Introduction to Intel® FPGA IP Cores

Updated for Intel® Quartus® Prime Design Suite: **23.1**

Online Version
Send Feedback

UG-01056

ID: 683102
Version: 2023.04.03

Contents

1. Introduction to Intel® FPGA IP Cores

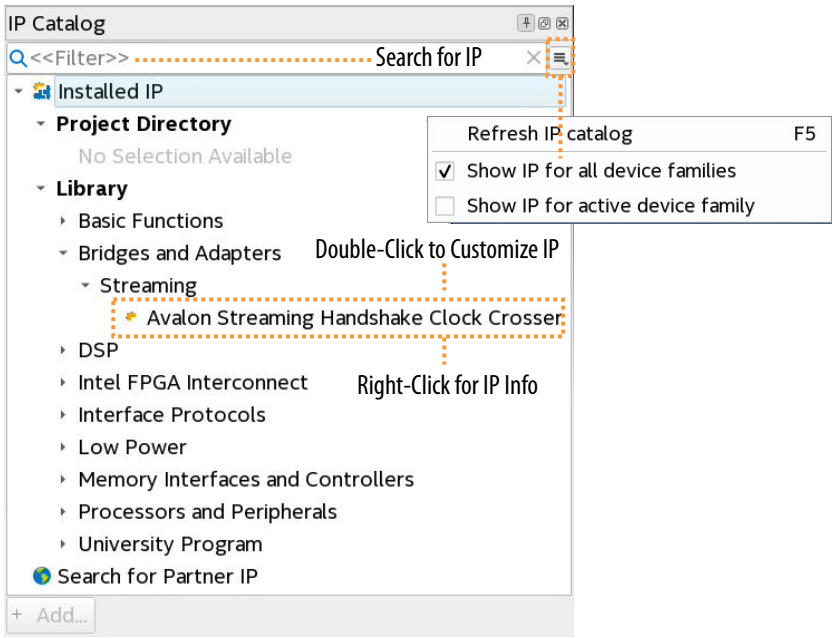
Intel and strategic IP partners offer a broad portfolio of configurable IP cores optimized for Intel FPGA devices.

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 1. Intel FPGA IP Catalog



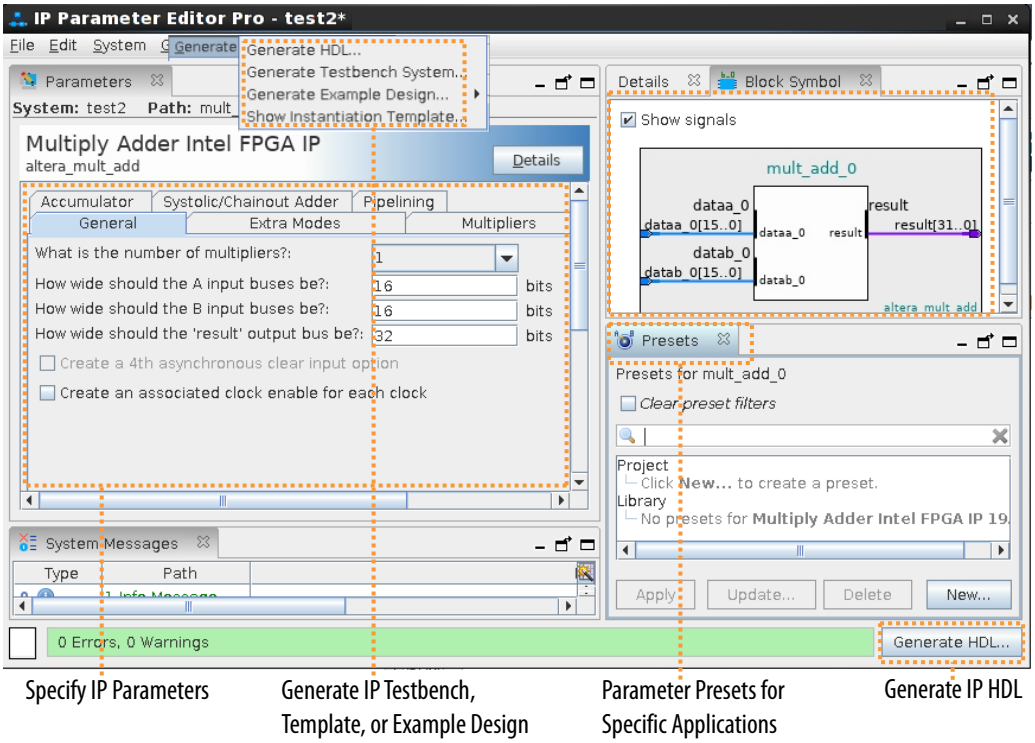
1.1. IP Catalog and Parameter Editor

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.(1)

Figure 2. Example IP Parameter Editor



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

1.1.1. 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 (Standard)* for information on use of IP in Platform Designer (Standard) and Platform Designer, respectively.

Related Information

Creating a System with Platform Designer

1.2. 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 3. IP Core Installation Path

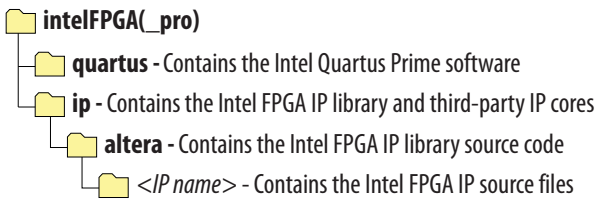


Table 1. IP Core Installation Locations

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.

1.2.1. 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.

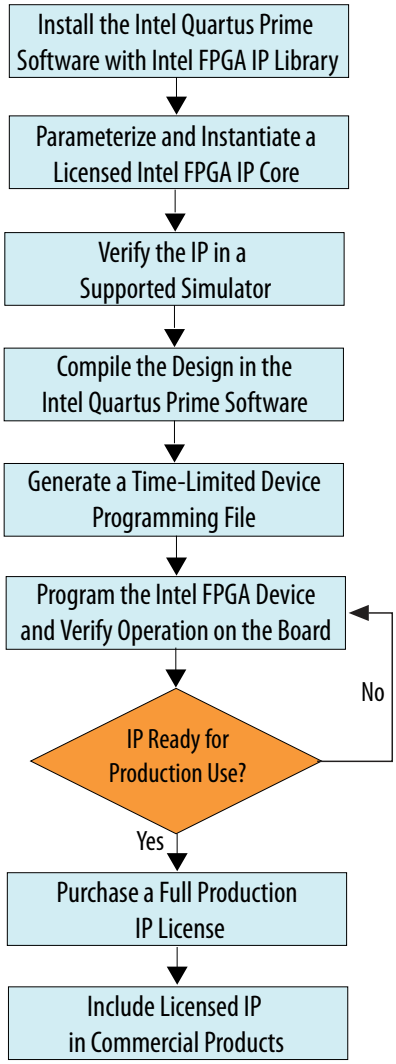
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **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 4. 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

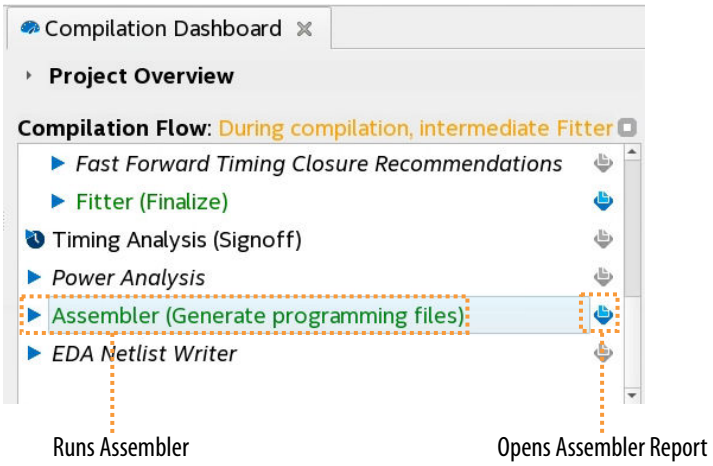
1.2.2. Checking the IP License Status

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:

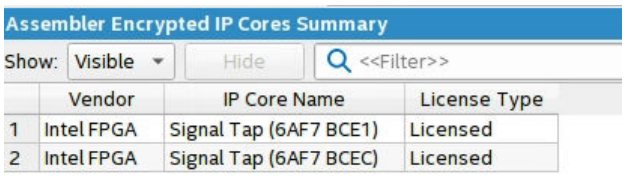
1. Click **Assembler** on the Compilation Dashboard.
2. When the Assembler (and any prerequisite stages of compilation) complete, click the **Report** icon for the Assembler in the Compilation Dashboard.

Figure 5. Assembler Report Icon in Compilation Dashboard



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

Figure 6. Encrypted IP Cores Summary Report



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

1. Type the following command:

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

2. 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     ;
+--------+-------------------------------------------+--------------+
```

1.2.3. 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:

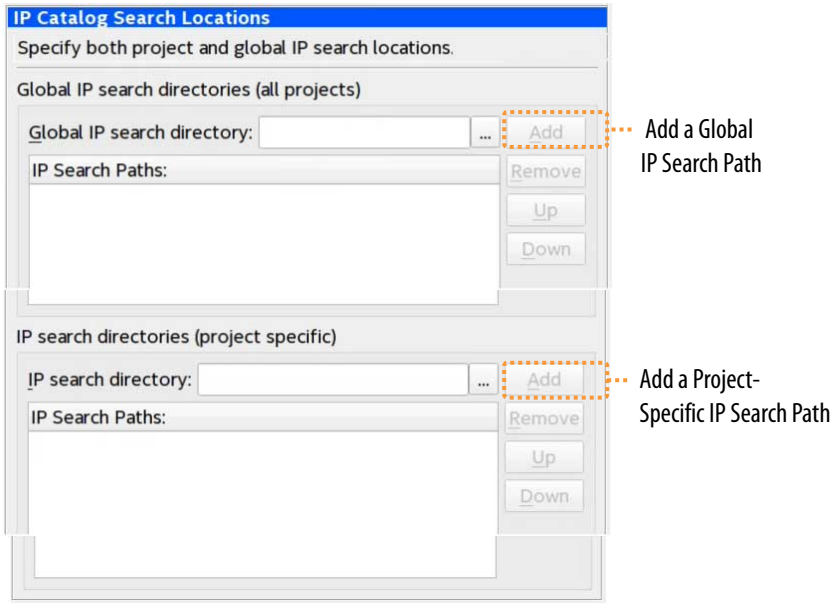
- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

1.2.4. 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.

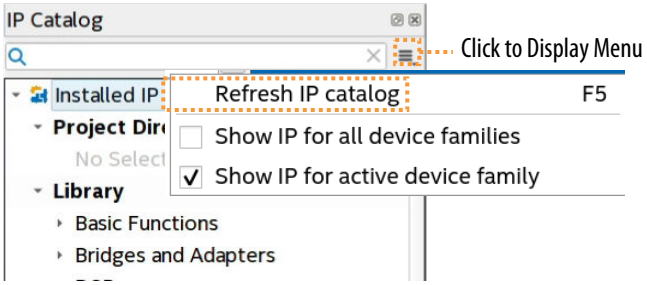
Follow these steps to add custom or third-party IP to the IP Catalog:

Figure 7. 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 (Standard), or click **File ➤ Refresh System** in Platform Designer (Standard).

Figure 8. Refreshing IP Catalog



1.3. Best Practices for Intel FPGA IP

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

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

  Note: When generating IP cores, do not generate files into a directory that has a space in the directory name or path. Spaces are not legal characters for IP core paths or names.
- When you generate an IP core using the IP Catalog, the Intel Quartus Prime software generates a `.qsys` (for Platform Designer (Standard)-generated IP cores) or a `.ip` file (for Intel Quartus Prime Pro Edition) or a `.qip` file. The Intel Quartus Prime Pro Edition software automatically adds the generated `.ip` to your project. In the Intel Quartus Prime Standard Edition software, add the `.qip` to your project. Do not add the parameter editor generated file (`.v` or `.vhd`) to your design without the `.qsys` or `.qip` file. Otherwise, you cannot use the IP upgrade or IP parameter editor feature.
- Plan your directory structure ahead of time. Do not change the relative path between a `.qsys` file and it's generation output directory. If you must move the `.qsys` file, ensure that the generation output directory remains with the `.qsys` file.
- Do not add IP core files directly from the `/quartus/libraries/megafunctions` directory in your project. Otherwise, you must update the files for each subsequent software release. Instead, use the IP Catalog and then add the `.qip` to your project.

- Do not use IP files that the Intel Quartus Prime software generates for RAM or FIFO blocks targeting older device families (even though the Intel Quartus Prime software does not issue an error). The RAM blocks that Intel Quartus Prime generates for older device families are not optimized for the latest device families.
- When generating a ROM function, save the resulting `.mif` or `.hex` file in the same folder as the corresponding IP core's `.qsys` or `.qip` file. For example, moving all of your project's `.mif` or `.hex` files to the same directory causes relative path problems after archiving the design.
- Always use the Intel Quartus Prime `ip-setup-simulation` and `ip-make-simscript` utilities to generate simulation scripts for each IP core or Platform Designer (Standard) system in your design. These utilities produce a single simulation script that does not require manual update for upgrades to Intel Quartus Prime software or IP versions, as Simulating Intel FPGA IP Cores on page 24 describes.

1.4. IP General Settings

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

Table 2. 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**

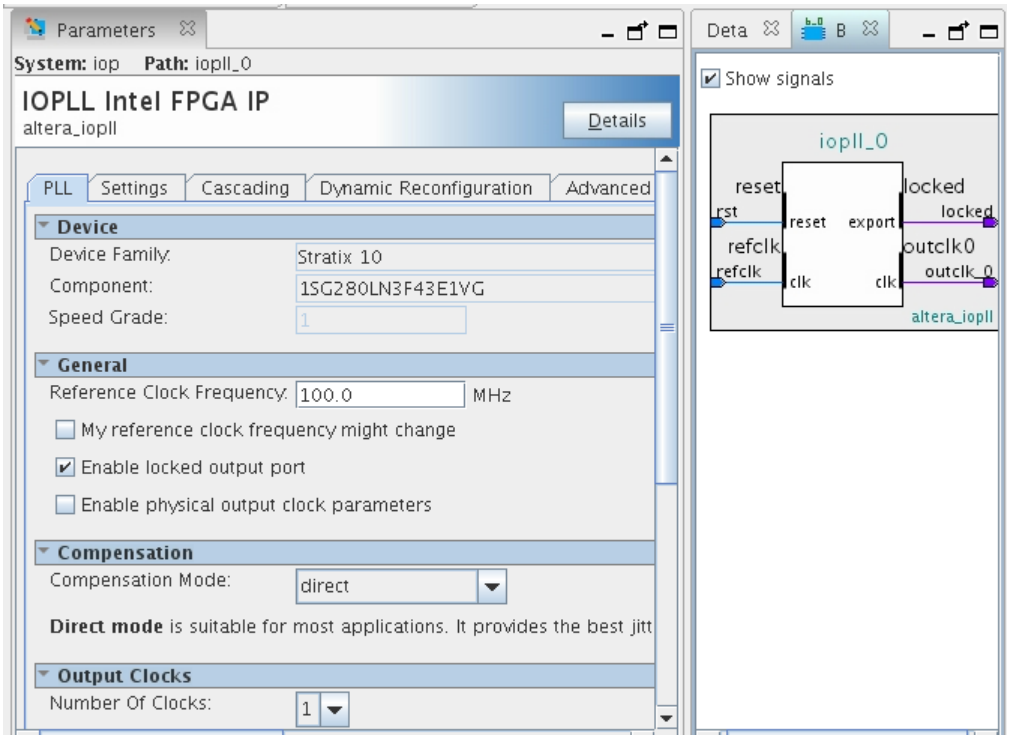
1.5. Specifying the IP Core Parameters and Options (Intel Quartus Prime Pro Edition)

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 9. 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*.

Related Information

- Generating IP Cores (Intel Quartus Prime Standard Edition) on page 16
- Generating a Combined Simulator Setup Script on page 28

1.5.1. 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.

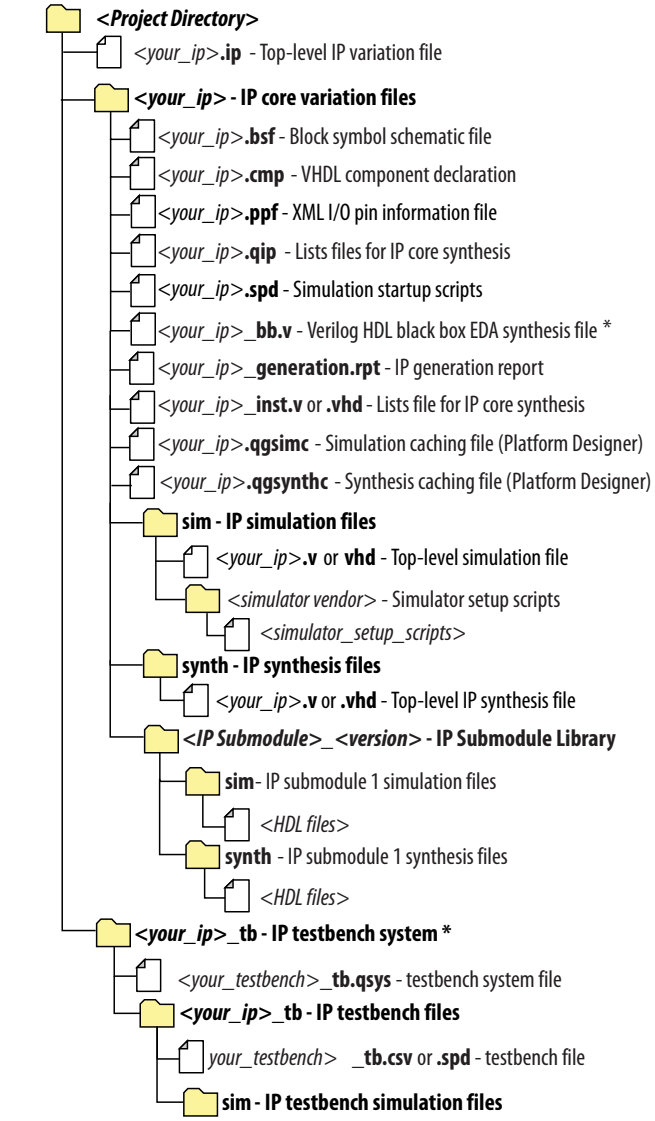
Table 3. Output Files of Intel FPGA IP Generation

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*>.qip	Contains all information to integrate and compile the IP component.
<*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 `.sopcinfo` 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.
/cadence	Contains a shell script `ncsim_setup.sh` and other setup files to set up and run an NCSim 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 10. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)



* If supported and enabled for your IP core variation.

1.5.2. 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
```

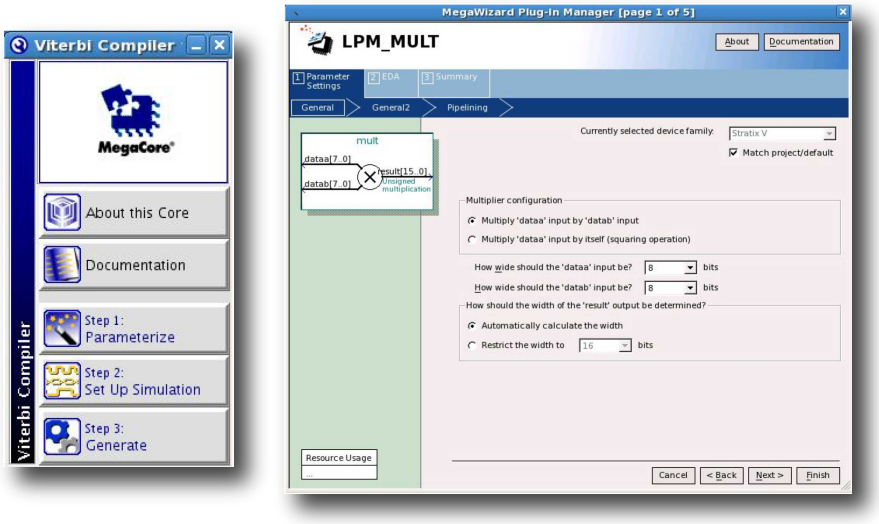
2. Run `qsys-generate` to generate the IP core variation:

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

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

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

Figure 11. Legacy Parameter Editors



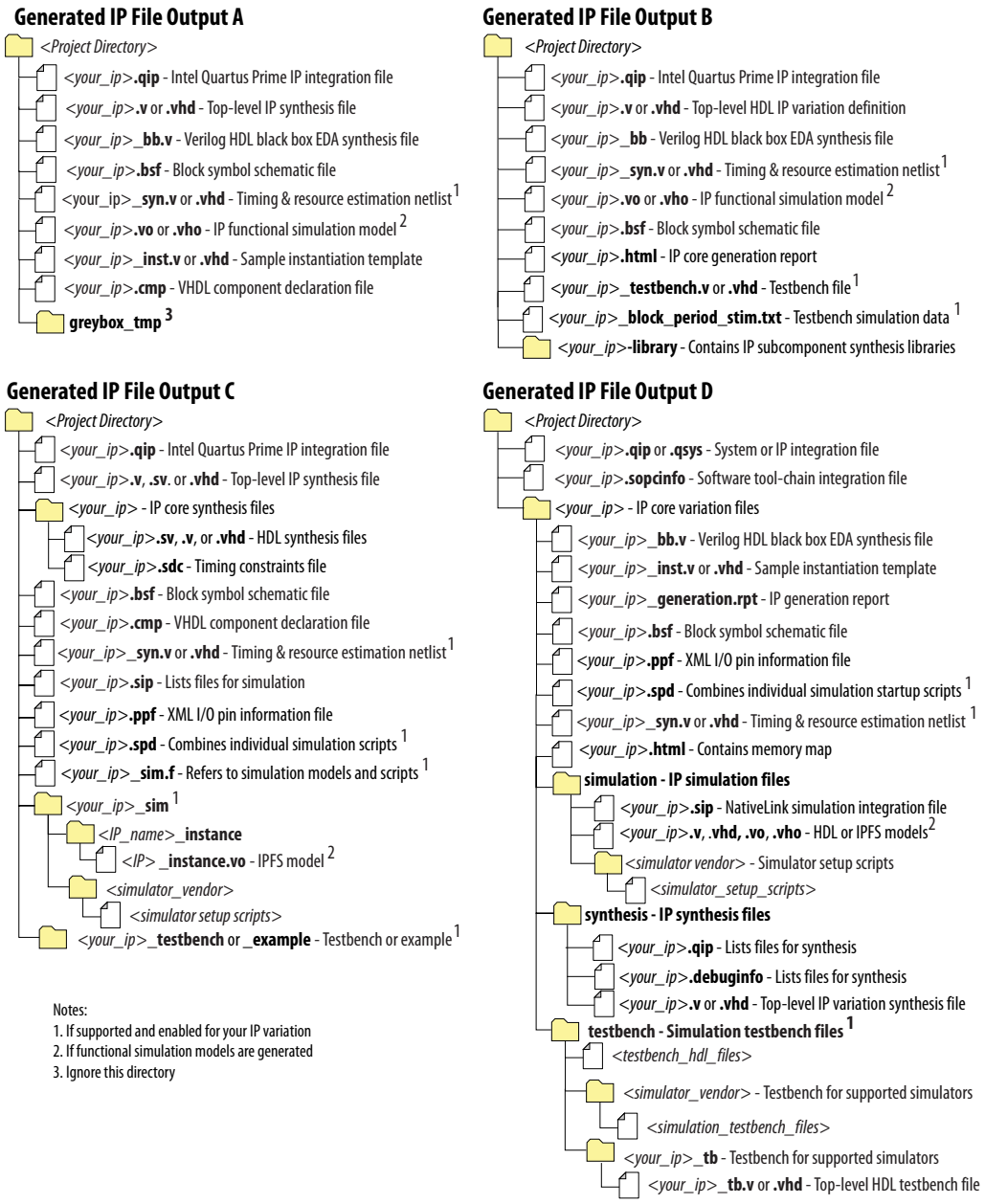
1. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name and output HDL file type for your IP variation. This name identifies the IP core variation files in your project. Click **OK**. Do not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

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

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

The Intel Quartus Prime Standard Edition legacy parameter editors generate one of the following output file structures for individual IP cores:

Figure 12. IP Core Generated Files (Legacy Parameter Editors)



1.7. 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.

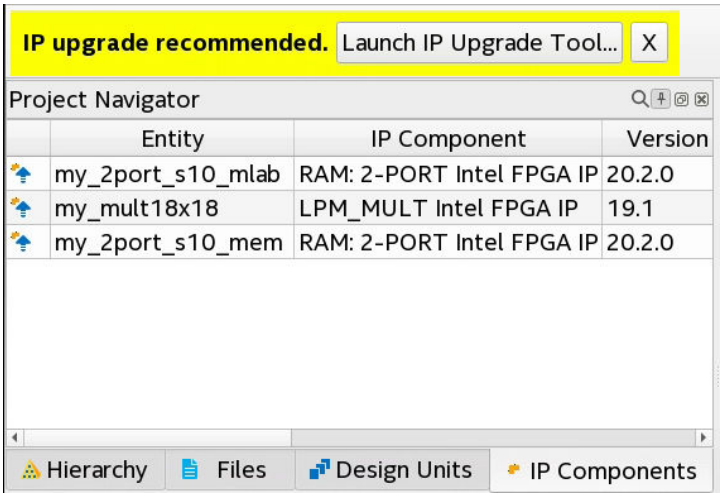
Table 4. Modifying an IP Variation

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

1.8. 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 13. 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.

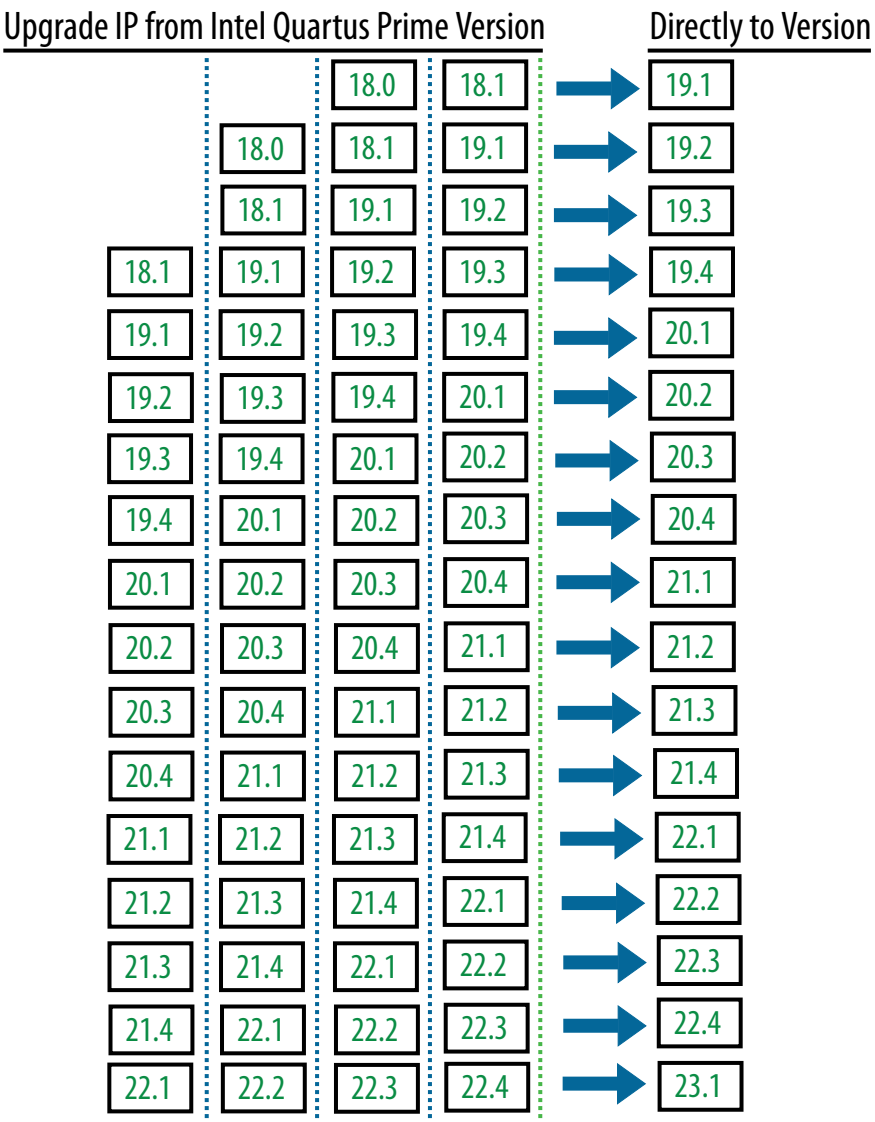
Note: Upgrading IP cores may append a unique identifier to the original IP core entity names, without similarly modifying the IP instance name. There is no requirement to update these entity references in any supporting Intel Quartus Prime file, such as the Intel Quartus Prime Settings File (`.qsf`), 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 5. 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 Upgrade Optional 	Indicates that upgrade is optional for this IP variation in the current version of the Intel Quartus Prime software. You can upgrade this IP variation to take advantage of the latest development of this IP core. Alternatively, you can retain previous IP core characteristics by declining to upgrade. Refer to the Description for details about IP core version differences. If you do not upgrade the IP, the IP variation synthesis and simulation files are unchanged and you cannot modify parameters until upgrading.
IP Upgrade Required 	Indicates that you must upgrade the IP variation before compiling in the current version of the Intel Quartus Prime software. Refer to the Description for details about IP core version differences.
IP Upgrade Unsupported 	Indicates that upgrade of the IP variation is not supported in the current version of the Intel Quartus Prime software due to incompatibility with the current version of the Intel Quartus Prime software. The Intel Quartus Prime software prompts you to replace the unsupported IP core with a supported equivalent IP core from the IP Catalog. Refer to the Description for details about IP core version differences and links to Release Notes.
IP End of Life 	Indicates that Intel designates the IP core as end-of-life status. You may or may not be able to edit the IP core in the parameter editor. Support for this IP core discontinues in future releases of the Intel Quartus Prime software.
IP Upgrade Mismatch Warning 	Provides warning of non-critical IP core differences in migrating IP to another device family.
IP has incompatible subcores 	Indicates that the current version of the Intel Quartus Prime software does not support compilation of your IP variation, because the IP has incompatible subcores.
Compilation of IP Not Supported 	Indicates that the current version of the Intel Quartus Prime software does not support compilation of your IP variation. This can occur if another edition of the Intel Quartus Prime software, 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 14. Intel Quartus Prime Pro Edition IP Version Upgrade Paths

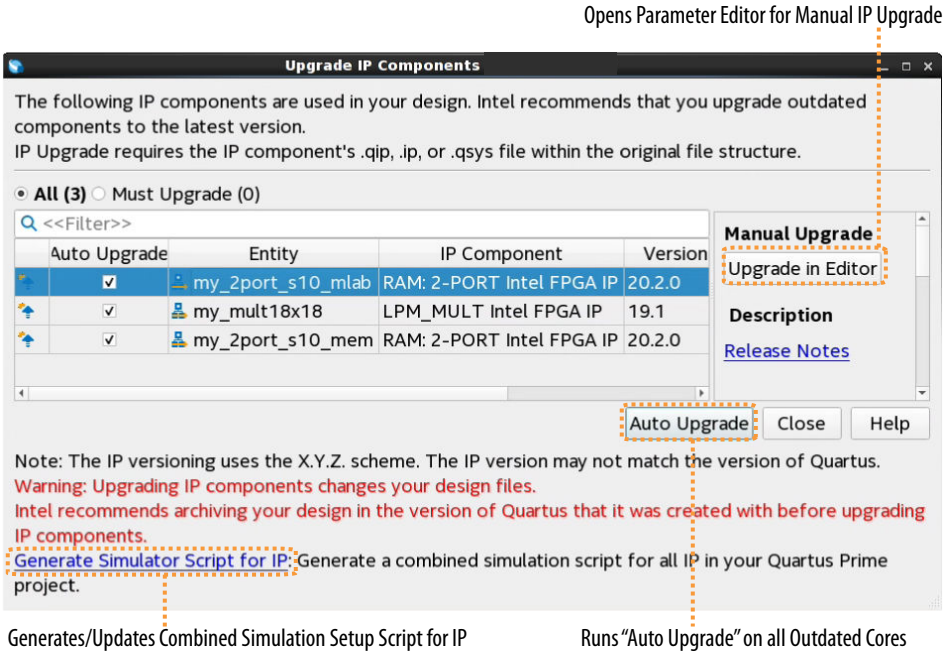


Follow these steps to upgrade IP cores:

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 15. Upgrading IP Cores (Intel Quartus Prime Pro Edition Example)



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

1.8.1. Upgrading IP Cores at Command-Line

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

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

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

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

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

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

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

1.8.2. Migrating IP Cores to a Different Device

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

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

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

Related Information

Intel FPGA IP Release Notes

1.8.3. 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:

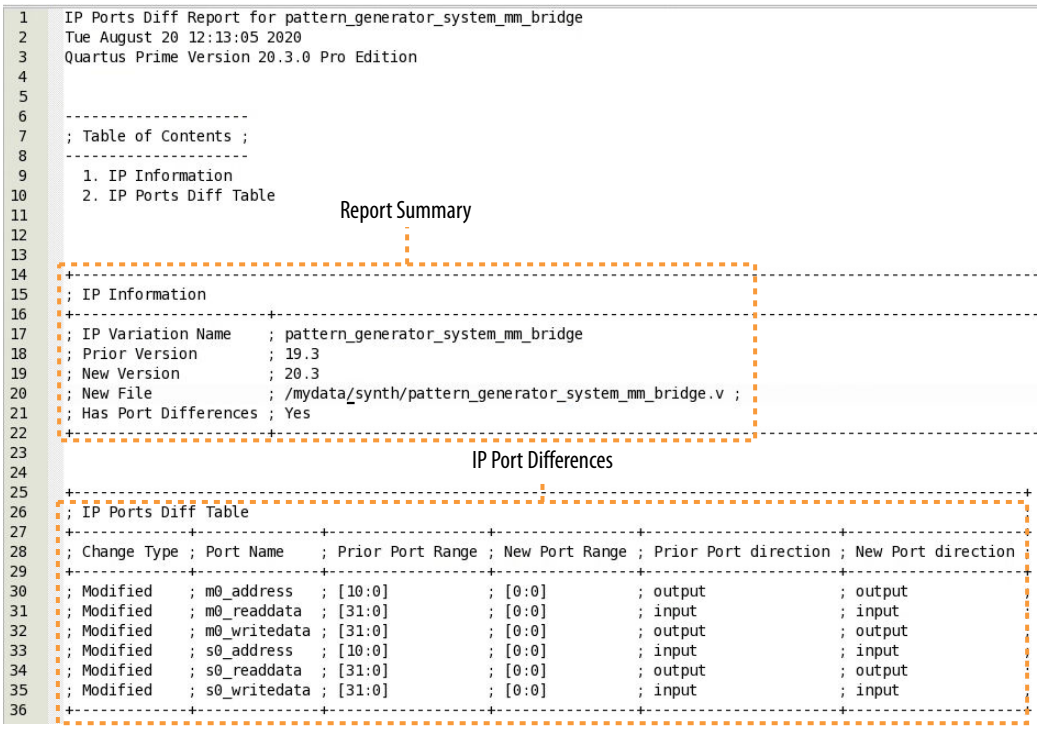
Table 6. IP Upgrade Error Information

Upgrade IP Components Field	Description
Status	Displays the "Success" or "Failed" status of each upgrade or migration. Click the status of any upgrade that fails to open the **IP Upgrade Report**.
Version	Dynamically updates the version number when upgrade is successful. The text is red when the IP requires upgrade.
Device Family	Dynamically updates to the new device family when migration is successful. The text is red when the IP core requires upgrade.
Auto Upgrade	Runs automatic upgrade on all IP cores that support auto upgrade. Also, automatically generates a `<Project Directory>/ip_upgrade_port_diff_reports` 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 16. IP Port Differences Report




```
IP Ports Diff Report for pattern_generator_system_mm_bridge
Tue August 20 12:13:05 2020
Quartus Prime Version 20.3.0 Pro Edition


--------------------
; Table of Contents ;
--------------------
  1. IP Information
  2. IP Ports Diff Table

+-------------------------------------------------------------------------------------------------------------------------------+
; IP Information                                                                                                                 ;
+----------------------+--------------------------------------------------------------------------------------------------------+
; IP Variation Name    ; pattern_generator_system_mm_bridge                                                                       ;
; Prior Version        ; 19.3                                                                                                    ;
; New Version          ; 20.3                                                                                                    ;
; New File             ; /mydata/synth/pattern_generator_system_mm_bridge.v ;
; Has Port Differences ; Yes                                                                                                    ;
+----------------------+--------------------------------------------------------------------------------------------------------+

+-------------------------------------------------------------------------------------------------------------------------------+
; IP Ports Diff Table                                                                                                            ;
+-------------+--------------+-----------------+---------------+---------------------+---------------------+
; Change Type ; Port Name    ; Prior Port Range ; New Port Range ; Prior Port direction ; New Port direction ;
+-------------+--------------+-----------------+---------------+---------------------+---------------------+
; Modified    ; m0_address   ; [10:0]          ; [0:0]         ; output              ; output              ;
; Modified    ; m0_readdata  ; [31:0]          ; [0:0]         ; input               ; input               ;
; Modified    ; m0_writedata ; [31:0]          ; [0:0]         ; output              ; output              ;
; Modified    ; s0_address   ; [10:0]          ; [0:0]         ; input               ; input               ;
; Modified    ; s0_readdata  ; [31:0]          ; [0:0]         ; output              ; output              ;
; Modified    ; s0_writedata ; [31:0]          ; [0:0]         ; input               ; input               ;
+-------------+--------------+-----------------+---------------+---------------------+---------------------+
```

1.9. 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.

1.9.1. Supported Simulators

The Intel Quartus Prime software does not include a native simulator, but provides support for the specific RTL- and gate-level EDA simulators in Intel Quartus Prime Pro Edition Supported Simulators.

Intel also provides the Questa* - Intel FPGA Edition simulator, a version of the Questa Advanced simulator targeted for Intel FPGA devices. 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.

Table 7. Intel Quartus Prime Pro Edition Supported Simulators

Vendor	Simulator	Version	Platform	Supports Siemens EDA Verification IP
Aldec	Active-HDL*	13.0	Windows 64-bit	No
Aldec	Riviera-PRO	2021.10	Windows, Linux, 64-bit	No
Cadence	Xcelium Parallel Simulator	21.09.003	Linux 64-bit	Yes
Intel	Questa Intel FPGA Edition	2022.1	Windows, Linux, 64-bit	Yes
Siemens EDA	QuestaSim* Simulator(2)	2021.4	Windows, Linux, 64-bit	Yes
Synopsys	VCS, VCS MX	S-2021.09-1	Linux 64-bit	Yes

Table 8. Intel Quartus Prime Standard Edition Supported Simulators

Vendor	Simulator	Version	Platform
Aldec	Active-HDL	10.3	Windows
Aldec	Riviera-PRO	2015.10	Windows, Linux
Cadence	Incisive Enterprise*	14.20	Linux
Siemens EDA	Questa Intel FPGA Edition	10.5b	Windows, Linux
Siemens EDA	ModelSim PE	10.4d	Windows
Siemens EDA	ModelSim SE	10.4d	Windows, Linux
Siemens EDA	QuestaSim	10.4d	Windows, Linux
Synopsys	VCS VCS MX	2014,12-SP1	Linux

1.9.2. Supported Simulation Flows

The Intel Quartus Prime software supports scripted and specialized simulation flows.

Table 9. Simulation Flows

Simulation Flow	Description
Scripted Simulation Flows	Scripted simulation supports custom control of all aspects of simulation, such as custom compilation commands, or multipass simulation flows. Use a version-independent top-level simulation script that sources Intel Quartus Prime-generated IP simulation setup scripts. The Intel Quartus Prime software can generate a combined simulator setup script for all IP cores, for each supported simulator.
NativeLink Simulation Flow	NativeLink automates Intel Quartus Prime integration with your EDA simulator. Setup NativeLink to generate simulation scripts, compile simulation libraries, and automatically launch your simulator following design compilation. Specify your own compilation, elaboration, and simulation scripts for testbench and simulation model files. Do not use NativeLink if you require direct control over every aspect of simulation.

continued...

(2) Also supports ModelSim* SE, Questa Advanced, Core, Prime, and Ultra variants.

Simulation Flow	Description
	Note: The Intel Quartus Prime Pro Edition software does not support NativeLink simulation.
Specialized Simulation Flows	Specialized simulation flows support various design scenarios: • For simulation of example designs, refer to the example design pr IP documentation. • For simulation of Platform Designer designs, refer to *Simulating Platform Designer (Standard) Systems* in *Intel Quartus Prime Pro Edition User Guide: Platform Designer*. • For simulation of the Nios® II processor, refer to *AN 351: Simulating Nios II Embedded Processor Designs*.

Related Information

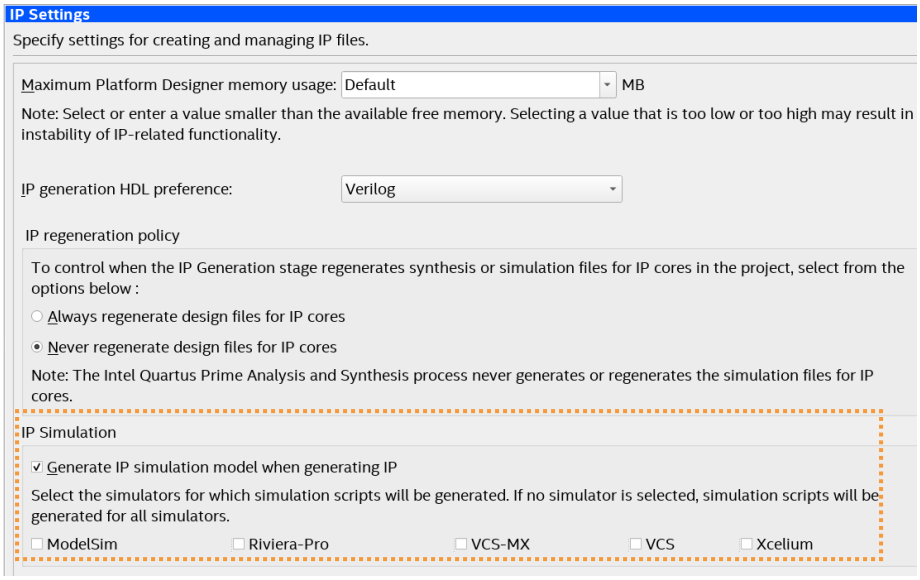
- Scripting IP Simulation on page 28
- Generating a Combined Simulator Setup Script on page 28

1.9.3. 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:

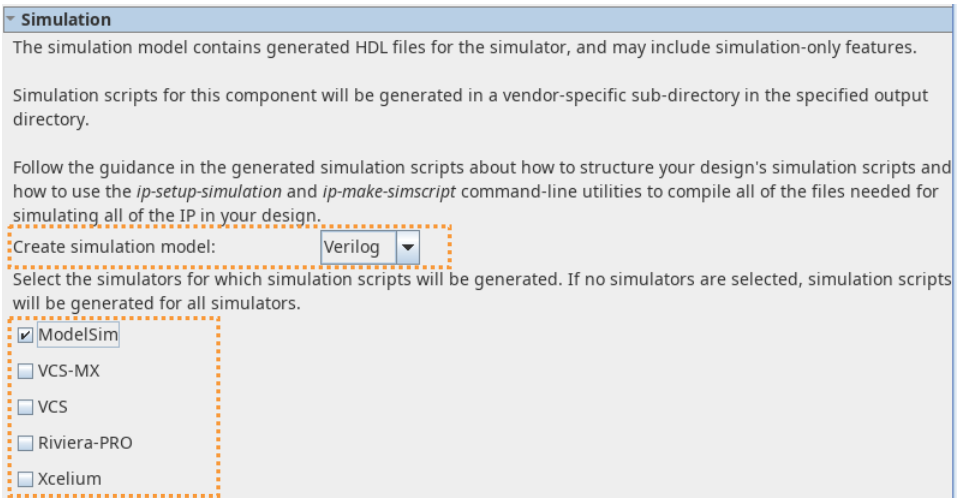
1. To specify your supported simulator and options for design simulation file generation, click **Assignment ➤ Settings ➤ EDA Tool Settings ➤ Simulation**.
2. To specify your supported simulator and options for IP simulation file generation, click **Assignments ➤ Settings ➤ IP Settings** and specify 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 17. 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 18. Simulation Options in Generation Dialog Box



Related Information

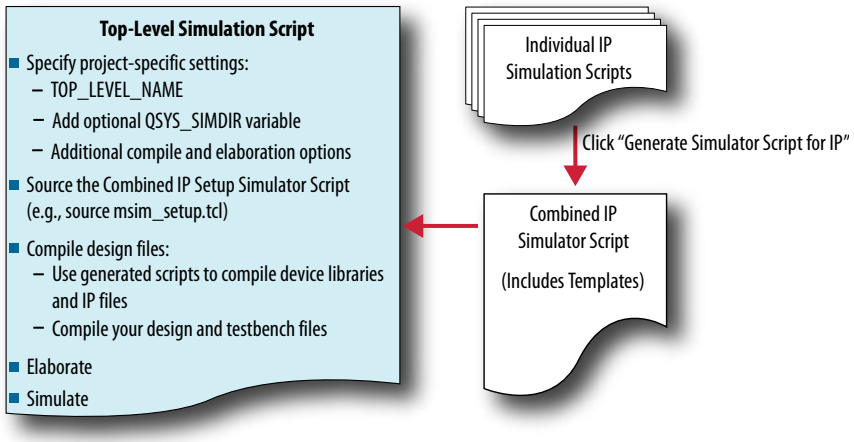
- Generating IP Functional Simulation Models (Intel Quartus Prime Standard Edition) on page 38
- Specifying the IP Core Parameters and Options (Intel Quartus Prime Pro Edition) on page 11

1.9.4. 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 19. 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.

1.9.4.1. 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 10. ip-setup-simulation Utility

Utility	Syntax

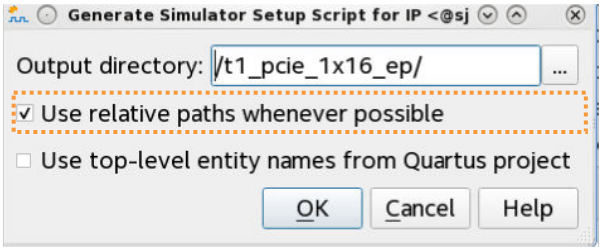
`ip-setup-simulation` generates a combined, version-independent simulation script for all Intel FPGA IP cores in your project. The command also automates regeneration of the script after upgrading software or IP versions. Use the `compile-to-work` option to compile all simulation files into a single work library if your simulation environment requires. Use the `--use-relative-paths` option to use relative paths whenever possible.	`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`.

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.

Figure 20. 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:

(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.

- 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.

Related Information

- Sourcing Aldec ActiveHDL or Riviera Pro Simulator Setup Scripts on page 30
- Sourcing Cadence Incisive Simulator Setup Scripts on page 31
- Sourcing ModelSim or QuestaSim Simulator Setup Scripts on page 34
- Sourcing Synopsys VCS Simulator Setup Scripts on page 35
- Sourcing Synopsys VCS MX Simulator Setup Scripts on page 36
- Specifying the IP Core Parameters and Options (Intel Quartus Prime Pro Edition) on page 11

1.9.4.1.1. 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
```

1.9.4.1.2. 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`.

1.9.4.1.3. 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`.

1.9.4.1.4. 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
```

1.9.4.1.5. 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`.

1.9.4.1.6. 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`.

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

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

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

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

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

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

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

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

Table 11. Execution Paths for EDA Simulators

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

continued...

Simulator	Path
	<*simulator install path*>/bin (Linux)

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

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

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

To generate IP functional simulation models:

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

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

1.10. 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.

1.10.1. 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.

1.10.1.1. Accessing HDL Code Templates

The Intel Quartus Prime software includes code examples or templates for inferred RAMs, ROMs, shift registers, arithmetic functions, and DSP functions optimized for Intel FPGA devices. To access HDL code templates to define these IP cores in HDL:

1. Open a file in the text editor.
2. Click **Edit ➤ Insert template**.
3. In the **Insert Template** dialog box, click the + icon to expand either the **Verilog HDL** category or the **VHDL** category, depending on the HDL you prefer.
4. Under **Full Designs**, expand the navigation tree to display the type of functions you want to infer.
5. Select the function to display the code in the Preview pane and click **Insert**.

1.10.1.1.1. Example Top-Level Verilog HDL Module

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

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

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

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

1.10.1.1.2. Example Top-Level VHDL Module

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

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

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

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

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

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

1.11. 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:

- If you are using the Intel Quartus Prime Pro Edition software version 19.3 or later, the public encryption key is in *<install directory>*\quartus\common\misc\public_key.
- If you are using a version of the Intel Quartus Prime Pro Edition software earlier than version 19.3, 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

1.12. Introduction to Intel FPGA IP Cores Archives

For the latest and previous versions of this user guide, refer to Introduction to Intel FPGA IP Cores. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

1.13. Introduction to Intel FPGA IP Cores Revision History

This document has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
2023.04.03	23.1	• Updated *Support for the IEEE 1735 Encryption Standard* topic for new installed location of public encryption key. • Updated *Intel Quartus Prime Pro Edition IP Version Upgrade Paths*.
2021.09.27	21.3	• Added support for Questa Intel FPGA Edition simulator. • Removed support for ModelSim - Intel FPGA Edition simulator. • Updated simulator support in *Simulator Support for Mentor Verification IP Bus Functional Models (BFMs)* topic. • Updated *Intel Quartus Prime Pro Edition IP Version Upgrade Paths* figure for latest versions. • Revised *Generating IP Simulation Files* topic for new simulation options.
2020.11.09	20.3	• 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	• Added details to "Support for the IEEE 1735 Encryption Standard." • Added "Intel FPGA IP Versioning" topic. • Added "Checking the IP License Status" topic.
2019.05.13	18.1	Updated heading title in "Support for the IEEE 1735 Encryption Standard."
2019.04.16	18.1	• Added "Introduction to Intel FPGA IP Cores Archives." • Updated the keyname and added --help information to "Support for the IEEE 1735 Encryption Standard."
2018.10.24	18.1	• Updated information about obtaining IEEE 1735 Encryption key.
2018.09.24	18.1	• Added statement that the Intel Quartus Prime software installer does not support spaces in the installation path. • Added "Intel FPGA IP Best Practices" topic.
2018.05.07	18.0	• Updated screenshots of IP Catalog and Parameter Editor for latest IP names. • Added note about Generate Combined Simulator Setup Scripts command limitations. • Added information about generation of simulation files for Xcelium* • Revised product branding for Intel standards. • Revised topics on Intel FPGA IP Evaluation Mode (formerly OpenCore). • Added note that IP core encryption is supported only in Intel Quartus Prime Pro Edition. • Revised product branding for Intel standards.

Date	Version	Changes
2016.10.31	16.1	• 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.
2016.08.07	16.0	• Intel rebranding.
2016.05.02	16.0	• Described **Generate Simulator Setup Script for IP** feature. • Add information about unique hash codes preventing name collisions. • Removed support for NativeLink in Pro Edition. • Updated all GUI descriptions and screenshots for latest version.
2016.02.05	15.1.1	• Corrected list of files ip-make-simscript generates. • Removed incorrect statement about running ip-make-simscript. • Revised Incorporating IP Simulation Scripts in Top-Level Scripts graphic.
2015.11.02	15.1.0	• Added Generating Version-Agnostic 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. • Updated Intel Quartus Prime product name throughout.
2015.05.04	15.0	• The latest version of the ModelSim-Altera software supports native, mixed language (VHDL/Verilog HDL/SystemVerilog) co-simulation of plain text HDL. • 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.
2014.12.1	14.1	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	• Added information about specifying parameters for IP cores targeting Arria 10 devices. • Added information about the latest IP output for Quartus II 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.

continued...

Date	Version	Changes
Aug 2014	13.1	• Changed title from *Introduction to Megafunctions* to *Introduction to Altera IP Cores*. • Increased scope of document to include updated information about licensing, customizing, upgrading, and simulating all Altera IP cores. • Replaced MegaWizard Plug-In Manager with IP Catalog information.
May 2014	14.0	
May 2013	13.0	• Reorganization of content into topics. • First tracking of changes in Document Revision History.