Introduction to Altera IP Cores

2015.05.04

UG-01056 Subscribe Send Feedback

Altera® and strategic IP partners offer a broad portfolio of off-the-shelf, configurable IP cores optimized for Altera devices. The Quartus® II software installation includes the Altera IP library. You can integrate optimized and verified Altera IP cores into your design to shorten design cycles and maximize performance. You can evaluate any Altera IP core in simulation and compilation in the Quartus II software. The Quartus II software also supports integration of IP cores from other sources. Use the IP Catalog to efficiently parameterize and generate synthesis and simulation files for a custom IP variation.

The Altera IP library includes the following categories of IP cores:

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

Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera and other supported IP cores.

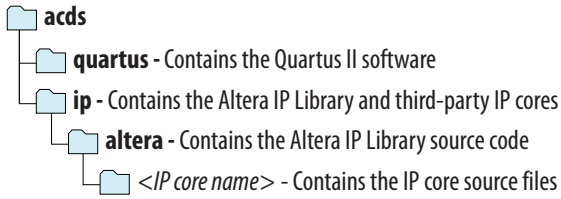
Related Information

- **IP User Guide Documentation**
- **Altera IP Release Notes**

Installing and Licensing IP Cores

The Altera IP Library provides many useful IP core functions for your production use without purchasing an additional license. Some Altera MegaCore® IP functions require that you purchase a separate license for production use. However, the OpenCore® feature allows evaluation of any Altera IP core in simulation and compilation in the Quartus II software. After you are satisfied with functionality and perffformance, visit the Self Service Licensing Center to obtain a license number for any Altera product.

Figure 1: IP Core Installation Path



Note: The default IP installation directory on Windows is **<*drive*>:\altera**<*version number*>; on Linux it is <*home directory*>**/altera/** <*version number*>.

Related Information

- **Adding IP Cores to IP Catalog** on page 4
- **Altera Licensing Site**
- **Altera Software Installation and Licensing Manual**

OpenCore Plus IP Evaluation

Altera's free OpenCore Plus feature allows you to evaluate licensed MegaCore IP cores in simulation and hardware before purchase. You need only purchase a license for MegaCore IP cores if you decide to take your design to production. OpenCore Plus supports the following evaluations:

- Simulate the behavior of a licensed 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.

OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely. This requires a connection between your board and the host computer.

Note: All IP cores that use OpenCore Plus time out simultaneously when any IP core in the design times out.

IP Catalog and Parameter Editor

The Quartus II IP Catalog (**Tools** > **IP Catalog**) and parameter editor help you easily customize and integrate IP cores into your project. You can use the IP Catalog and parameter editor to select, customize, and generate files representing your custom IP variation.

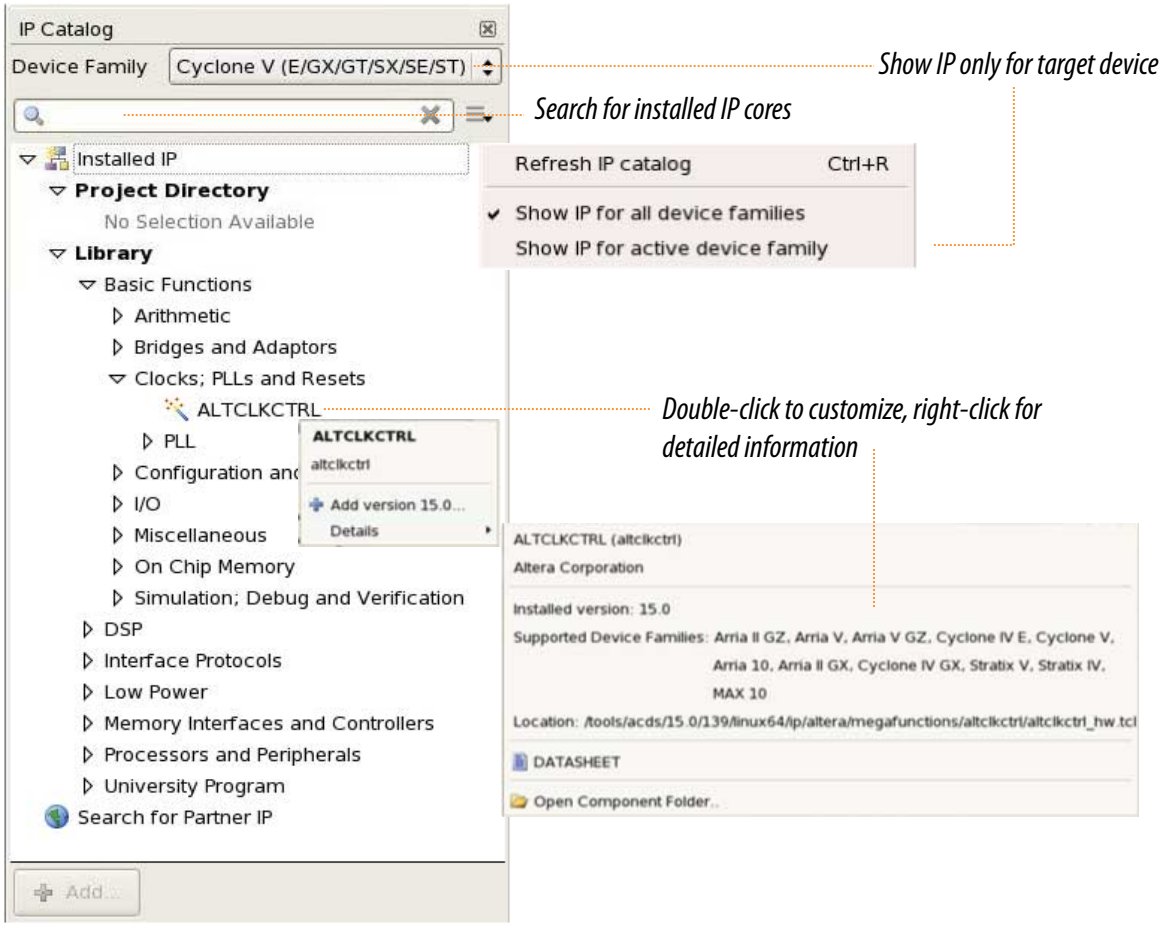
Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera IP cores.

The IP Catalog lists installed IP cores available for your design. Double-click any IP core to launch the parameter editor and generate files representing your IP variation. 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 Qsys system file (**.qsys**) or Quartus II IP file (**.qip**) representing the IP core in your project. You can also parameterize an IP variation without an open project.

Use the following features to help you quickly locate and select 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, open the IP core's installation folder, and view links to documentation.
- Click **Search for Partner IP**, to access partner IP information on the Altera website.

Figure 2: Quartus II IP Catalog



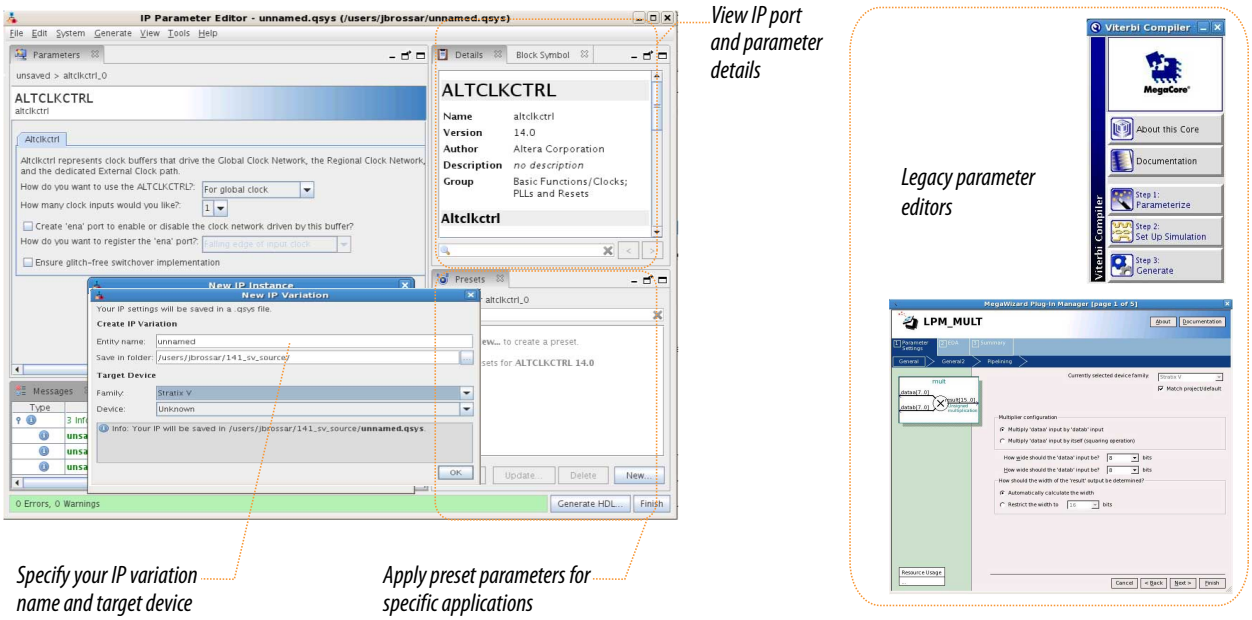
Note: The IP Catalog is also available in Qsys (**View** > **IP Catalog**). The Qsys IP Catalog includes exclusive system interconnect, video and image processing, and other system-level IP that are not available in the Quartus II IP Catalog. For more information about using the Qsys IP Catalog, refer to *Creating a System with Qsys* in the *Quartus II Handbook*.

Using the Parameter Editor

The parameter editor helps you to configure IP core ports, parameters, and output file generation options.

- Use preset settings in the parameter editor (where provided) to instantly apply preset parameter values for specific applications.
- View port and parameter descriptions, and links to documentation.
- Generate testbench systems or example designs (where provided).

Figure 3: IP Parameter Editors



Adding IP Cores to IP Catalog

The IP Catalog automatically displays Altera IP cores found in the project directory, in the Altera installation directory, and in the defined IP search path. The IP Catalog can include Altera-provided IP components, third-party IP components, custom IP components that you provide, and previously generated Qsys systems.

You can use the **IP Search Path** option (**Tools** > **Options**) to include custom and third-party IP components in the IP Catalog. The IP Catalog displays all IP cores in the IP search path. The Quartus II software searches the directories listed in the IP search path for the following IP core files:

- Component Description File (**_hw.tcl**)—Defines a single IP core.
- IP Index File (**.ipx**)—Each **.ipx** file indexes a collection of available IP cores, or a reference to other directories to search. In general, **.ipx** files facilitate faster searches.

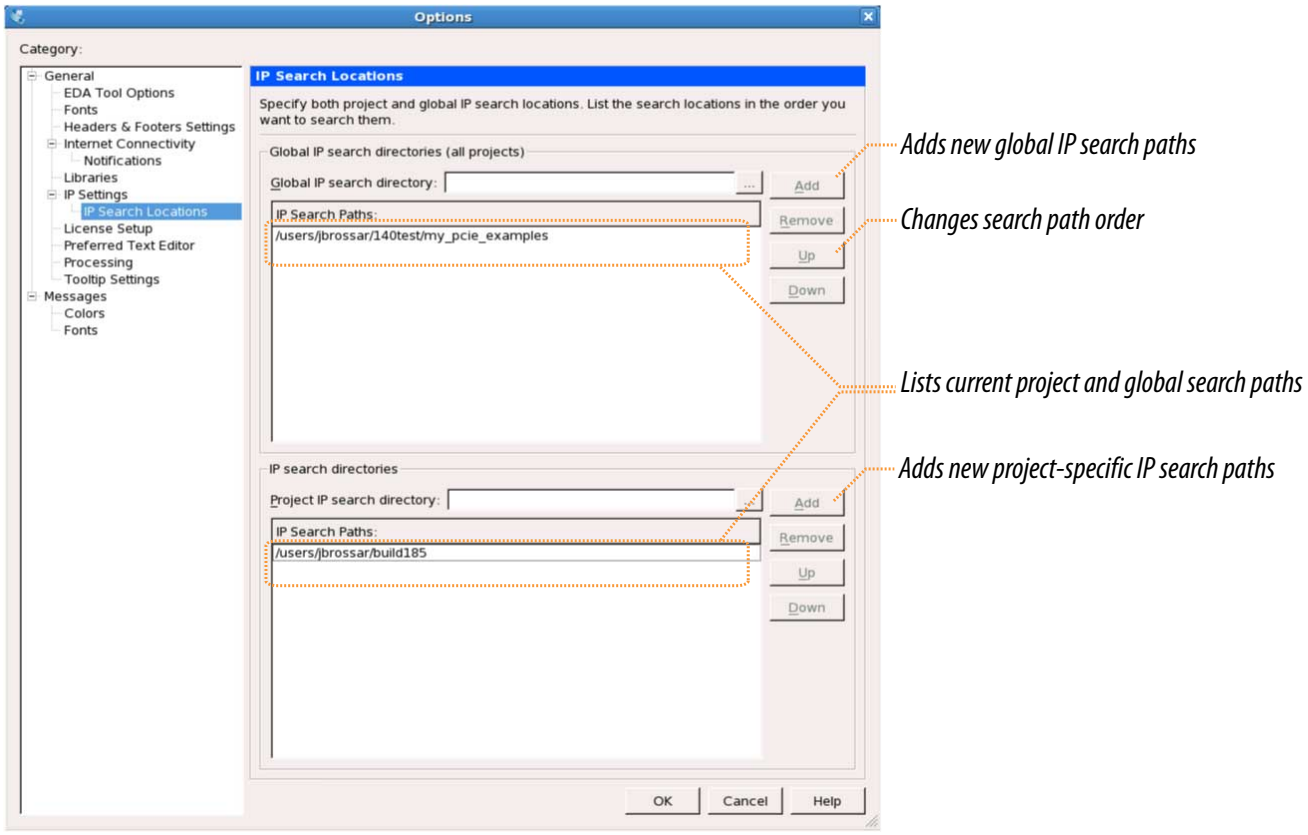
The Quartus II software searches some directories recursively and other directories only to a specific depth. When the search is recursive, the search stops at any directory that contains an **_hw.tcl** or **.ipx** file.

In the following list of search locations, a recursive descent is annotated by **. A single * signifies any file.

Table 1: IP Search Locations

Location	Description
PROJECT_DIR/*	Finds IP components and index files in the Quartus II project directory.
PROJECT_DIR/ip//***	Finds IP components and index files in any subdirectory of the **/ip** subdirectory of the Quartus II project directory.

Figure 4: Specifying IP Search Locations



If the Quartus II software recognizes two IP cores with the same name, the following search path precedence rules determine the resolution of files:

1. Project directory.
2. Project database directory.
3. Project IP search path specified in **IP Search Locations**, or with the `SEARCH_PATH` assignment in the Quartus II Settings File (**.qsf**) for the current project revision.
4. Global IP search path specified in **IP Search Locations**, or with the `SEARCH_PATH` assignment in the **quartus2.ini** file.
5. Quartus software libraries directory, such as **<*Quartus Installation*>\libraries**.

Note: If you add a component to the search path, you must refresh your system by clicking **File** > **Refresh** to update the IP Catalog.

General Settings for IP

You can use the following settings to control how the Quartus II software manages IP cores in your project.

Table 2: IP Core General Setting Locations

Setting Location	Description
Tools > **Options** > **IP Settings** Or **Assignments** > **Settings** > **IP Settings** (only enabled with open project)	• Specify your **IP generation HDL preference**. The parameter editor generates IP files in your preferred HDL by default. • Increase **Maximum Qsys memory usage size** if you experience slow processing for large systems, or if Qsys reports an Out of Memory error. • Specify whether to **Automatically add Quartus II IP files** to all projects. Disable this option to control addition of IP files manually. You may want to experiment with IP before adding to a project. • Use the **IP Regeneration Policy** setting to control when synthesis files are regenerated for each IP variation. Typically you **Always regenerate synthesis files for IP cores** after making changes to an IP variation.
Tools > **Options** > **IP Catalog Search Locations** Or **Assignments** > **Settings** > **IP Catalog Search Locations**	• Specify project and global IP search locations. The Quartus II software searches for IP cores in the project directory, in the Altera installation directory, and in the IP search path.
Assignments > **Settings** > **Simulation**	• **NativeLink Settings** allow you to automatically compile testbenches for supported simulators. You can also specify a script to compile the testbench, and a script to set up the simulation.

Specifying IP Core Parameters and Options

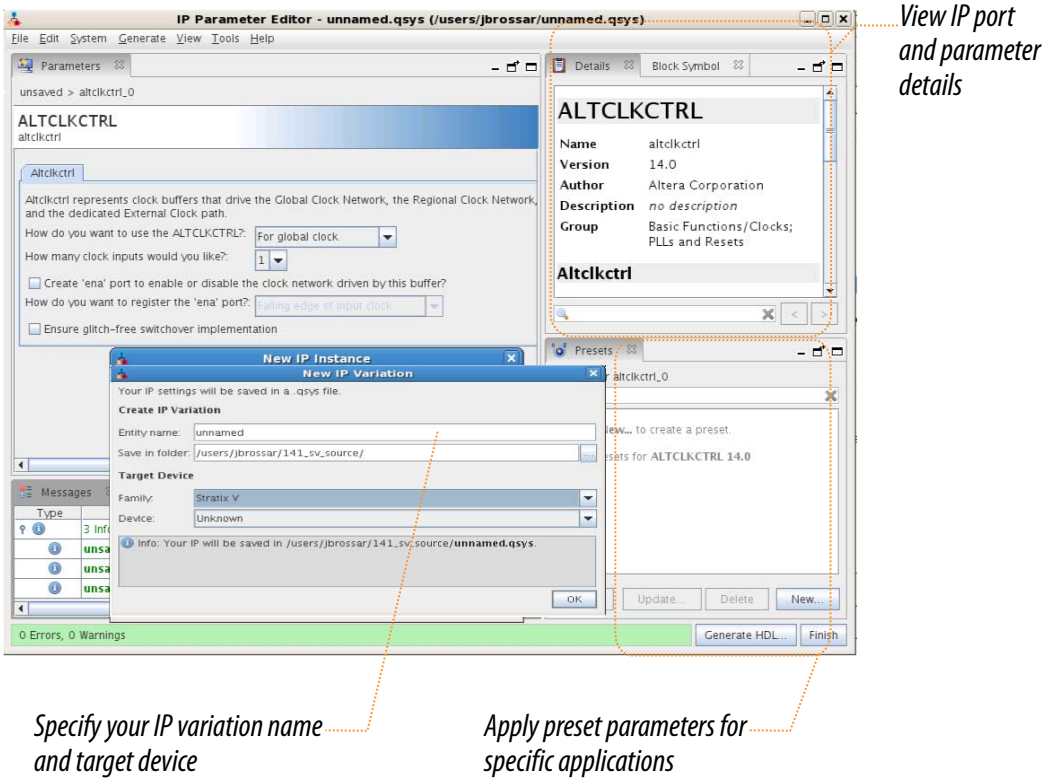
You can quickly configure a custom IP variation in the parameter editor. Use the following steps to specify IP core options and parameters in the parameter editor. Refer to *Specifying IP Core Parameters and Options (Legacy Parameter Editors)* for configuration of IP cores using the legacy parameter editor.

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 for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>**.qsys**. Click **OK**.
3. Specify the parameters and options for your IP variation in the parameter editor, including one or more of the following. Refer to your IP core user guide for information about specific IP core 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.

4. Click **Generate HDL**, the **Generation** dialog box appears.
5. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
6. To generate a simulation testbench, click **Generate** > **Generate Testbench System**.
7. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate** > **HDL Example**.
8. Click **Finish**. The parameter editor adds the top-level **.qsys** file to the current project automatically. If you are prompted to manually add the **.qsys** file to the project, click **Project** > **Add/Remove Files in Project** to add the file.
9. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Figure 5: IP Parameter Editor



Related Information

Specifying IP Core Parameters and Options (Legacy Parameter Editors) on page 10

Files Generated for Altera IP Cores

The Quartus II software generates the following IP core output file structure:

Figure 6: IP Core Generated Files

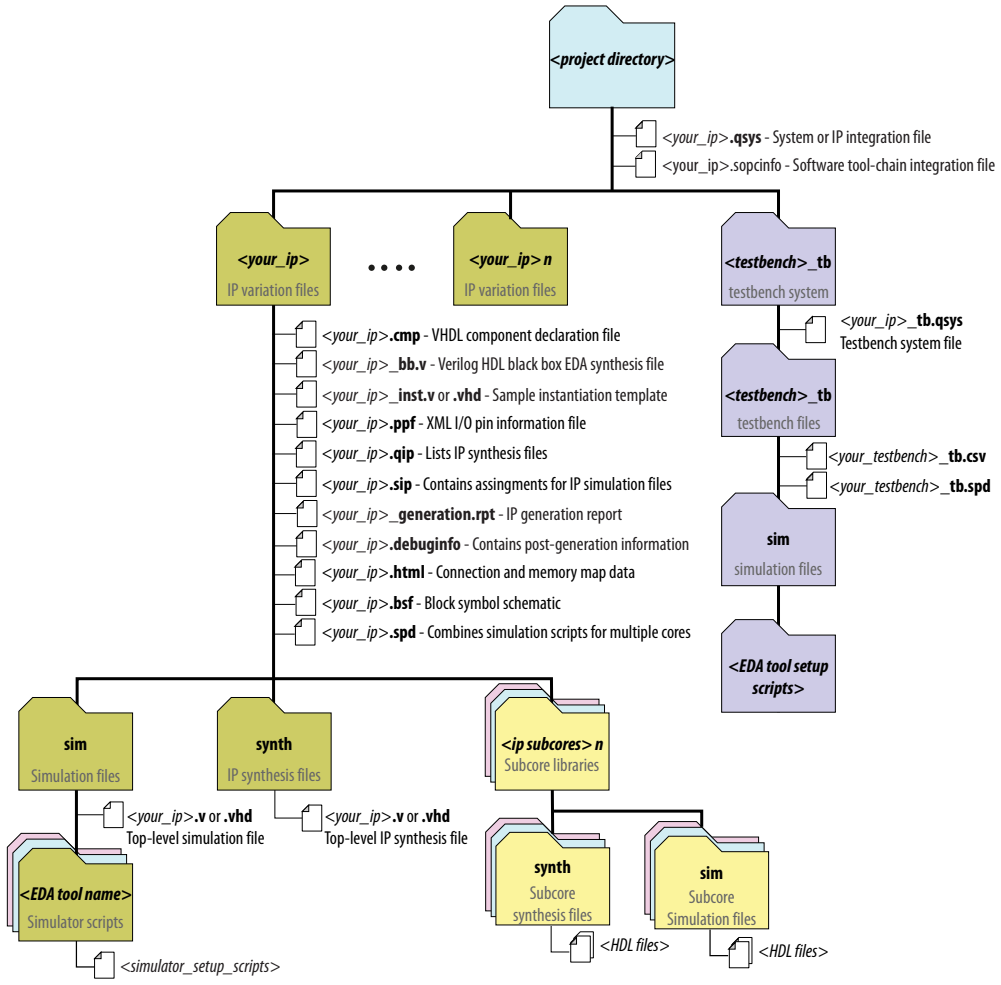


Table 3: IP Core Generated Files

File Name	Description
***<my_ip>*.qsys**	The Qsys system or top-level IP variation file. *<my_ip>* is the name that you give your IP variation.
***<system>*.sopcinfo**	Describes the connections and IP component parameterizations in your Qsys system. You can parse its contents to get requirements when you develop software drivers for IP components. Downstream tools such as the Nios II tool chain use this file. The **.sopcinfo** file and the **system.h** file generated for the Nios II tool chain include address map information for each slave relative to each master that accesses the slave. Different masters may have a different address map to access a particular slave component.

File Name	Description
<*my_ip*>.cmp	The VHDL Component Declaration (**.cmp**) file is a text file that contains local generic and port definitions that you can use in VHDL design files.
<*my_ip*>.html	A report that contains connection information, a memory map showing the address of each slave with respect to each master to which it is connected, and parameter assignments.
<*my_ip*>_generation.rpt	IP or Qsys generation log file. A summary of the messages during IP generation.
<*my_ip*>.debuginfo	Contains post-generation information. Used to pass System Console and Bus Analyzer Toolkit information about the Qsys interconnect. The Bus Analysis Toolkit uses this file to identify debug components in the Qsys interconnect.
<*my_ip*>.qip	Contains all the required information about the IP component to integrate and compile the IP component in the Quartus II software.
<*my_ip*>.csv	Contains information about the upgrade status of the IP component.
<*my_ip*>.bsf	A Block Symbol File (**.bsf**) representation of the IP variation for use in Quartus II Block Diagram Files (**.bdf**).
<*my_ip*>.spd	Required input file for `ip-make-simscript` to generate simulation scripts for supported simulators. The **.spd** file contains a list of files generated for simulation, along with information about memories that you can initialize.
<*my_ip*>.ppf	The Pin Planner File (**.ppf**) stores the port and node assignments for IP components created for use with the Pin Planner.
<*my_ip*>_bb.v	You can use the Verilog black-box (**_bb.v**) file as an empty module declaration for use as a black box.
<*my_ip*>.sip	Contains information required for NativeLink simulation of IP components. You must add the **.sip** file to your Quartus project.
<*my_ip*>_inst.v or **_inst.vhd**	HDL example instantiation template. You can copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*my_ip*>.regmap	If the IP contains register information, the .**regmap** file generates. The .**regmap** file describes the register map information of master and slave interfaces. This file complements the .**sopcinfo** file by providing more detailed register information about the system. This enables register display views and user customizable statistics in System Console.

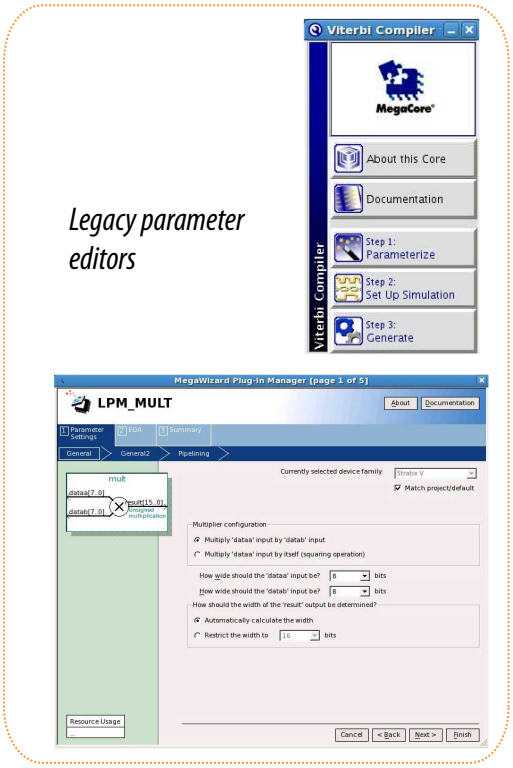
File Name	Description
<*my_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals connected to HPS within a Qsys system. During synthesis, the **.svd** files for slave interfaces visible to System Console masters are stored in the **.sof** file in the debug section. System Console reads this section, which Qsys can query for register map information. For system slaves, Qsys can access the registers by name.
<*my_ip*>.v or **<*my_ip*>.vhd**	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a ModelSim® script **msim_setup.tcl** to set up and run a simulation.
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.
/submodules	Contains HDL files for the IP core submodule.
<*child IP cores*>/	For each generated child IP core directory, Qsys generates **/synth** and **/sim** sub-directories.

Specifying IP Core Parameters and Options (Legacy Parameter Editors)

Some IP cores use a legacy version of the parameter editor for configuration and generation. Use the following steps to configure and generate an IP variation using a legacy parameter editor.

Note: The legacy parameter editor generates a different output file structure than the latest parameter editor. Refer to *Specifying IP Core Parameters and Options* for configuration of IP cores that use the latest parameter editor.

Figure 7: 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**.
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: 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.

Related Information

Specifying IP Core Parameters and Options on page 6

Files Generated for Altera IP Cores (Legacy Parameter Editors)

The Quartus II software generates one of the following output file structures for Altera IP cores that use a legacy parameter editor.

Figure 8: IP Core Generated Files (Legacy Parameter Editor)

Note: To manually add an IP variation to a Quartus II project, click **Project** > **Add/Remove Files in Project** and add only the IP variation **.qip** or **.qsys** file, but not both, to the project. Do not manually add the top-level HDL file to the project.

Modifying an IP Variation

You can easily modify the parameters of any Altera IP core variation in the parameter editor to match your design requirements. Use any of the following methods to modify an 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.

Scripting IP Core Generation

You can alternatively use command-line utilities to define and generate an IP core variation outside of the Quartus II GUI.Use `qsys-script` to run a Tcl file that parameterizes an IP variation you define in a script. Then, use `qsys-generate` to generate a **.qsys** file representing your parameterized IP variation.

The `qsys-generate` command is the same as when generating using the Qsys GUI. For command-line help listing all options for these executables, type <*executable name*> `--help`

To create a instance of a parameterizable Altera IP core at the command line, rather than using the GUI, follow these steps:

1. Run `qsys-script` to execute a Tcl script, similar to the example, that instantiates the IP and sets the IP parameters defined by the script:

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

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

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

Note: Creating an IP generation script is an advanced feature that requires access to special IP core parameters. For more information about creating an IP generation script, contact your Altera sales representative.

Table 5: qsys-generate Command-Line Options

Option	Usage	Description
`<1st arg file>`	Required	The name of the **.qsys** system file to generate.
`--synthesis=<VERILOG|VHDL>`	Optional	Creates synthesis HDL files that Qsys uses to compile the system in a Quartus II project. You must specify the preferred generation language for the top-level RTL file for the generated Qsys system.

Option	Usage	Description
`--block-symbol-file`	Optional	Creates a Block Symbol File (**.bsf**) for the Qsys system.
`--simulation=<VERILOG|VHDL>`	Optional	Creates a simulation model for the Qsys system. The simulation model contains generated HDL files for the simulator, and may include simulation-only features. You must specify the preferred simulation language.
`--testbench=<SIMPLE|STANDARD>`	Optional	Creates a testbench system that instantiates the original system, adding bus functional models (BFMs) to drive the top-level interfaces. When you generate the system, the BFMs interact with the system in the simulator.
`--testbench-simulation=<VERILOG|VHDL>`	Optional	After you create the testbench system, you can create a simulation model for the testbench system.
`--search-path=<value>`	Optional	If you omit this command, Qsys uses a standard default path. If you provide this command, Qsys searches a comma-separated list of paths. To include the standard path in your replacement, use "$", for example, "`/extra/dir,$`".
`--jvm-max-heap-size=<value>`	Optional	The maximum memory size that Qsys uses for allocations when running `qsys-generate`. You specify the value as `<size><unit>`, where `unit` is m (or M) for multiples of megabytes or g (or G) for multiples of gigabytes. The default value is 512m.
`--family=<value>`	Optional	Specifies the device family.
`--part=<value>`	Optional	Specifies the device part number. If set, this option overrides the `--family` option.

Option	Usage	Description
`--allow-mixed-language-simulation`	Optional	Enables a mixed language simulation model generation. If true, if a preferred simulation language is set, Qsys uses a `fileset` of the component for the simulation model generation. When false, which is the default, Qsys uses the language specified with `--file-set=<value>` for all components for simulation model generation. The current version of the ModelSim-Altera simulator supports mixed language simulation.

Upgrading IP Cores

IP core variants generated with a previous version of the Quartus II software may require upgrading before use in the current version of the Quartus II software. Click **Project** > **Upgrade IP Components** to identify and upgrade outdated IP core variants.

Icons in the **Upgrade IP Components** dialog box indicate when IP upgrade is required, optional, or unsupported for IP cores in your design. This dialog box may open automatically when you open a project containing upgradeable IP variations. You must upgrade IP cores that require upgrade before you can compile the IP variation in the current version of the Quartus II software.

The upgrade process preserves the original IP variation file in the project directory as <*my_variant*>_**BAK.qsys** for IP targeting Arria 10 and later devices, and as <*my_variant*>_**BAK.v**, **.sv**, or **.vhd** for legacy IP targeting 28nm devices and greater.

Note: Upgrading IP cores for Arria 10 and later devices may append a unique identifier to the original IP core entity name(s), without similarly modifying the IP instance name. There is no requirement to update these entity references in any supporting Quartus II file; such as the Quartus II Settings File (**.qsf**), Synopsys Design Constraints File (**.sdc**), or SignalTap File (**.stp**), if these files contain instance names. The Quartus II software reads only the instance name and ignores the entity name in paths that specify both names. Use only instance names in assignments.

Table 6: IP Core Upgrade Status

IP Core Status	Description
IP Upgraded 	Your IP variation uses the lastest version of the IP core.

IP Core Status	Description
IP Upgrade Optional 	Upgrade is optional for this IP variation in the current version of the Quartus II 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 Mismatch Warning 	Warning of non-critical IP core differences in migrating IP to another device family.
IP Upgrade Required 	You must upgrade the IP variation before compiling in the current version of the Quartus II software. Refer to the Description for details about IP core version differences.
IP Upgrade Unspported 	Upgrade of the IP variation is not supported in the current version of the Quartus II software due to incompatibility with the current version of the Quartus II software. You are prompted 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 	Altera 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 Quartus II software.
Encrypted IP Core 	The IP variation is encrypted.

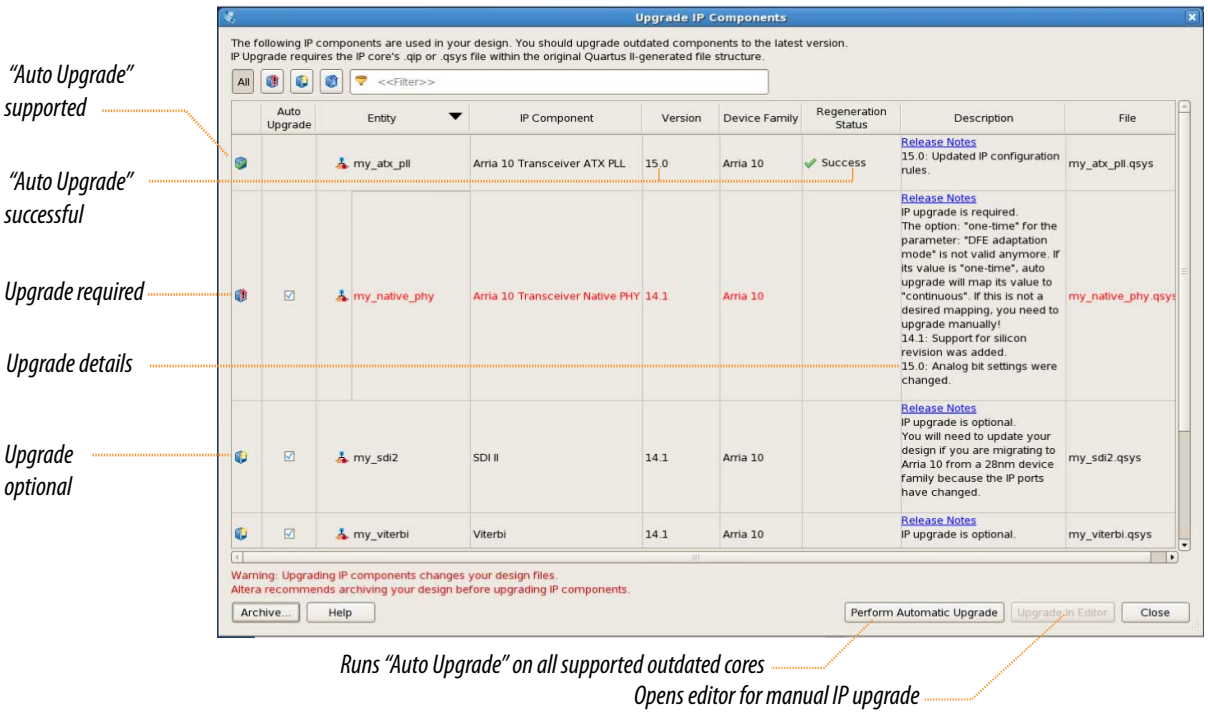
Follow these steps to upgrade IP cores:

1. In the latest version of the Quartus II software, open the Quartus II project containing an outdated IP core variation. The **Upgrade IP Components** dialog automatically displays the status of IP cores in your project, along with instructions for upgrading each core. Click **Project** > **Upgrade IP Components** to access this dialog box manually.
2. To upgrade one or more IP cores that support automatic upgrade, ensure that the **Auto Upgrade** option is turned on for the IP core(s), and then click **Perform Automatic Upgrade**. The **Status** and

Version columns update when upgrade is complete. Example designs provided with any Altera IP core regenerate automatically whenever you upgrade an IP core.

3. To manually upgrade an individual IP core, select the IP core and then 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 9: Upgrading IP Cores



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

Related Information

Altera IP Release Notes

Upgrading IP at Command-Line

You can upgrade an IP core at the command-line, as long as the IP core supports auto upgrade. IP cores that do not support automatic upgrade do not support command-line upgrade.

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

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

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

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

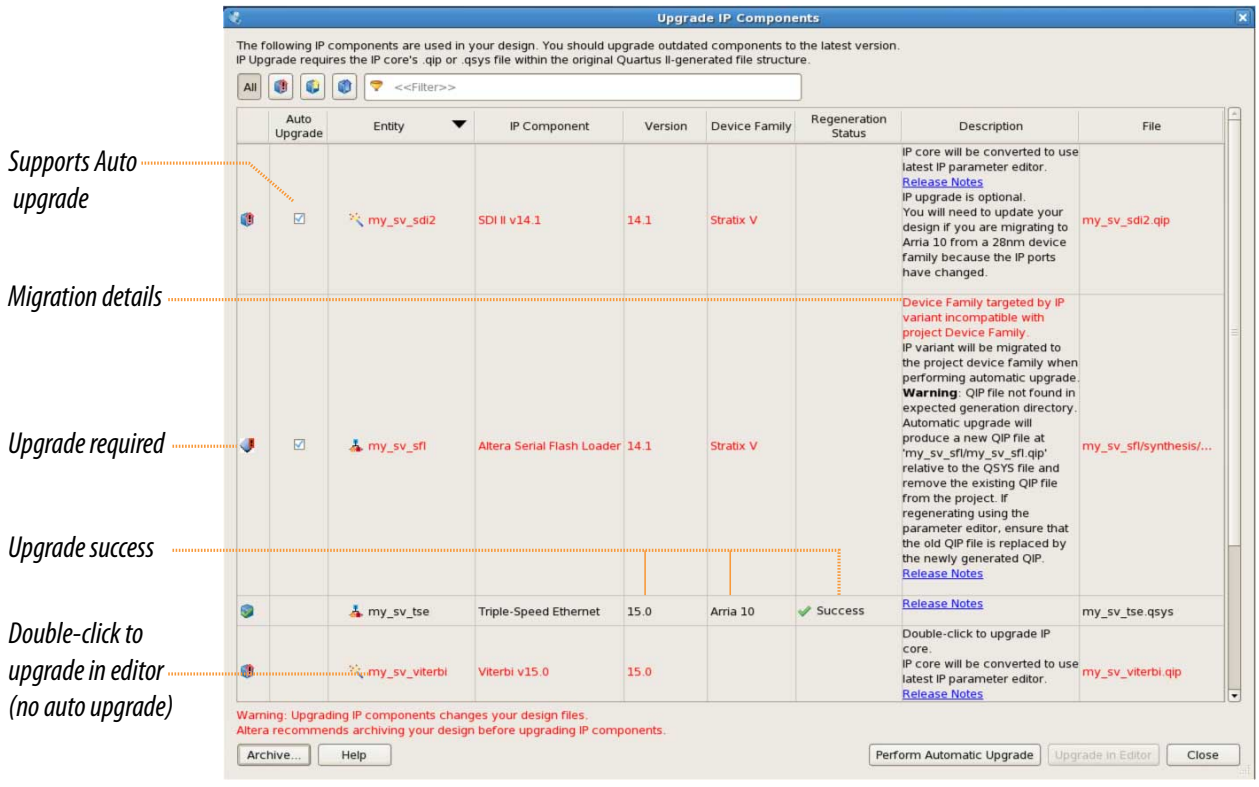
```
quartus_sh –ip_upgrade –variation_files “<my_ip1>.<qsys,.v, .vhd>>;
<my_ip_filepath/my_ip2>.<hdl>”  <qii_project>

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

Migrating IP Cores to a Different Device

IP migration allows you to target the latest device families with IP originally generated for a different device. Some Altera IP cores migrate automatically, some IP cores require manual IP regeneration, and some do not support device migration and must be replaced in your design.
The text and icons in the **Upgrade IP Components** dialog box identifies the migration support for each IP core in the design.

Note: Migration of some IP cores requires installed support for the original and migration device families. For example, migration from a Stratix V device to an Arria 10 device requires installation of Stratix V and Arria 10 device families with the Quartus II software.

Figure 10: Upgrading IP Cores



1. Click **File** > **Open Project** and open the Quartus II project containing IP for migration to another device in the original version of the Quartus II software.
2. To specify a different target device for migration, click **Assignments** > **Device** and select the target device family.
3. To display IP cores requiring migration, click **Project** > **Upgrade IP Components**. The **Description** field prompts you to run auto update or double-click IP cores for migration.
4. To migrate one or more IP cores that support automatic upgrade, ensure that the **Auto Upgrade** option is turned on for the IP core(s), and then click **Perform Automatic Upgrade**. The **Status** and **Version** columns update when upgrade is complete.
5. To migrate an IP core that does not support automatic upgrade, double-click the IP core name, and then click **OK**. The parameter editor appears.
   a. If the parameter editor specifies a **Currently selected device family**, turn off **Match project/ default**, and then select the new target device family.
   b. Click **Finish** to migrate the IP variation using best-effort mapping to new parameters and settings. A new parameter editor opens displaying best-effort mapped parameters.
   c. Click **Generate HDL**, and then confirm the **Synthesis** and **Simulation** file options. Verilog HDL is the defauilt output file format specified. If your original IP core was generated for VHDL, select **VHDL** to retain the original output format.
6. To regenerate the new IP variation for the new target device, click **Generate**. When generation is complete, click **Close**.
7. Click **Finish** to complete migration of the IP core. Click **OK** if you are prompted to overwrite IP core files. The **Device Family** column displays the new target device name when migration is complete. The migration process replaces <*my_ip*>**.qip** with the <*my_ip*>**.qsys** top-level IP file in your project.

   Note: If migration does not replace <*my_ip*>**.qip** with <*my_ip*>**.qsys**, click **Project > Add/Remove Files in Project** to replace the file in your project.
8. Review the latest parameters in the parameter editor or generated HDL for correctness. IP migration may change ports, parameters, or functionality of the IP core. During migration, the IP core'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 your upgraded IP core is represented by a symbol in a supporting Block Design File schematic, replace the symbol with the newly generated <*my_ip*>**.bsf** after migration.

   Note: The migration process may change the IP variation interface, parameters, and functionality. This may require you to change your design or to re-parameterize your variant after the **Upgrade IP Components** dialog box indicates that migration is complete. The **Description** field identifies IP cores that require design or parameter changes.

Related Information

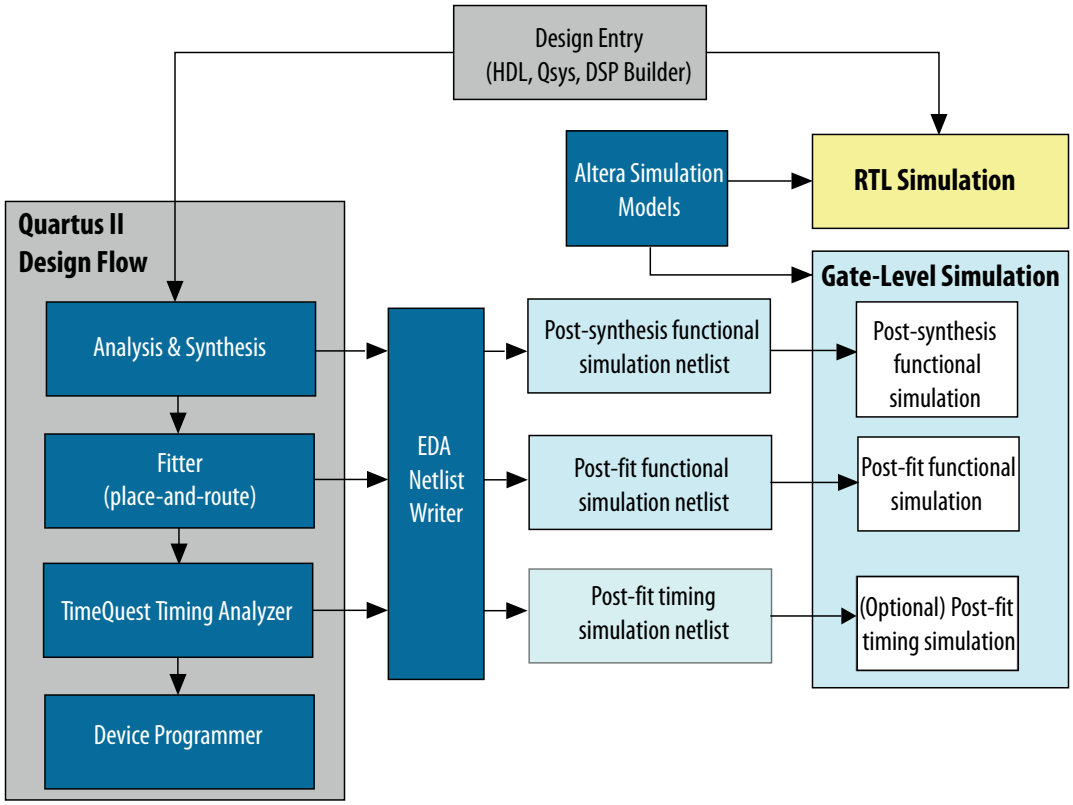
Altera IP Release Notes

Simulating Altera IP Cores in other EDA Tools

The Quartus II software supports RTL and gate-level design simulation of Altera IP cores in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

You can use the functional simulation model and the testbench or example design generated with your IP core for simulation. The functional simulation model and testbench files are generated in a project subdirectory. This directory may also include scripts to compile and run the testbench. For a complete list of models or libraries required to simulate your IP core, refer to the scripts generated with the testbench. You can use the Quartus II NativeLink feature to automatically generate simulation files and scripts. NativeLink launches your preferred simulator from within the Quartus II software.

Figure 11: Simulation in Quartus II Design Flow



Note: Post-fit timing simulation is supported only for Stratix IV and Cyclone IV devices in the current version of the Quartus II software. Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model. Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

Related Information

Simulating Altera Designs

Simulation Flows

The Quartus II software supports various method for integrating your supported simulator into the design flow.

Table 7: Simulation Flows

Simulation Flow	Description
NativeLink flow	The NativeLink automated flow supports a variety of design flows. Do not use NativeLink if you require direct control over every aspect of simulation. • Use NativeLink to generate simulation scripts to compile your design and simulation libraries, and to automatically launch your simulator. • Specify your own compilation, elaboration, and simulation scripts for testbench and simulation model files that have not been analyzed by the Quartus II software. • Use NativeLink to supplement your scripts by automatically compiling design files, IP simulation model files, and Altera simulation library models.
Custom flows	Custom flows support manual control of all aspects of simulation, including the following: • Manually compile and simulate testbench, design, IP, and simulation model libraries, or write scripts to automate compilation and simulation in your simulator. • Use the Simulation Library Compiler to compile simulation libraries for all Altera devices and supported third-party simulators and languages. Use the custom flow if you require any of the following: • Custom compilation commands for design, IP, or simulation library model files (for example, macros, debugging or optimization options, or other simulator-specific options). • Multi-pass simulation flows. • Flows that use dynamically generated simulation scripts.
Specialized flows	Altera supports specialized flows for various design variations, including the following: • For simulation of Altera example designs, refer to the documentation for the example design or to the IP core user guide. • For simulation of Qsys designs, refer to *Creating a System with Qsys*. • For simulation of designs that include the Nios II embedded processor, refer to *Simulating a Nios II Embedded Processor*.

Related Information

- **IP User Guide Documentation**
- **Creating a System with Qsys**
- **Simulating a Nios II Embedded Processor**

Simulator Support

The Quartus II software supports specific EDA simulator versions for RTL and gate-level simulation.

Table 8: Supported Simulators

Vendor	Simulator	Version	Platform
Aldec	Active-HDL	10.1	Windows
Aldec	Riviera-PRO	2014.10	Windows, Linux
Cadence	Incisive Enterprise	14.1	Linux
Mentor Graphics	ModelSim-Altera (provided)	10.3d	Windows, Linux
Mentor Graphics	ModelSim PE	10.3d	Windows
Mentor Graphics	ModelSim SE	10.3d	Windows, Linux
Mentor Graphics	QuestaSim	10.3d	Windows, Linux
Synopsys	VCS/VCS MX	2014,03-SP1	Linux

Simulation Levels

The Quartus II software supports RTL and gate-level simulation of IP cores in supported EDA simulators.

Table 9: Supported Simulation Levels

Simulation Level	Description	Simulation Input
RTL	Cycle-accurate simulation using Verilog HDL, SystemVerilog, and VHDL design source code with simulation models provided by Altera and other IP providers.	• Design source/testbench • Altera simulation libraries • Altera IP plain text or IEEE encrypted RTL models • IP simulation models • Altera IP functional simulation models • Altera IP bus functional models • Qsys-generated models • Verification IP
Gate-level functional	Simulation using a post-synthesis or post-fit functional netlist testing the post-synthesis functional netlist, or post-fit functional netlist.	• Testbench • Altera simulation libraries • Post-synthesis or post-fit functional netlist • Altera IP bus functional models
Gate-level timing	Simulation using a post-fit timing netlist, testing functional and timing performance. Supported only for the Stratix IV and Cyclone IV device families.	• Testbench • Altera simulation libraries • Post-fit timing netlist • Post-fit Standard Delay Output File (**.sdo**)

Note: Gate-level timing simulation of an entire design can be slow and should be avoided. Gate-level timing simulation is supported only for the Stratix IV and Cyclone IV device families. Use TimeQuest static timing analysis rather than gate-level timing simulation.

HDL Support

The Quartus II software provides the following HDL support for EDA simulators.

Table 10: HDL Support

Language	Description
VHDL	• For VHDL RTL simulation, compile design files directly in your simulator. To use NativeLink automation, analyze and elaborate your design in the Quartus II software, and then use the NativeLink simulator scripts to compile the design files in your simulator. You must also compile simulation models from the Altera simulation libraries and simulation models for the IP cores in your design. Use the Simulation Library Compiler or NativeLink to compile simulation models. • For gate-level simulation, the EDA Netlist Writer generates a synthesized design netlist VHDL Output File (**.vho**). Compile the **.vho** in your simulator. You may also need to compile models from the Altera simulation libraries. • IEEE 1364-2005 encrypted Verilog HDL simulation models are encrypted separately for each Altera-supported simulation vendor. If you want to simulate the model in a VHDL design, you need either a simulator that is capable of VHDL/Verilog HDL co-simulation, or any Mentor Graphics single language VHDL simulator.
Verilog HDL SystemVerilog	• For RTL simulation in Verilog HDL or SystemVerilog, compile your design files in your simulator. To use NativeLink automation, analyze and elaborate your design in the Quartus II software, and then use the NativeLink simulator scripts to compile your design files in your simulator. You must also compile simulation models from the Altera simulation libraries and simulation models for the IP cores in your design. Use the Simulation Library Compiler or NativeLink to compile simulation models. • For gate-level simulation, the EDA Netlist Writer generates a synthesized design netlist Verilog Output File (**.vo**). Compile the **.vo** in your simulator.
Mixed HDL	• If your design is a mix of VHDL, Verilog HDL, and SystemVerilog files, you must use a mixed language simulator. Choose the most convenient supported language for generation of Altera IP cores in your design. • Altera provides the entry-level ModelSim-Altera software, along with precompiled Altera simulation libraries, to simplify simulation of Altera designs. The latest version of the ModelSim-Altera software supports native, mixed-language (VHDL/Verilog HDL/SystemVerilog) co-simulation of plain text HDL. If you have a VHDL-only simulator and need to simulate Verilog HDL modules and IP cores, you can either acquire a mixed-language simulator license from the simulator vendor, or use the ModelSim-Altera software.

Language	Description
Schematic	You must convert schematics to HDL format before simulation. You can use the converted VHDL or Verilog HDL files for RTL simulation.

Compiling Simulation Models

The Quartus II software includes simulation models for all Altera IP cores. These models include IP functional simulation models, and device family-specific models in the **<*Quartus II installation path*>/eda/sim_lib** directory. These models include IEEE encrypted Verilog HDL models for both Verilog HDL and VHDL simulation.

Before running simulation, you must compile the appropriate simulation models from the Altera simulation libraries using any of the following methods:

- Use the NativeLink feature to automatically compile your design, Altera IP, simulation model libraries, and testbench.
- Run the Simulation Library Compiler to compile all RTL and gate-level simulation model libraries for your device, simulator, and design language.
- Compile Altera simulation models manually with your simulator.

After you compile the simulation model libraries, you can reuse these libraries in subsequent simulations.

Note: The specified timescale precision must be within 1ps when using Altera simulation models.

Related Information

Altera Simulation Models

Generating IP Simulation Files for RTL Simulation

The Quartus II software supports both Verilog HDL and VHDL simulation of encrypted and unencrypted Altera IP cores. If your design includes Altera IP cores, you must compile any corresponding IP simulation models in your simulator with the rest of your design and testbench. The Quartus II software generates and copies the simulation models for IP cores to your project directory.

You can use the following files to simulate your Altera IP variation.

Table 11: Altera IP Simulation Files

File Type	Description	File Name
Simulator setup script	Simulator-specific script to compile, elaborate, and simulate Altera IP models and simulation model library files. Copy the commands into your simulation script, or edit these files to compile, elaborate, and simulate your design and testbench.	Cadence • **cds.lib** • **ncsim_setup.sh** • **hdl.var** Mentor Graphics • **msim_setup.tcl** Synopsys • **synopsys_sim.setup** • **vcs_setup.sh** • **vcsmx_setup.sh** Aldec • **rivierapro_setup.tcl**
Quartus II Simulation IP File (**.sip**)	Contains IP core simulation library mapping information. The **.sip** files enable NativeLink simulation and the Quartus II Archiver for IP cores.	**<*design name*>.sip**
IP functional simulation models	IP functional simulation models are cycle-accurate VHDL or Verilog HDL models generated by the Quartus II software for some Altera IP cores. IP functional simulation models support fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.	**<*my_ip*>.vho** **<*my_ip*>.vo**
IEEE encrypted models	Arria V, Cyclone V, Stratix V, and newer simulation model libraries and IP simulation models are provided in Verilog HDL and IEEE encrypted Verilog HDL. VHDL simulation of these models is supported using your simulator's co-simulation capabilities. IEEE encrypted Verilog HDL models are significantly faster than IP functional simulation models.	**<*my_ip*>.v**

Generating IP Functional Simulation Models for RTL Simulation

Altera provides IP functional simulation models for some Altera IP cores. To generate IP functional simulation models, follow these steps:

- Turn on the **Generate Simulation Model** option when parameterizing the IP core.
- When you simulate your design, compile only the **.vo** or **.vho** for these IP cores in your simulator. In this case you should not compile the corresponding HDL file. The encrypted HDL file supports synthesis by only the Quartus II software.

Note: Altera 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.

Note: Many recently released Altera IP cores support RTL simulation using IEEE Verilog HDL encryption. IEEE encrypted models are significantly faster than IP functional simulation models. You can simulate the models in both Verilog HDL and VHDL designs.

Related Information

AN 343: OpenCore Evaluation of AMPP Megafunctions

Generating Simulation Scripts

You can automatically generate simulation scripts to set up supported simulators. These scripts compile the required device libraries and system design files in the correct order, and then elaborate or load the top-level design for simulation. You can also use scripts to modify the top-level simulation environment, independent of IP simulation files that are replaced during regeneration. You can modify the scripts to set up supported simulators.

Use the NativeLink feature to generate simulation scripts to automate simulation steps. You can reuse these generated files and simulation scripts in a custom simulation flow. NativeLink optionally generates scripts for your simulator in the project subdirectory.

1. Click **Assignments > Settings**.
2. Under **EDA Tool Settings,** click **Simulation**.
3. Select the **Tool name** of your simulator.
4. Click **More NativeLink Settings**.
5. Turn on **Generate third-party EDA tool command scripts without running the EDA tool**.

Table 12: NativeLink Generated Scripts for RTL Simulation

Simulator(s)	Simulation File	Use
Mentor Graphics ModelSim QuestaSim	**/simulation/modelsim/<*my_ip*>.do**	Source directly with your simulator.
Aldec Riviera Pro	**/simulation/modelsim/<*my_ip*>.do**	Source directly with your simulator.
Synopsys VCS	**/simulation/modelsim/<*revision name*>_<*rtl or gate*>.vcs**	Add your testbench file name to this options file to pass the file to VCS using the `-file` option. If you specify a testbench file to NativeLink, NativeLink generates an **.sh** script that runs VCS.
Synopsys VCS MX	**/simulation/scsim/<*revision name*>_vcsmx_<*rtl or gate*>_<*verilog or vhdl*>.tcl**	Run this script at the command line using the command: `quartus_sh -t <script>` Any testbench you specify with NativeLink is included in this script.
Cadence Incisive (NC SIM)	**/simulation/ncsim/<*revision name*>_ncsim_<*rtl or gate*>_<*verilog or vhdl*>.tcl**	Run this script at the command line using the command: `quartus_sh -t <script>`. Any testbench you specify with NativeLink is included in this script.

You can use the following script variables:

- `TOP_LEVEL_NAME`—The top-level entity of your simulation is often a testbench that instantiates your design, and then your design instantiates IP cores and/or Qsys systems. Set the value of `TOP_LEVEL_NAME` to the top-level entity.
- `QSYS_SIMDIR`—Specifies the top-level directory containing the simulation files.
- Other variables control the compilation, elaboration, and simulation process.

Generating Custom Simulation Scripts with ip-make-simscript

Use the `ip-make-simscript` utility to generate simulation command scripts for multiple IP cores or Qsys systems. Specify all Simulation Package Descriptor files (**.spd**), each of which lists the required simulation files for the corresponding IP core or Qsys system. The IP parameter editor generates the **.spd** files.

`ip-make-simscript` compiles IP simulation models into various simulation libraries. Use the `compile-to-work` option to compile all simulation files into a single work library. Use this option only if you require a simplified library structure.

When you specify multiple **.spd** files, the `ip-make-simscript` utility generates a single simulation script containing all required simulation information. The default value of `TOP_LEVEL_NAME` is the `TOP_LEVEL_NAME` defined in the IP core or Qsys **.spd** file.

Set appropriate variables in the script, or edit the variable assignment directly in the script. If the simulation script is a Tcl file that is sourced in the simulator, set the variables before sourcing the script. If the simulation script is a shell script, pass in the variables as command-line arguments to the shell script.

- Type `ip-make-simscript` at the command prompt to run.
- Type `ip-make-simscript --help` for help on command options and syntax.

Table 13: ip-make-simscript Examples

Option	Description	Status
`--spd=`*<file>*	Describes the list of compiled files and memory model hierarchy. If your design includes multiple IP cores or Qsys systems that include **.spd** files, use this option for each file. You can specify multiple **.spd** files as a comma-separated list. For example: `ip-make-simscript --spd=,ip1.spd, ip2.spd,`	Required
`--output-directory=`*<directory>*	Specifies the location of output files. If unspecified, the default setting is the directory from which `ip-make-simscript` is run.	Optional
`--compile-to-work`	Compiles all design files to the default work library. Use this option only if you encounter problems managing your simulation with multiple libraries.	Optional

Option	Description	Status
`--use-relative-paths`	Uses relative paths whenever possible.	Optional

Synthesizing Altera IP Cores in Other EDA Tools

You can use supported EDA tools to synthesize a design that includes Altera IP cores. When you generate the IP core synthesis files for use with third-party EDA synthesis tools, you can optionally 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 Quartus II 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 Quartus II project.

Related Information

Quartus II Integrated Synthesis

Instantiating IP Cores in HDL

You can instantiate an IP core directly in your HDL code by calling the IP core name and declaring its parameters, in the same manner as any other module, component, or subdesign. When instantiating an IP core in VHDL, you must include the associated libraries.

Accessing HDL Code Templates

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

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 then click **Insert**.

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
```

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;
```

Document Revision History

This document has the following revision history.

Date	Version	Changes
2015.05.04	15.0.0	• 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	14.1.0	Added information about new **Assignments** > **Settings** > **IP Settings** that control frequency of synthesis file regeneration and automatic addtion of IP files to the project.
2014.08.18	14.0a10.0	• Added information about specifying parameters for IP cores targeting Arria 10 devices. • Added information about the latest IP output for 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.
June 2014	14.0.0	• 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 2013	13.0 .1	• Reorganization of content into topics. • First tracking of changes in Document Revision History.