. Troubleshooting IP or Platform Designer System Upgrade

The **Upgrade IP Components** dialog box reports the version and status of each Intel FPGA 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 21. 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 <i><Project Directory>/ip_upgrade_port_diff_reports</i> 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 you 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 71. IP Port Differences Report

```

1 IP Ports Diff Report for pattern_generator_system_mm_bridge
2 Tue August 20 12:13:05 2020
3 Quartus Prime Version 20.3.0 Pro Edition
4
5
6 -----
7 ; Table of Contents ;
8 -----
9 1. IP Information
10 2. IP Ports Diff Table
11                                         Report Summary
12
13 -----
14 +-----+
15 ; IP Information
16 +-----+
17 ; IP Variation Name      ; pattern_generator_system_mm_bridge
18 ; Prior Version         ; 19.3
19 ; New Version           ; 20.3
20 ; New File              ; /mydata/_synth/pattern_generator_system_mm_bridge.v ;
21 ; Has Port Differences ; Yes
22 +-----+
23
24
25 +-----+
26 ; IP Ports Diff Table
27 +-----+-----+-----+-----+-----+
28 ; Change Type ; Port Name    ; Prior Port Range ; New Port Range ; Prior Port direction ; New Port direction
29 +-----+-----+-----+-----+-----+-----+
30 ; Modified    ; m0_address   ; [10:0]          ; [0:0]        ; output       ; output
31 ; Modified    ; m0_readdata  ; [31:0]          ; [0:0]        ; input        ; input
32 ; Modified    ; m0_writedata ; [31:0]          ; [0:0]        ; output       ; output
33 ; Modified    ; s0_address   ; [10:0]          ; [0:0]        ; input        ; input
34 ; Modified    ; s0_readdata  ; [31:0]          ; [0:0]        ; output       ; output
35 ; Modified    ; s0_writedata ; [31:0]          ; [0:0]        ; input        ; input
36 +-----+-----+-----+-----+-----+

```

4.11. Simulating Intel FPGA IP Cores

The Intel Quartus Prime software supports IP core RTL simulation in specific EDA simulators. IP generation optionally creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. You can 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:

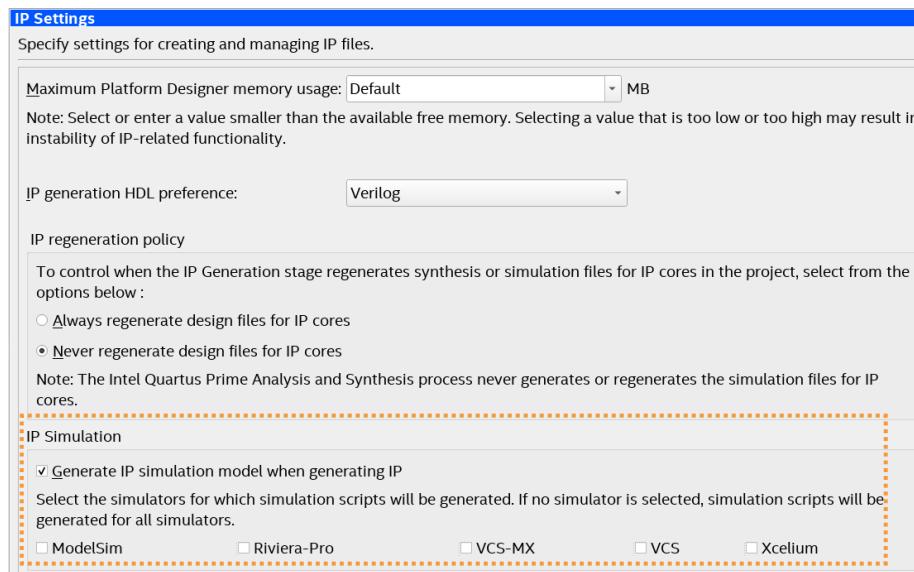
1. Generate IP HDL, testbench (or example design), and simulator setup script files.
2. Set up your simulator environment and any simulation scripts.
3. Compile simulation model libraries.
4. Run your simulator.

4.11.1. 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 specify options for the generation of IP simulation files, follow these steps:

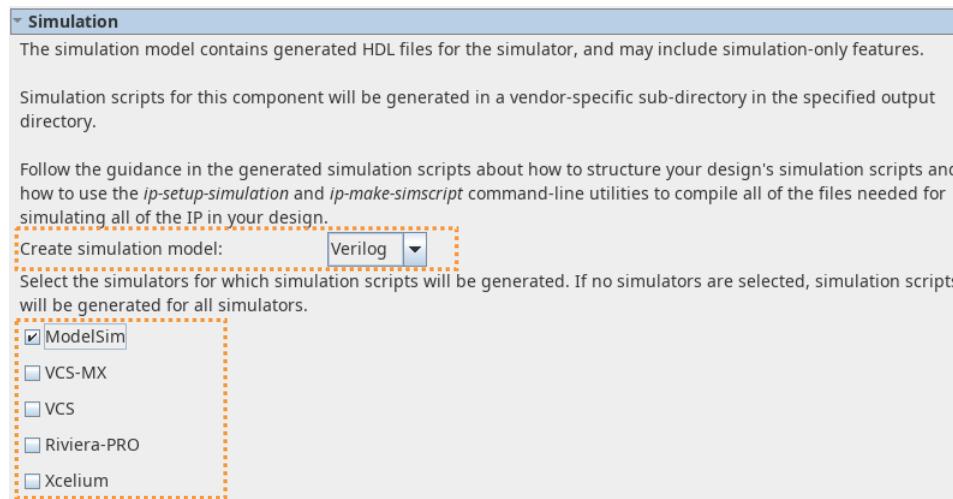
1. To specify your supported simulator and options for design simulation file generation, click **Assignment** > **Settings** > **EDA Tool Settings** > **Simulation**.
2. To specify your supported simulator and options for IP simulation file generation, click **Assignments** > **Settings** > **IP Settings** and specify the following:
 - To enable automatic generation of simulation models for all IP in the project when you generate IP during compilation, turn on the **Generate IP simulation model when generating IP** option under **IP Simulation**.
 - To specify one or more supported simulators for which to generate setup scripts, turn on one or more simulator option, or disable all simulator options to generate scripts for all simulators automatically.
3. To generate the simulation files, click **Processing** > **Start Compilation** to compile the design. The simulation models and setup scripts for the Intel FPGA IP generate in the `<your_project>/<ip_name>/sim/<vendor>` directory.

Figure 72. Project-Wide IP Generation Settings



You can optionally override these project-level **IP Settings** when you generate HDL for individual IP cores with the IP parameter editor. Prior to generation, you can specify a supported simulator, or specify no simulator to generate the setup scripts for all simulators in the parameter editor.

Figure 73. Simulation Options in Generation Dialog Box

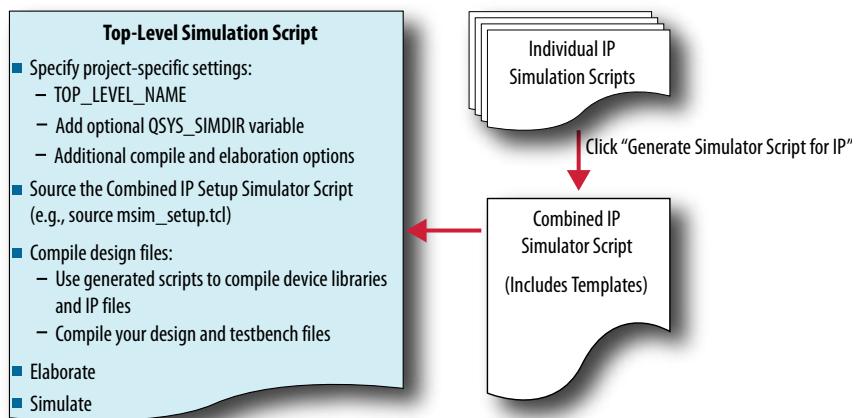


4.11.2. Scripting IP Simulation

The Intel Quartus Prime software supports the use of scripts to automate simulation processing in your preferred simulation environment. Use the scripting methodology that you prefer to control simulation.

Use a version-independent, top-level simulation script to control design, testbench, and IP core simulation. Because Intel Quartus Prime-generated simulation file names may change after IP upgrade or regeneration, your top-level simulation script must "source" the generated setup scripts, rather than using the generated setup scripts directly. Follow these steps to generate or regenerate combined simulator setup scripts:

Figure 74. Incorporating Generated Simulator Setup Scripts into a Top-Level Simulation Script



1. Click **Tools > Generate Simulator Script for IP** (or run the ip-setup-simulation utility) to generate or regenerate a combined simulator setup script for all IP for each simulator.
2. Use the templates in the generated script to source the combined script in your top-level simulation script. Each simulator's combined script file contains a rudimentary template that you adapt for integration of the setup script into a top-level simulation script.

This technique eliminates manual update of simulation scripts if you modify or upgrade the IP variation.

4.11.2.1. Generating a Combined Simulator Setup Script

You can run the **Generate Simulator Setup Script for IP** command to generate a combined simulator setup script.

You can then source this combined script from a top-level simulation script. Click **Tools > Generate Simulator Setup Script for IP** (or use of the ip-setup-simulation utility at the command-line) to generate or update the combined scripts, after any of the following occur:

- IP core initial generation or regeneration with new parameters
- Intel Quartus Prime software version upgrade
- IP core version upgrade

Table 22. ip-setup-simulation Utility

Utility	Syntax
<p>ip-setup-simulation generates a combined, version-independent simulation script for all Intel FPGA IP cores in your project. The command also automates regeneration of the script after upgrading software or IP versions. Use the <code>--compile-to-work</code> option to compile all simulation files into a single work library if your simulation environment requires. Use the <code>--use-relative-paths</code> option to use relative paths whenever possible.</p>	<pre>ip-setup-simulation --quartus-project=<my proj> --output-directory=<my_dir> --use-relative-paths --compile-to-work --use-relative-paths and --compile-to-work are optional. For command-line help listing all options for these executables, type: <utility name> --help.</pre>

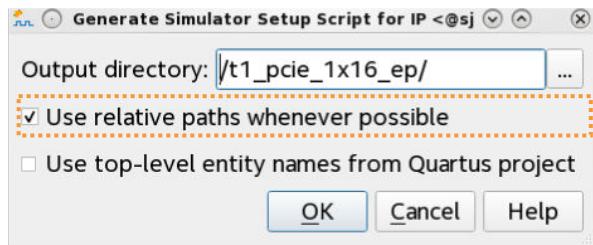
To generate a combined simulator setup script for all project IP cores for each simulator:⁽³⁾

1. Click **Tools > Generate Simulator Setup Script for IP** (or run the ip-setup-simulation utility). Specify the **Output Directory** and library compilation options. Click **OK** to generate the file. By default, the files generate into the /<project directory>/<simulator>/ directory using relative paths.

Note: For designs with F-tile IP, do not turn on the **Use top-level entity names from Quartus project** option.

⁽³⁾ If your design contains one or more F-tile IPs, you must first perform **Start Analysis & Elaboration** and then **Support-Logic Generation** before performing these steps.

Figure 75. Generate Simulator Setup Script for IP Dialog Box



2. To incorporate the generated simulator setup script into your top-level simulation script, refer to the template section in the generated simulator setup script as a guide to creating a top-level script:
 - a. Copy the specified template sections from the simulator-specific generated scripts and paste them into a new top-level file.
 - b. Remove the comments at the beginning of each line from the copied template sections.
 - c. Specify the customizations you require to match your design simulation requirements, for example:
 - Specify the TOP_LEVEL_NAME variable to the design's simulation top-level file. The top-level entity of your simulation is often a testbench that instantiates your design. Then, your design instantiates IP cores or Platform Designer systems. Set the value of TOP_LEVEL_NAME to the top-level entity.
 - If necessary, set the QSYS_SIMDIR variable to point to the location of the generated IP simulation files.
 - Specify any other changes, such as using the grep command-line utility to search a transcript file for error signatures, or e-mail a report.
3. Re-run **Tools > Generate Simulator Setup Script for IP** (or ip-setup-simulation) after regeneration of an IP variation.

4.11.2.1.1. Sourcing Aldec ActiveHDL* or Riviera Pro* Simulator Setup Scripts

Follow these steps to incorporate the generated ActiveHDL* or Riviera Pro* simulation scripts into a top-level project simulation script.

1. The generated simulation script contains the following template lines. Cut and paste these lines into a new file. For example, sim_top.do.

```
# # Start of template
# # If the copied and modified template file is "aldec.do", run it as:
# # vsim -c -do aldec.do
#
# # Source the generated sim script
# source rivierapro_setup.tcl
# # Compile eda/sim_lib contents first
# dev_com
# # Override the top-level name (so that elab is useful)
# set TOP_LEVEL_NAME top
# # Compile the standalone IP.
# com
# # Compile the top-level
# vlog -sv2k5 ../../top.sv
# # Elaborate the design.
# elab
# # Run the simulation
```

```
# run
# # Report success to the shell
# exit -code 0
# # End of template
```

2. Delete the first two characters of each line (comment and space):

```
# Start of template
# If the copied and modified template file is "aldec.do", run it as:
# vsim -c -do aldec.do
#
# Source the generated sim script
source rivierapro_setup.tcl
# Compile eda/sim_lib contents first
dev_com
# Override the top-level name (so that elab is useful)
set TOP_LEVEL_NAME top
# Compile the standalone IP.
com
# Compile the top-level
vlog -sv2k5 ../../top.sv
# Elaborate the design.
elab
# Run the simulation
run
# Report success to the shell
exit -code 0
# End of template
```

3. Modify the TOP_LEVEL_NAME and compilation step appropriately, depending on the simulation's top-level file. For example:

```
set TOP_LEVEL_NAME sim_top
vlog -sv2k5 ../../sim_top.sv
```

4. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes that you require to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
5. Run the new top-level script from the generated simulation directory:

```
vsim -c -do <path to sim_top>.tcl
```

4.11.2.1.2. Sourcing Cadence Incisive* Simulator Setup Scripts

Follow these steps to incorporate the generated Cadence Incisive* IP simulation scripts into a top-level project simulation script.

1. The generated simulation script contains the following template lines. Cut and paste these lines into a new file. For example, ncsim.sh.

```
# # Start of template
# # If the copied and modified template file is "ncsim.sh", run it as:
# # ./ncsim.sh
#
# # Do the file copy, dev_com and com steps
# source ncsim_setup.sh
# SKIP_ELAB=1
# SKIP_SIM=1
#
# # Compile the top level module
# ncvlog -sv "$QSYS_SIMDIR/../top.sv"
#
# # Do the elaboration and sim steps
# # Override the top-level name
# # Override the sim options, so the simulation
```

```
# # runs forever (until $finish()).
# source ncsim_setup.sh
# SKIP_FILE_COPY=1
# SKIP_DEV_COM=1
# SKIP_COM=1
# TOP_LEVEL_NAME=top
# USER_DEFINED_SIM_OPTIONS=""
# # End of template
```

2. Delete the first two characters of each line (comment and space):

```
# Start of template
# If the copied and modified template file is "ncsim.sh", run it as:
# ./ncsim.sh
#
# Do the file copy, dev_com and com steps
source ncsim_setup.sh
SKIP_ELAB=1
SKIP_SIM=1
# Compile the top level module
ncvlog -sv "$QSYS_SIMDIR/../../top.sv"
# Do the elaboration and sim steps
# Override the top-level name
# Override the sim options, so the simulation
# runs forever (until $finish()).
source ncsim_setup.sh
SKIP_FILE_COPY=1
SKIP_DEV_COM=1
SKIP_COM=1
TOP_LEVEL_NAME=top
USER_DEFINED_SIM_OPTIONS=""
# End of template
```

3. Modify the TOP_LEVEL_NAME and compilation step appropriately, depending on the simulation's top-level file. For example:

```
TOP_LEVEL_NAME=sim_top \
ncvlog -sv "$QSYS_SIMDIR/../../top.sv"
```

4. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes that you require to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
5. Run the resulting top-level script from the generated simulation directory by specifying the path to ncsim.sh.

4.11.2.1.3. Sourcing Cadence Xcelium Simulator Setup Scripts

1. The generated simulation script contains the following template lines. Cut and paste these lines into a new file. For example, xmsim.sh.

```
# #Start of template
# # Xcelium Simulation Script.
# # If the copied and modified template file is "xmsim.sh", run it as:
# # ./xmsim.sh
#
# # Do the file copy, dev_com and com steps
# source <script generation output directory>/xcelium/xcelium_setup.sh \
# SKIP_ELAB=1 \
# SKIP_SIM=1 \
# USER_DEFINED_COMPILE_OPTIONS=<compilation options for your design> \
# USER_DEFINED_VHDL_COMPILE_OPTIONS=<VHDL compilation options for your \
# design> \
# USER_DEFINED_VERILOG_COMPILE_OPTIONS=<Verilog compilation options for
```

```
# your design> \
# QSYS_SIMDIR=<script generation output directory>
# #
# # Compile all design files and testbench files, including the top level.
# # (These are all the files required for simulation other than the files
# # compiled by the IP script)
# #
# xmvlog <compilation options> <design and testbench files>
# #
# # TOP_LEVEL_NAME is used in this script to set the top-level simulation
# # or testbench module/entity name.
# #
# # Run the IP script again to elaborate and simulate the top level:
# # - Specify TOP_LEVEL_NAME and USER_DEFINED_ELAB_OPTIONS.
# # - Override the default USER_DEFINED_SIM_OPTIONS. For example, to run
# # until $finish(), set to an empty string: USER_DEFINED_SIM_OPTIONS="".
# #
# source <script generation output directory>/xcelium/xcelium_setup.sh \
# SKIP_FILE_COPY=1 \
# SKIP_DEV_COM=1 \
# SKIP_COM=1 \
# TOP_LEVEL_NAME=<simulation top> \
# USER_DEFINED_ELAB_OPTIONS=<elaboration options for your design> \
# USER_DEFINED_SIM_OPTIONS=<simulation options for your design>
# # End of template
```

2. Delete the first two characters of each line (comment and space):

```
# Start of template
# Xcelium Simulation Script (Beta Version).
# If the copied and modified template file is "xmsim.sh", run it as:
# ./xmsim.sh
#
# Do the file copy, dev_com and com steps
source <script generation output directory>/xcelium/xcelium_setup.sh \
SKIP_ELAB=1 \
SKIP_SIM=1 \
USER_DEFINED_COMPILE_OPTIONS=<compilation options for your design> \
USER_DEFINED_VHDL_COMPILE_OPTIONS=<VHDL compilation options for your design> \
\
USER_DEFINED_VERILOG_COMPILE_OPTIONS=<Verilog compilation options for your
design> \
QSYS_SIMDIR=<script generation output directory>
#
# Compile all design files and testbench files, including the top level.
# (These are all the files required for simulation other than the files
# compiled by the IP script)
#
xmvlog <compilation options> <design and testbench files>
#
# TOP_LEVEL_NAME is used in this script to set the top-level simulation or
# testbench module/entity name.
#
# Run the IP script again to elaborate and simulate the top level:
# - Specify TOP_LEVEL_NAME and USER_DEFINED_ELAB_OPTIONS.
# - Override the default USER_DEFINED_SIM_OPTIONS. For example, to run
# until $finish(), set to an empty string: USER_DEFINED_SIM_OPTIONS="".
#
source <script generation output directory>/xcelium/xcelium_setup.sh \
SKIP_FILE_COPY=1 \
SKIP_DEV_COM=1 \
SKIP_COM=1 \
TOP_LEVEL_NAME=<simulation top> \
USER_DEFINED_ELAB_OPTIONS=<elaboration options for your design> \
USER_DEFINED_SIM_OPTIONS=<simulation options for your design>
# End of template
```

3. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes that you require to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
4. Run the resulting top-level script from the generated simulation directory by specifying the path to xmsim.sh.

4.11.2.1.4. Sourcing ModelSim* or QuestaSim Simulator Setup Scripts

Follow these steps to incorporate the generated ModelSim* or QuestaSim IP simulation scripts into a top-level project simulation script.

1. The generated simulation script contains the following template lines. Cut and paste these lines into a new file. For example, sim_top.do.

```
# # Start of template
# # If the copied and modified template file is "mentor.do", run it
# # as: vsim -c -do mentor.do
#
# # Source the generated sim script
# source msim_setup.tcl
# # Compile eda/sim_lib contents first
# dev_com
# # Override the top-level name (so that elab is useful)
# set TOP_LEVEL_NAME top
# # Compile the standalone IP.
# com
# # Compile the top-level
# vlog -sv ../../top.sv
# # Elaborate the design.
# elab
# # Run the simulation
# run -a
# # Report success to the shell
# exit -code 0
# # End of template
```

2. Delete the first two characters of each line (comment and space):

```
# Start of template
# If the copied and modified template file is "mentor.do", run it
# as: vsim -c -do mentor.do
#
# Source the generated sim script source msim_setup.tcl
# Compile eda/sim_lib contents first
dev_com
# Override the top-level name (so that elab is useful)
set TOP_LEVEL_NAME top
# Compile the standalone IP.
com
# Compile the top-level vlog -sv ../../top.sv
# Elaborate the design.
elab
# Run the simulation
run -a
# Report success to the shell
exit -code 0
# End of template
```

3. Modify the TOP_LEVEL_NAME and compilation step appropriately, depending on the location of the simulation's top-level file. For example:

```
set TOP_LEVEL_NAME sim_top vlog -sv ../../sim_top.sv
```

4. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes required to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
5. Run the resulting top-level script from the generated simulation directory:

```
vsim -c -do <path to sim_top>.tcl
```

4.11.2.1.5. Sourcing Synopsys VCS Simulator Setup Scripts

Follow these steps to incorporate the generated Synopsys VCS simulation scripts into a top-level project simulation script.

1. The generated simulation script contains these template lines. Cut and paste the lines preceding the “helper file” into a new executable file. For example, `synopsys_vcs.f`.

```
# # Start of template
# # If the copied and modified template file is "vcs_sim.sh", run it
# # as: ./vcs_sim.sh
#
# # Override the top-level name
# # specify a command file containing elaboration options
# # (system verilog extension, and compile the top-level).
# # Override the sim options, so the simulation
# # runs forever (until $finish()).
# source vcs_setup.sh
# TOP_LEVEL_NAME=top
# USER_DEFINED_ELAB_OPTIONS="-f ../../synopsys_vcs.f"
# USER_DEFINED_SIM_OPTIONS=""
#
# # helper file: synopsys_vcs.f
# +systemverilogext+.sv
# ../../top.sv
# # End of template
```

2. Delete the first two characters of each line (comment and space) for the `vcs.sh` file, as shown below:

```
# Start of template
# If the copied and modified template file is "vcs_sim.sh", run it
# as: ./vcs_sim.sh
#
# Override the top-level name
# specify a command file containing elaboration options
# (system verilog extension, and compile the top-level).
# Override the sim options, so the simulation
# runs forever (until $finish()).
source vcs_setup.sh
TOP_LEVEL_NAME=top
USER_DEFINED_ELAB_OPTIONS="-f ../../synopsys_vcs.f"
USER_DEFINED_SIM_OPTIONS=""
```

3. Delete the first two characters of each line (comment and space) for the `synopsys_vcs.f` file, as shown below:

```
# helper file: synopsys_vcs.f
+systemverilogext+.sv
../../top.sv
# End of template
```

4. Modify the TOP_LEVEL_NAME and compilation step appropriately, depending on the simulation's top-level file. For example:

```
TOP_LEVEL_NAME=sim_top
```

5. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes required to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
6. Run the resulting top-level script from the generated simulation directory by specifying the path to vcs_sim.sh.

4.11.2.1.6. Sourcing Synopsys VCS MX Simulator Setup Scripts

Follow these steps to incorporate the generated Synopsys VCS MX simulation scripts for use in top-level project simulation scripts.

1. The generated simulation script contains these template lines. Cut and paste the lines preceding the "helper file" into a new executable file. For example, vcsmx.sh.

```
# # Start of template
# # If the copied and modified template file is "vcsmx_sim.sh", run
# # it as: ./vcsmx_sim.sh
# #
# # Do the file copy, dev_com and com steps
# source vcsmx_setup.sh
# SKIP_ELAB=1

# SKIP_SIM=1
#
# # Compile the top level module
# vlogan +v2k
#     +systemverilogext+.sv "$QSYS_SIMDIR/../../top.sv"

# # Do the elaboration and sim steps
# # Override the top-level name
# # Override the sim options, so the simulation runs
# # forever (until $finish()).
# source vcsmx_setup.sh
# SKIP_FILE_COPY=1
# SKIP_DEV_COM=1
# SKIP_COM=1
# TOP_LEVEL_NAME="-top top"
# USER_DEFINED_SIM_OPTIONS=""
# # End of template
```

2. Delete the first two characters of each line (comment and space), as shown below:

```
# Start of template
# If the copied and modified template file is "vcsmx_sim.sh", run
# it as: ./vcsmx_sim.sh
#
# Do the file copy, dev_com and com steps
source vcsmx_setup.sh
SKIP_ELAB=1
SKIP_SIM=1

# Compile the top level module
vlogan +v2k +systemverilogext+.sv "$QSYS_SIMDIR/../../top.sv"

# Do the elaboration and sim steps
# Override the top-level name
# Override the sim options, so the simulation runs
# forever (until $finish()).
```

```
source vcsmx_setup.sh
SKIP_FILE_COPY=1
SKIP_DEV_COM=1
SKIP_COM=1
TOP_LEVEL_NAME="-top top"
USER_DEFINED_SIM_OPTIONS=""
# End of template
```

3. Modify the TOP_LEVEL_NAME and compilation step appropriately, depending on the simulation's top-level file. For example:

```
TOP_LEVEL_NAME="-top sim_top"
```

4. Make the appropriate changes to the compilation of your top-level file, for example:

```
vlogan +v2k +systemverilogext+.sv "$QSYS_SIMDIR/../../sim_top.sv"
```

5. If necessary, add the QSYS_SIMDIR variable to point to the location of the generated IP simulation files. Specify any other changes required to match your design simulation requirements. The scripts offer variables to set compilation or simulation options. Refer to the generated script for details.
6. Run the resulting top-level script from the generated simulation directory by specifying the path to vcsmx_sim.sh.

4.12. Generating Simulation Files for Platform Designer Systems and IP Variants

If your design contains Intel FPGA IP or a Platform Designer system, you must first generate files for RTL simulation of the IP or system with the Intel Quartus Prime Platform Designer before running simulation.

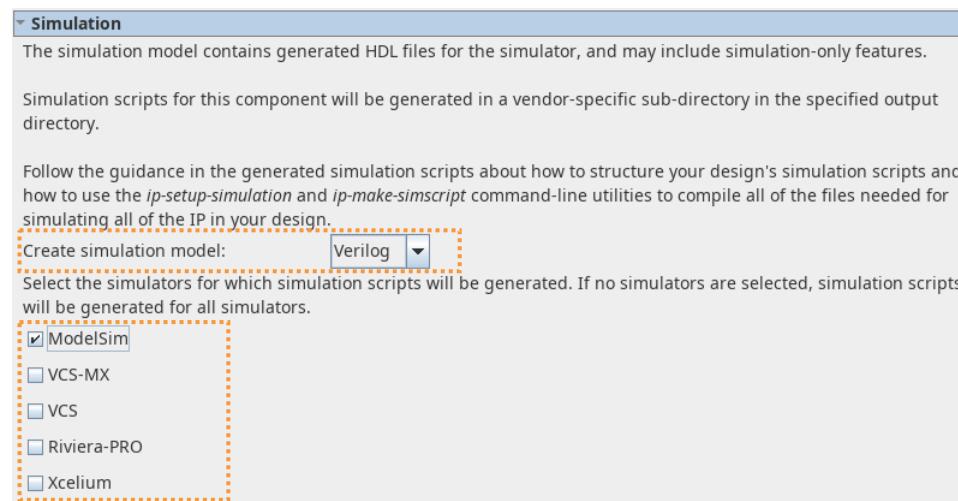
When you generate the system (or IP variant), Platform Designer optionally creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core.

You can use the functional simulation model and any testbench or example design for simulation of the IP or system. The 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.

To generate the simulation model and simulator setup scripts for your Platform Designer system or component, follow these steps:

1. Click **Tools > Platform Designer**. Platform Designer opens or creates a Platform Designer system or IP variant.
2. In Platform Designer, after specifying parameters, click **Generate > Generate HDL**. The **Generation** dialog box appears.
3. Under **Simulation**, specify **Verilog** or **VHDL** for the **Create simulation model** option.

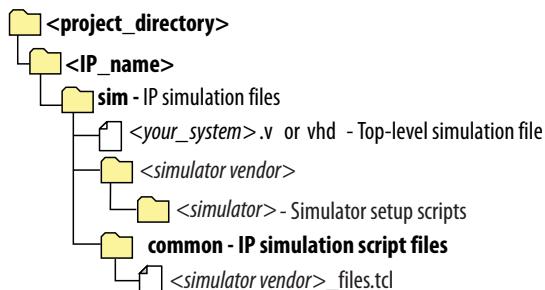
Figure 76. Simulation Options in Generation Dialog Box



4. Turn on or off the **ModelSim**, **VCS-MX**, **VCS**, **Riviera-Pro**, or **Xcelium** option to generate simulator setup scripts for the simulation tool. If you turn on no simulator options, the scripts generate for all simulators.
5. Click the **Generate** button. Platform Designer generates the simulation models and setup scripts for your system or IP component in the following directory:

```
<top-level system name>/<system name>/<sim>/<simulator>
```

Figure 77. Generated Simulation Files Location



By default, Platform Designer generates the simulation scripts for the currently loaded system and all subsystems. Alternatively, you can open a subsystem to generate a simulation script only for that subsystem.

You can use scripts to compile the required device libraries and system design files in the correct order and elaborate or load the top-level system for simulation.

Table 23. Simulation Script Variables

The simulation scripts provide variables that allow flexibility in your simulation environment.

Variable	Description
TOP_LEVEL_NAME	If the testbench Platform Designer system is not the top-level instance in your simulation environment because you instantiate the Platform Designer testbench within your own top-level simulation file, set the TOP_LEVEL_NAME variable to the top-level hierarchy name.
QSYS_SIMDIR	If the simulation files generated by Platform Designer are not in the simulation working directory, use the QSYS_SIMDIR variable to specify the directory location of the Platform Designer simulation files.
QUARTUS_INSTALL_DIR	Points to the Quartus installation directory that contains the device family library.

Example 1. Top-Level Simulation HDL File for a Testbench System

The example below shows the pattern_generator_tb generated for a Platform Designer system called pattern_generator. The top.sv file defines the top-level module that instantiates the pattern_generator_tb simulation model, as well as a custom SystemVerilog test program with BFM transactions, called test_program.

```
module top();
  pattern_generator_tb tb();
  test_program pgm();
endmodule
```

4.13. 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.

4.14. 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.

4.14.1. 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_dataaa;

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

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

4.14.2. 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_dataaa : std_logic_vector(31 downto 0);
begin

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

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

4.15. Support for the IEEE 1735 Encryption Standard

The Intel Quartus Prime Pro Edition software supports the IEEE 1735 v1 encryption standard for IP core file decryption. You can encrypt the Verilog HDL or VHDL IP files with the `encrypt_1735` utility, or with a third-party encryption tool that supports the IEEE 1735 standard. You can then use the encrypted files in the Intel Quartus Prime Pro Edition software and simulation tools that support the IEEE 1735 encryption standard.

The encryption key is the same for Verilog HDL and VHDL. You can pass parameters to the instantiation of an encrypted module using the same method as a non-encrypted module.

Type `encrypt_1735 --help` at the Intel Quartus Prime command line to view syntax and all supported options for the `encrypt_1735` utility.

```
encrypt_1735 [-h | --help[=<option|topic>] | -v]
encrypt_1735 <other options>

Options:
-----
-?
-f <argument file>
-h
--256_bit[=<value>]
--help[=<option|topic>]
--language=<verilog | systemverilog| vhdl>
--lower_priority
--of=<some_file>
--quartus
--simulation[=<aldec | cadence | mentor | synopsys (comma delimited)>]
--tcl_jou_file=<[tcl_jou_filename=]on|off>
--tcl_log_file=<[tcl_log_filename=]on|off>
```

Adding the following Verilog or VHDL pragma to your RTL, along with the public key, enables the Intel Quartus Prime software to use the key to decrypt IP core files.

Verilog/SystemVerilog Encryption Pragma (Third-Party Tools):

```
`pragma protect key_keyowner="Intel Corporation"
`pragma protect data_method="aes128-cbc"
`pragma protect key_method="rsa"
`pragma protect key_keyname="Intel-FPGA-Quartus-RSA-1"
`pragma protect key_public_key
<encrypted session key>

`pragma protect begin
`pragma protect end
```

VHDL Encryption Pragma (Third-Party Tools):

```
`protect key_keyowner = "Intel Corporation"
`protect data_method="aes128-cbc"
`protect key_method = "rsa"
`protect key_keyname = "Intel-FPGA-Quartus-RSA-1"
`protect key_block
<Encrypted session key>
```

Only file encryption with a third-party tool requires the public encryption key. File encryption with the Intel Quartus Prime Pro Edition software does not require the public encryption key.

Use one of the following methods to obtain the public encryption key:

- To obtain the encryption key, login or register for a My-Intel account, and then submit an Intel Premier Support case requesting the encryption key.
- If you are ineligible for Intel Premier Support, you can submit a question regarding the "IEEE 1735 Encryption Public Key" to the Intel Community Forum for assistance.

Note:

The Intel Quartus Prime Standard Edition software does not support IEEE 1735 encryption.

Related Information

- [My-Intel.com](#)
- [Intel Community Forum](#)

4.16. Introduction to Intel FPGA IP Cores Revision History

This chapter has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
2022.12.12	22.4	<ul style="list-style-type: none"> • Revised <i>Applying Preset Parameters for Specific Applications</i> topic for board-aware features. • Added new <i>Viewing, Applying, and Deleting IP Presets</i> topic. • Added new <i>Example IP Preset File (.qprs)</i> topic. • Revised <i>Customizing IP Presets</i> topic for board-aware features. • Added new <i>Defining Preset Pin Assignments</i> section.
2021.10.04	21.3	<ul style="list-style-type: none"> • Updated <i>Intel Quartus Prime Pro Edition IP Version Upgrade Paths</i> figure for latest versions.
2021.03.29	21.1	<ul style="list-style-type: none"> • Enhanced <i>Simulating Intel FPGA Designs</i> topic with screenshot, links, and additional contextual details. • Updated supported simulator versions and removed support for Cadence Incisive Enterprise* in <i>Simulator Support</i> topic. • Revised <i>Generating IP Simulation Files</i> topic for new simulation file output options. • Revised <i>Using the EDA Netlist Writer</i> wording for clarity.
2020.11.09	20.3	<ul style="list-style-type: none"> • Revised "Introduction to Intel FPGA IP Cores" topic to include Bridges and Adapters and Intel FPGA Interconnect categories in IP Catalog. Updated IP Catalog image. • Revised wording of "Intel FPGA IP Versioning" topic for clarity. • Added screenshot to "Checking the IP License Status" topic. • Added "IP Version Upgrade Paths" diagram to "Upgrading IP Cores" topic. • Updated IP Port Differences Report image in "Troubleshooting IP or System Upgrade" topic.
2019.09.30	19.3	<ul style="list-style-type: none"> • Added "Checking the IP License Status" topic. • Added details to "Support for the IEEE 1735 Encryption Standard." • Added Intel FPGA IP Versioning" topic.
2019.05.13	18.1	<ul style="list-style-type: none"> • Added archives topic. • Updated the keyname and added --help information to "Support for the IEEE 1735 Encryption Standard."
2018.10.24	18.1	<ul style="list-style-type: none"> • Updated information about obtaining IEEE 1735 Encryption key.
2018.09.24	18.1	<ul style="list-style-type: none"> • 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 <i>Getting Started User Guide</i>.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.05.07	18.0	<ul style="list-style-type: none">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	<ul style="list-style-type: none">Revised product branding for Intel standards.Revised topics on Intel FPGA IP Evaluation Mode (formerly OpenCore).
2017.05.08	17.0	<ul style="list-style-type: none">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	<ul style="list-style-type: none">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.
2018.05.07	18.0	Initial release as separate chapter of <i>Getting Started User Guide</i> .

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 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 of project assignment upgrade that migration requires.

Related Information

- [Modify Entity Name Assignments](#) on page 118
- [Resolve Timing Constraint Entity Names](#) on page 118
- [Verify Generated Node Name Assignments](#) on page 119
- [Replace Logic Lock \(Standard\) Regions](#) on page 119
- [Modify Signal Tap Logic Analyzer Files](#) on page 121
- [Remove Unsupported Feature Assignments](#) on page 122

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 24. 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

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

Example 2. 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 3. 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 4. 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 5. 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 6. 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 > 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 > 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) 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_ASYNCRONOUS_SIGNAL_PIPELINING  
PHYSICAL_SYNTHESIS_MAP_LOGIC_TO_MEMORY_FOR_AREA
```

5.3. Upgrade IP Cores and Platform Designer Systems

Upgrade all IP cores and Platform Designer 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 systems for synthesis files. The Intel Quartus Prime Pro Edition does not support this scheme. To upgrade all IP cores and Platform Designer systems in your project, click **Project > Upgrade IP Components**.⁽⁴⁾

Table 25. IP Core and Platform Designer System Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
IP and Platform Designer system generation use a proprietary Verilog HDL configuration scheme within the top level of IP cores and Platform Designer 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: <ul style="list-style-type: none">• 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 <code>altera_pcie_a10_1ip_151</code> 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*.

Related Information

- [Introduction to Intel FPGA IP Cores](#) on page 75

⁽⁴⁾ 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."

- [Managing Intel Quartus Prime Projects](#) on page 10

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 124
- [Update Entity Auto-Discovery](#) on page 125
- [Ensure Distinct VHDL Namespace for Each Library](#) on page 126
- [Remove Unsupported Parameter Passing](#) on page 126
- [Remove Unsized Constant from WYSIWYG Instantiation](#) on page 126
- [Remove Non-Standard Pragmas](#) on page 127
- [Declare Objects Before Initial Values](#) on page 127
- [Confine SystemVerilog Features to SystemVerilog Files](#) on page 127
- [Avoid Assignment Mixing in Always Blocks](#) on page 128
- [Avoid Unconnected, Non-Existent Ports](#) on page 128
- [Avoid Illegal Parameter Ranges](#) on page 128
- [Update Verilog HDL and VHDL Type Mapping](#) on page 129

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 26. 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 27. 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: <pre>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</pre>	

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 28. 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: <code>set_global_assignment -name AUTO_DISCOVER_AND_SORT OFF</code>

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 29. 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.utilities_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:	
<pre>library myLib; use myLib.utilities_pack.all;</pre>	

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 30. 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: <code>set_parameter -name N 4</code>
Example RTL:	
<pre>set_parameter -entity mid_level -to my_ram -name SIZE 16</pre>	

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 31. Object Declaration Differences

Other Quartus Software Products	Intel Quartus Prime Pro Edition
From the Example RTL, synthesis initializes the output p_prog_iol with the value of p_prog_iol_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: <pre>1 output p_prog_iol = p_prog_iol_reg; 2 reg p_prog_iol_reg;</pre>	

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 32. 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 Edition synthesis honors SystemVerilog features only in .sv files.
Example RTL: <pre>localparam num_mem_locations = 1050; wire mem_addr [\$clog2(num_mem_locations)-1 : 0];</pre>	

5.4.9. Avoid Assignment Mixing in Always Blocks

Intel Quartus Prime 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 Edition migration.

Table 33. 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 <code>ALWAYS</code> block.

5.4.10. Avoid Unconnected, Non-Existent Ports

Intel Quartus Prime 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 Edition migration.

Table 34. 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 <code>ALWAYS</code> block.

5.4.11. Avoid Illegal Parameter Ranges

Intel Quartus Prime 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	Added information about removing assignments from the qsf file that point to legacy output files.
2018.05.07	18.0.0	First release as separate chapter of <i>Getting Started User Guide</i> .
2017.11.06	17.1.0	<ul style="list-style-type: none">Added Verilog HDL Macro example.Updated for latest Intel branding conventions.
2017.05.08	17.0.0	<ul style="list-style-type: none">Removed statement about limitations for safe state machines. The Compiler supports safe state machines. State machine inference is enabled by default.
2016.10.31	16.1.0	<ul style="list-style-type: none">Implemented Intel rebranding.Described unsupported Intel Quartus Prime Standard Edition physical synthesis options.Changed title from "Remove Filling Vectors" to "Remove Unsized Constant".
2016.05.03	16.0.0	<ul style="list-style-type: none">Added topic on Safe State Machine encoding.Corrected statement about Verilog Compilation Unit.Corrected typo in Modify Entity Name Assignments.Clarified limitations for multiple Logic Lock instances in the same region.
2015.11.02	15.1.0	<ul style="list-style-type: none">First version of document.

6. Document Archives

For the latest and previous versions of this user guide, refer to [Intel Quartus Prime Pro Edition User Guide: Getting Started](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

A. Intel Quartus Prime Pro Edition User Guides

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

Related Information

- [Intel Quartus Prime Pro Edition User Guide: Getting Started](#)
Introduces the basic features, files, and design flow of the Intel Quartus Prime Pro Edition software, including managing Intel Quartus Prime Pro Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- [Intel Quartus Prime Pro Edition User Guide: Platform Designer](#)
Describes creating and optimizing systems using Platform Designer, a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- [Intel Quartus Prime Pro Edition User Guide: Design Recommendations](#)
Describes best design practices for designing FPGAs with the Intel Quartus Prime Pro Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Pro Edition synthesis optimally implements your design in hardware.
- [Intel Quartus Prime Pro Edition User Guide: Design Compilation](#)
Describes set up, running, and optimization for all stages of the Intel Quartus Prime Pro Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- [Intel Quartus Prime Pro Edition User Guide: Design Optimization](#)
Describes Intel Quartus Prime Pro 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, optimizing device resource usage, device floorplanning, and implementing engineering change orders (ECOs).
- [Intel Quartus Prime Pro Edition User Guide: Programmer](#)
Describes operation of the Intel Quartus Prime Pro 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 Pro Edition User Guide: Block-Based Design](#)
Describes block-based design flows, also known as modular or hierarchical design flows. These advanced flows enable preservation of design blocks (or logic that comprises a hierarchical design instance) within a project, and reuse of design blocks in other projects.

- [Intel Quartus Prime Pro 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 Pro Edition User Guide: Third-party Simulation](#)
Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Siemens EDA, 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 Pro Edition User Guide: Third-party Synthesis](#)
Describes support for optional synthesis of your design in third-party synthesis tools by Siemens EDA, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- [Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools](#)
Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin*.
- [Intel Quartus Prime Pro Edition User Guide: Debug Tools](#)
Describes a portfolio of Intel Quartus Prime Pro 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, system debugging toolkits, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- [Intel Quartus Prime Pro Edition User Guide: Timing Analyzer](#)
Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro 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 Pro Edition User Guide: Power Analysis and Optimization](#)
Describes the Intel Quartus Prime Pro 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 Pro 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 Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- [Intel Quartus Prime Pro Edition User Guide: PCB Design Tools](#)
Describes support for optional third-party PCB design tools by Siemens EDA and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- [Intel Quartus Prime Pro Edition User Guide: Scripting](#)
Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.