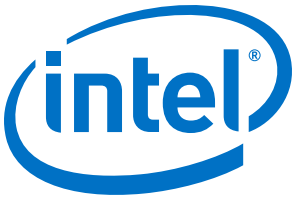


Intel® Quartus® Prime Pro Edition User Guide

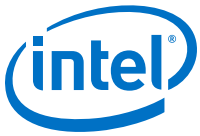
Third-party Simulation

Updated for Intel® Quartus® Prime Design Suite: **20.1**

Subscribe
Send Feedback

UG-20137 | 2020.11.02
Latest document on the web: **PDF** | **HTML**

Contents



1. Simulating Intel FPGA Designs

This document describes simulating designs that target Intel FPGA devices. Simulation verifies design behavior before device programming. The Intel® Quartus® Prime software supports RTL- and gate-level design simulation in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

1.1. Simulator Support

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

Table 1. Supported Simulators

Vendor	Simulator	Version	Platform
Aldec	Active-HDL*	10.5	Windows* 32-bit only
Aldec	Riviera-PRO*	2018.10	Windows, Linux, 64-bit only
Cadence	Incisive Enterprise*	15.20	Linux, 64-bit only
Cadence	Xcelium* Parallel Simulator	18.03	Linux 64-bit only
Mentor Graphics*	ModelSim* - Intel FPGA Edition	10.6d	Windows, Linux, 32-bit only
Mentor Graphics	ModelSim PE	10.6d	Windows 32-bit only
Mentor Graphics	ModelSim SE	10.6d	Windows, Linux, 64-bit only
Mentor Graphics	QuestaSim*	10.6d	Windows, Linux, 64-bit only
Synopsys*	VCS* VCS MX	2017.12-SP2-4	Linux 64-bit only

Simulator Support for Mentor Verification IP Bus Functional Models (BFMs)

The following simulators support simulation of the Mentor Verification IP bus functional models (BFMs) that you use in simulation of hard processor system (HPS) designs:

- Mentor Graphics ModelSim (including ModelSim - Intel FPGA Edition), and QuestaSim 10.1d.
- Synopsys VCS and VCS MX 2012.09.
- Cadence Incisive Enterprise* Simulator (IES) 12.10.013.

1.2. Simulation Levels

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

Table 2. 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 Intel and other IP providers.	• Design source/testbench • Intel simulation libraries • Intel FPGA IP plain text or IEEE encrypted RTL models • IP simulation models • Intel FPGA IP functional simulation models • Intel FPGA IP bus functional 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 • Intel simulation libraries • Post-synthesis or post-fit functional netlist • Intel FPGA IP bus functional models

1.3. HDL Support

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

Table 3. HDL Support

Language	Description
VHDL	• For VHDL RTL simulation, compile design files directly in your simulator. You must also compile simulation models from the Intel FPGA simulation libraries and simulation models for the IP cores in your design. Use the Simulation Library Compiler 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 Intel FPGA simulation libraries. • IEEE 1364-2005 encrypted Verilog HDL simulation models are encrypted separately for each simulation vendor that the Quartus Prime software supports. To simulate the model in a VHDL design, you must have a simulator that is capable of VHDL/Verilog HDL co-simulation.
Verilog HDL -SystemVerilog	• For RTL simulation in Verilog HDL or SystemVerilog, compile your design files in your simulator. You must also compile simulation models from the Intel FPGA simulation libraries and simulation models for the IP cores in your design. Use the Simulation Library Compiler 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 Intel FPGA IP cores in your design. • Intel FPGA provides the entry-level ModelSim - Intel FPGA Edition software, along with precompiled Intel FPGA simulation libraries, to simplify simulation of Intel FPGA designs. Starting in version 15.0, the ModelSim - Intel FPGA Edition 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 - Intel FPGA Edition software.
Schematic	You must convert schematics to HDL format before simulation. You can use the converted VHDL or Verilog HDL files for RTL simulation.

1.4. Simulation Flows

The Intel Quartus Prime software supports various simulation flows.

Table 4. 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 generates a combined simulator setup script for all IP cores, for each supported simulator.
Specialized Simulation Flows	Supports specialized simulation flows for specific design variations, including the following: • For simulation of example designs, refer to the documentation for the example design or to the IP core user guide. • For simulation of Platform Designer designs, refer to *Creating a System with Platform Designer* or *Creating a System with Platform Designer*. *Note:* The simulation setup script generated by Platform Designer requires Tcl version of 8.5 or higher. • For simulation of designs that include the Nios® II embedded processor, refer to *Simulating a Nios II Embedded Processor*.

Related Information

- AN 351: Simulating Nios II Embedded Processors Designs
- Creating a System With Platform Designer

1.5. Preparing for Simulation

Preparing for RTL or gate-level simulation involves compiling the RTL or gate-level representation of your design and testbench. You must also compile IP simulation models, models from the Intel FPGA simulation libraries, and any other model libraries required for your design.

1.5.1. Compiling Simulation Models

The Intel Quartus Prime software includes simulation models for all Intel FPGA IP cores. These models include IP functional simulation models, and device family-specific models in the `<Intel Quartus Prime 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 Intel Quartus Prime simulation libraries using any of the following methods:

- To automatically compile all required simulation model libraries for your design in your supported simulator, click **Tools ➤ Launch Simulation Library Compiler**. Specify options for your simulation tool, language, target device family, and output location, and then click **OK**.
- Compile Intel Quartus Prime simulation models manually with your simulator.

Use the compiled simulation model libraries to simulate your design. Refer to your EDA simulator's documentation for information about running simulation.

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

Related Information

Intel Quartus Prime Simulation Models
In Intel Quartus Prime Pro Edition Help

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

For this IP, the **Generate Example Design** button in IP parameter editor produces RTL, C, and MATLAB files for simulation. The use of these files in each environment is described in the following sub-sections.

1.6.1. Generating IP Simulation Files

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

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

Table 5. Intel FPGA IP Simulation Files

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

File Type	Description	File Name
		<my_dir>/synopsys/vcsmx/vcsmx_setup.sh

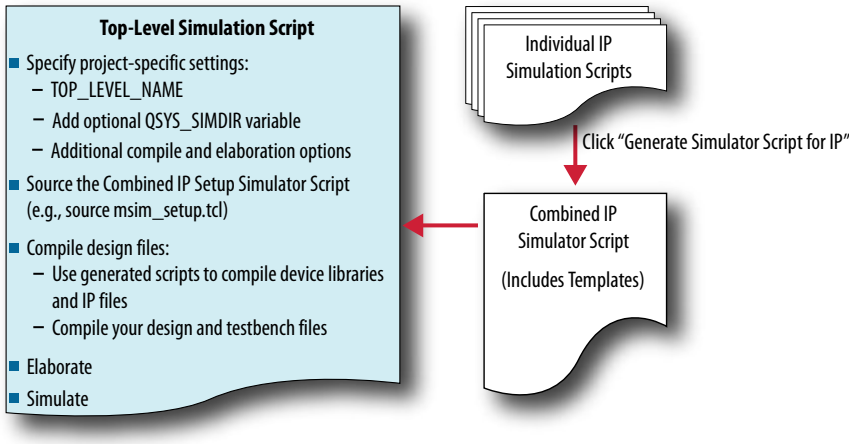
Note: Intel FPGA IP cores support a variety of cycle-accurate simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog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

1.6.2. Scripting IP Simulation

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

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

Figure 1. Incorporating Generated Simulator Setup Scripts into a Top-Level Simulation Script



1. Click **Project ➤ Upgrade IP Components ➤ 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.

1.6.2.1. Generating a Combined Simulator Setup Script (Intel Quartus Prime Pro Edition)

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

Note: This feature is available in the Intel Quartus Prime Pro Edition software for all devices. This feature is available in the Intel Quartus Prime Standard Edition software for only Intel Arria® 10 devices.

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

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

1. Generate, regenerate, or upgrade one or more IP core. Refer to *Generating IP Cores* or *Upgrading IP Cores*.
2. 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.
3. To incorporate the generated simulator setup script into your top-level simulation script, refer to the template section in the generated simulator setup script as a guide to creating a top-level script:
   a. Copy the specified template sections from the simulator-specific generated scripts and paste them into a new top-level file.
   b. Remove the comments at the beginning of each line from the copied template sections.
   c. Specify the customizations you require to match your design simulation requirements, for example:
      - Specify the `TOP_LEVEL_NAME` variable to the design's simulation top-level file. The top-level entity of your simulation is often a testbench that instantiates your design. Then, your design instantiates IP cores or Platform Designer systems. Set the value of `TOP_LEVEL_NAME` to the top-level entity.
      - If necessary, set the `QSYS_SIMDIR` variable to point to the location of the generated IP simulation files.
      - Compile the top-level HDL file (for example, a test program) and all other files in the design.
      - 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.
4. Re-run **Tools ➤ Generate Simulator Setup Script for IP** (or `ip-setup-simulation`) after regeneration of an IP variation.

Table 6. Simulation Script Utilities

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.

`ip-make-simscript` generates a combined simulation script for all IP cores that you specify on the command line. Specify one or more `.spd` files and an output directory in the command. Running the script compiles IP simulation models into various simulation libraries.	`ip-make-simscript`
	` --spd=<ipA.spd,ipB.spd>`
	` --output-directory=<directory>`

`ip-make-simscript` generates a combined simulation script for all IP cores and subsystems that you specify on the command line.	`ip-make-simscript`
	`--system-files=<ipA.ip, ipB.ip>`
	`--output-directory=<directory>`

1.6.2.2. Incorporating Simulator Setup Scripts from the Generated Template

You can incorporate generated IP core simulation scripts into a top-level simulation script that controls simulation of your entire design. After running `ip-setup-simulation` use the following information to copy the template sections and modify them for use in a new top-level script file.

1.6.2.2.1. 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.tcl`.

```
# # 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
```

1.6.2.2.2. 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`.

1.6.2.2.3. 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`.

1.6.2.2.4. Sourcing Mentor Graphics ModelSim Simulator Setup Scripts

Follow these steps to incorporate the generated ModelSim 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.tcl`.

```
# # 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 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
```

1.6.2.2.5. 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`.

1.6.2.2.6. 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 the 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`.

1.7. Running a Simulation (Custom Flow)

Use a custom simulation flow to support any of the following more complex simulation scenarios:

- Custom compilation, elaboration, or run commands for your design, IP, or simulation library model files (for example, macros, debugging/optimization options, simulator-specific elaboration or run-time options)
- Multi-pass simulation flows
- Flows that use dynamically generated simulation scripts

Use these to compile libraries and generate simulation scripts for custom simulation flows:

- Simulation Library Compiler—compile Intel FPGA simulation libraries for your device, HDL, and simulator. Generate scripts to compile simulation libraries as part of your custom simulation flow. This tool does not compile your design, IP, or testbench files.
- IP and Platform Designer simulation scripts—use the scripts generated for Intel FPGA IP cores and Platform Designer systems as templates to create simulation scripts. If your design includes multiple IP cores or Platform Designer systems, you can combine the simulation scripts into a single script, manually or by using the `ip-make-simscript` utility.

Use the following steps in a custom simulation flow:

1. Compile the design and testbench files in your simulator.
2. Run the simulation in your simulator.

1.8. The EDA Netlist Writer and Gate-level Netlists

The Quartus EDA Netlist Writer (`quartus_eda` on the command line) allows you to write gate-level (technology mapped) netlists for simulation and other applications. Intel does not recommend simulation of gate-level netlists as the main method of testing and validation. Gate level simulation is slower than RTL simulation and it is harder to debug because of the remapping of nodes and names.

The EDA Netlist Writer supports:

- Single-bit signal types
- One-dimensional arrays
- Two-dimensional arrays

The EDA Netlist Writer does not support complex data types, such as enums, stucts, unions, or interfaces, at the external boundary of the design or the partition that it produces.

The netlist writer can produce netlists after synthesis and after the completion of the fitter, the synthesized, and final snapshot respectively. It supports output of Verilog (`.vo`) and VHDL (`.vho`) netlists. It can also produce Verilog Quartus Map (`.vqm`) netlists for resynthesis.

Start the netlist writer from the command line using `quartus_eda`, followed by the set of options to specify the type of netlist to produce.

--simulation

The simulation flag specifies that `quartus_eda` creates a Verilog (`.vo`) or VHDL (`.vho`) gate-level netlist, for simulation by one of the supported simulators. This option requires you to also specify the target tool and format for the simulation.

--tool=<*modelsim|modelsim_oem|vcs|vcs_mx|ncsim|xcelium|rivierapro|activehdl|verilogxl*>

This option specifies that `quartus_eda` writes out a netlist for the specified third-party EDA tool. You can choose the third-party EDA tool from one of the three categories of available tools: simulation, timing analysis, or board level design and analysis.

This option overrides the settings specified in the Intel Quartus Prime Settings File (`.qsf`). Specify both the tool name and format to generate a netlist.

--format=<*vhdl|verilog|ibis (when using with –board_signal_integrity flag)*>

The `--format` option specifies whether the simulation option produces a Verilog or VHDL gate-level netlist.

--resynthesis

The `--resynthesis` flag specifies that `quartus_eda` creates a Verilog Quartus Map (`.vqm`) netlist. The software can resynthesize the netlist as an RTL input, from the gate-level netlist. Only use this option with partitions containing core logic only, not periphery. The sub-option is a flag only and takes no arguments.

--power

The `--power` flag specifies that `quartus_eda` generates a standard delay format output (`.sdo`) file. You can use this file in power analysis,but it is not a fully accurate timing simulation. Currently Intel only supports this option with Verilog HDL simulations in the ModelSim simulator.

--partition=<*partition name*>

The `--partition` selects an individual partition as the netlist output. For no partition argument, the software writes the entire design out to a single file. The partition argument takes a name of a partition in the design.

You can use the `--partition` option with the `--simulation` (`.vo`, `.vho`) and `--resynthesis` (`.vqm`) output.

--exclude_sub_partitions

The `--exclude_sub_partitions` flag limits the output to the netlist of this partition only. This flag is only valid when you use the `--partition` option, this flag outputs the netlist belonging to the partition you specify. The software instantiates subpartitions as module instances in the netlist. The sub-option is a flag only and takes no arguments.

When you specify the `--exclude_sub_partitions` flag, the software only writes out the contents of the selected partition. Each call of `quartus_eda` writes one netlist. If you write out the design one partition at a time, using the `exclude_sub_partitions` flag, you need to call `quartus_eda` for each partition in the design including the root.

You can specify the `root_partition` as the partition name in the `--partition` option to get the top level partition,which. is useful when using the `--exclude_sub_partitions` flag.

--module_name

The `--module_name` option allows you to rename a partition name in the generated netlist file. By default, the software uses the partition name as the module name in the netlist file. This option is only valid when you use the `--partition` option. You can rename any module using `--module_name=abc=xyz`. The generated file names format is: *<revision>.<partition name>.<vo or vho>*. By default, the software writes the netlist file to the `simulation` directory (e.g. `simulation/modelsim`), unless you specify an `output_directory` (using a command line option or `.qsf` assignment).

Related Information

- Intel Quartus Prime Pro Edition User Guide: PCB Design Tools
  The netlist writer produces IBIS models for IO simulation.
- Generating a VQM Netlist for other EDA Tools

1.9. Simulating Intel FPGA Designs Revision History

This document has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
2020.10.10	20.1	Renamed `--rename` to `--module_name` in *The EDA Netlist Writer and Gate-level Netlists*
2020.04.30	20.1	Added *The EDA Netlist Writer and Gate-level Netlists*
2019.04.01	19.1.0	• Updated supported simulator versions.
2018.12.19	18.1.0	• Added Simulator Support for Mentor Verification IP Bus Functional Models (BFMs) topic.
2018.09.24	18.1.0	• Updated supported simulator versions.
2018.05.07	18.0.0	• Updated list of supported simulation tools to include Cadence Xcelium Parallel Simulator. • Added `xcelium_setup.sh` to list of simulation setup scripts. • Added *Sourcing Xcelium Simulation Setup Scripts* topic.

Date	Version	Changes
2017.11.06	17.1.0	• Added Simulation Library Compiler details and another step to Quick Start Example
2017.05.08	17.0.0	• Updated simulator support table with latest version information.
2016.10.31	16.1.0	• Implemented Intel rebranding. • Removed information about gate-level timing simulation. • Updated simulator support table with latest version information.
2016.05.02	16.0.0	• Removed support for NativeLink in Pro Edition. Added NativeLink support limitations for Standard Edition. • Updated simulator support table with latest version information.
2015.11.02	15.1.0	• Added new Generating Version-Independent IP Simulation Scripts topic. • Added example IP simulation script templates for all supported simulators. • Added new Incorporating IP Simulation Scripts in Top-Level Scripts topic. • Updated simulator support table with latest version information. • Changed instances of *Quartus II* to *Quartus Prime*.
2015.05.04	15.0.0	• Updated simulator support table with latest. • Gate-level timing simulation limited to Stratix IV and Cyclone IV devices. • Added mixed language simulation support in the ModelSim - Intel FPGA Edition software.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
May 2013	13.0.0	• Updated introductory section and system and IP file locations.
November 2012	12.1.0	• Revised chapter to reflect latest changes to other simulation documentation.
June 2012	12.0.0	• Reorganization of chapter to reflect various simulation flows. • Added NativeLink support for newer IP cores.
November 2011	11.1.0	• Added information about encrypted Altera simulation model files. • Added information about IP simulation and NativeLink.

2. ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim

You can include your supported EDA simulator in the Intel Quartus Prime design flow. This document provides guidelines for simulation of designs with ModelSim or QuestaSim software. The entry-level ModelSim - Intel FPGA Edition includes precompiled simulation libraries.

Note: The latest version of the ModelSim - Intel FPGA Edition software supports native, mixed-language (VHDL/Verilog HDL/SystemVerilog) co-simulation of plain text HDL. If you have a VHDL-only simulator, you can use the ModelSim-Intel FPGA Edition software to simulate Verilog HDL modules and IP cores. Alternatively, you can purchase separate co-simulation software.

Related Information

- Simulating Intel FPGA Designs on page 4
- Managing Intel Quartus Prime Projects

2.1. Quick Start Example (ModelSim with Verilog)

You can adapt the following RTL simulation example to get started quickly with ModelSim:

1. To specify your EDA simulator and executable path, type the following Tcl package command in the Intel Quartus Prime tcl shell window:

   `set_user_option -name EDA_TOOL_PATH_MODELSIM <modelsim executable path>`

   `set_global_assignment -name EDA_SIMULATION_TOOL "MODELSIM (verilog)"`

2. Compile simulation model libraries using one of the following methods:
   - To automatically compile all required simulation model libraries for your design in your supported simulator, click **Tools ➤ Launch Simulation Library Compiler**. Specify options for your simulation tool, language, target device family, and output location, and then click **OK**.
   - Type the following commands to create and map Intel FPGA simulation libraries manually, and then compile the models manually:

     ```
     vlib <lib1>_ver
     vmap <lib1>_ver <lib1>_ver
     vlog -work <lib1> <lib1>
     ```

   Use the compiled simulation model libraries during simulation of your design. Refer to your EDA simulator's documentation for information about running simulation.

3. Compile your design and testbench files:

```
vlog -work work <design or testbench name>.v
```

4. Load the design:

```
vsim -L work -L <lib1>_ver -L <lib2>_ver work.<testbench name>
```

2.2. ModelSim, ModelSim-Intel FPGA Edition, and QuestaSim Guidelines

The following guidelines apply to simulation of designs in the ModelSim, ModelSim-Intel FPGA Edition, or QuestaSim software.

2.2.1. Using ModelSim-Intel FPGA Edition Precompiled Libraries

Precompiled libraries for both functional and gate-level simulations are provided for the ModelSim-Intel FPGA Edition software. You should not compile these library files before running a simulation. No precompiled libraries are provided for ModelSim or QuestaSim. You must compile the necessary libraries to perform functional or gate-level simulation with these tools.

The precompiled libraries provided in *<install path>*`/altera/` must be compatible with the version of the Intel Quartus Prime software that creates the simulation netlist. To verify compatibility of precompiled libraries with your version of the Intel Quartus Prime software, refer to the *<install path>*`/altera/version.txt` file. This file indicates the Intel Quartus Prime software version and build of the precompiled libraries.

Note: Encrypted simulation model files shipped with the Intel Quartus Prime software version 10.1 and later can only be read by ModelSim-Intel FPGA Edition software version 6.6c and later. These encrypted simulation model files are located at the *<Intel Quartus Prime System directory>*`/quartus/eda/sim_lib/`*<mentor>* directory.

2.2.2. Passing Parameter Information from Verilog HDL to VHDL

You must use in-line parameters to pass values from Verilog HDL to VHDL.

By default, the **x_on_violation_option** logic option is enabled for all design registers, resulting in an output of "X" at timing violation. To disable "X" propagation at timing violations on a specific register, disable the **x_on_violation_option** logic option for the specific register, as shown in the following example from the Intel Quartus Prime Settings File (`.qsf`).

```
set_instance_assignment -name X_ON_VIOLATION_OPTION OFF -to \
<register_name>
```

Example 1. In-line Parameter Passing Example

```
lpm_add_sub#(.lpm_width(12), .lpm_direction("Add"),
.lpm_type("LPM_ADD_SUB"),
.lpm_hint("ONE_INPUT_IS_CONSTANT=NO,CIN_USED=NO" ))

lpm_add_sub_component (
```

```
        .dataa (dataa),
        .datab (datab),
        .result (sub_wire0)
);
```

Note: The sequence of the parameters depends on the sequence of the GENERIC in the VHDL component declaration.

2.2.3. Increasing Simulation Speed

By default, the ModelSim and QuestaSim software runs in a debug-optimized mode.

To run the ModelSim and QuestaSim software in speed-optimized mode, add the following two vlog command-line switches. In this mode, module boundaries are flattened and loops are optimized, which eliminates levels of debugging hierarchy and may result in faster simulation. This switch is not supported in the ModelSim-Intel FPGA Edition simulator.

```
vlog -fast -05
```

2.2.4. Viewing Simulation Messages

ModelSim and QuestaSim error and warning messages are tagged with a vsim or vcom code. To determine the cause and resolution for a vsim or vcom error or warning, use the verror command.

For example, ModelSim may return the following error:

```
# ** Error: C:/altera_trn/DUALPORT_TRY/simulation/modelsim/DUALPORT_TRY.vho(31):
 (vcom-1136) Unknown identifier "stratixiv"
```

In this case, type the following command:

```
verror 1136
```

The following description appears:

```
# vcom Message # 1136:
# The specified name was referenced but was not found. This indicates
# that either the name specified does not exist or is not visible at
# this point in the code.
```

Note: If your design includes deep levels of hierarchy, and the **Maintain hierarchy** EDA tools option is turned on, this may result in a large number of module instances in post-fit or post-map netlist. This condition can exceed the ModelSim-Intel FPGA Edition instance limitation.

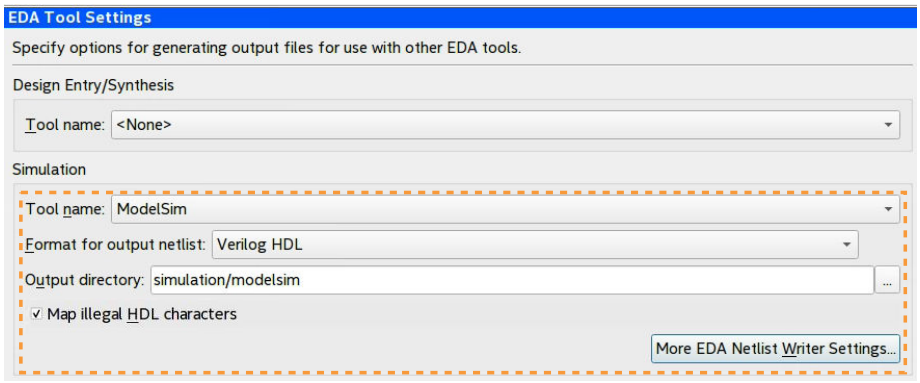
To avoid exceeding any ModelSim-Intel FPGA Edition instance limit, turn off **Maintain hierarchy** to reduce the number of modules instances to 1 in the post-fit or post-map netlist. To access this option, click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ More Settings**.

2.2.5. Generating Signal Activity Data for Power Analysis

Follow these steps to generate and use simulation signal activity data for power analysis:

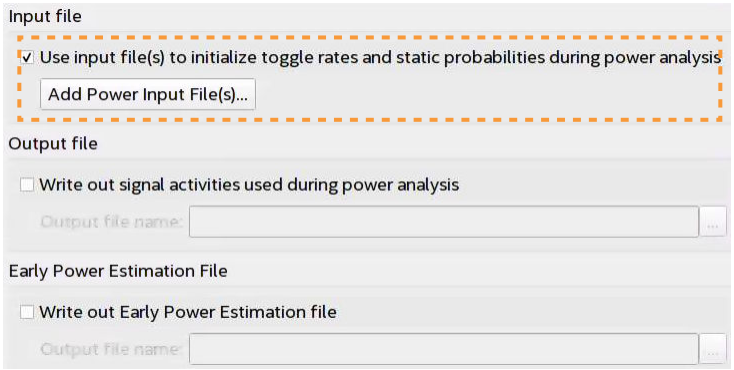
1. To run full compilation on your design, click **Processing ➤ Start Compilation**.
2. To specify settings for output of simulation files, click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ Simulation**. Select your simulator in **Tool name** and the **Format for output netlist** and **Output directory**.
3. Turn on **Map illegal HDL characters**. This setting directs the EDA Netlist Writer to map illegal characters for VHDL or Verilog HDL, and results in more accurate data for power analysis.

Figure 2. EDA Tool Settings for Simulation



4. For Intel Stratix® 10 designs, to generate a Standard Delay Output (.sdo) file that includes back-annotation of delays for power analysis, refer to Generating Standard Delay Output for Power Analysis on page 25.
5. In the Intel Quartus Prime software, click **Processing ➤ Power Analyzer Tool**. The **Power Analyzer** tab appears.
6. Under **Input file**, turn on **Use input files to initialize toggle rates and static probabilities during power analysis**, and then click **Add Power Input Files**. The **Power Analyzer Settings** page appears.

Figure 3. Specifying Power Analysis Input Files



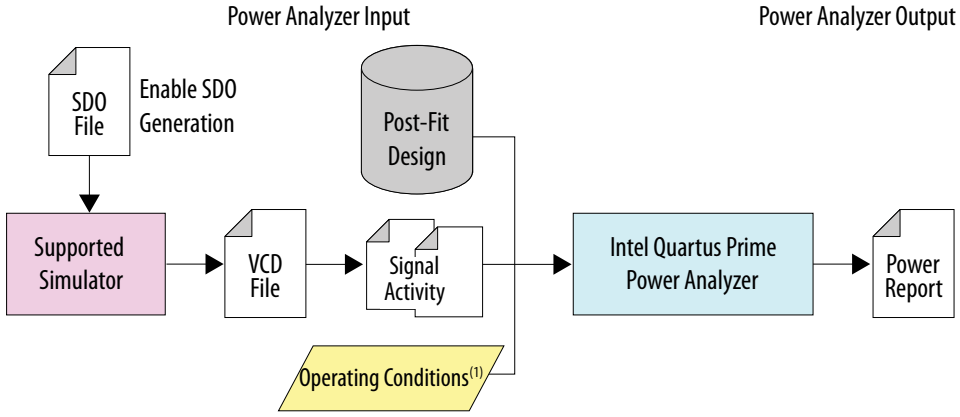
7. To specify a `.vcd` for power analysis, click **Add** and specify the **File name**, **Entity**, and **Simulation period** for the `.vcd`.
8. To enable glitch filtering during power analysis with the `.vcd` you generate, turn on **Perform glitch filtering on VCD files**.
9. To run the power analysis, click **Start** on the **Power Analyzer** tab. View the toggle rates in the power analysis results.

2.2.5.1. Generating Standard Delay Output for Power Analysis

To improve accuracy of power analysis, you can generate a Standard Delay Output (`.sdo`) file that includes back-annotated delay estimates for ModelSim simulation. ModelSim simulation can then output a more accurate `.vcd` for use as power analysis input. You must run **Fitter (Finalize)** before generating the `.sdo`.

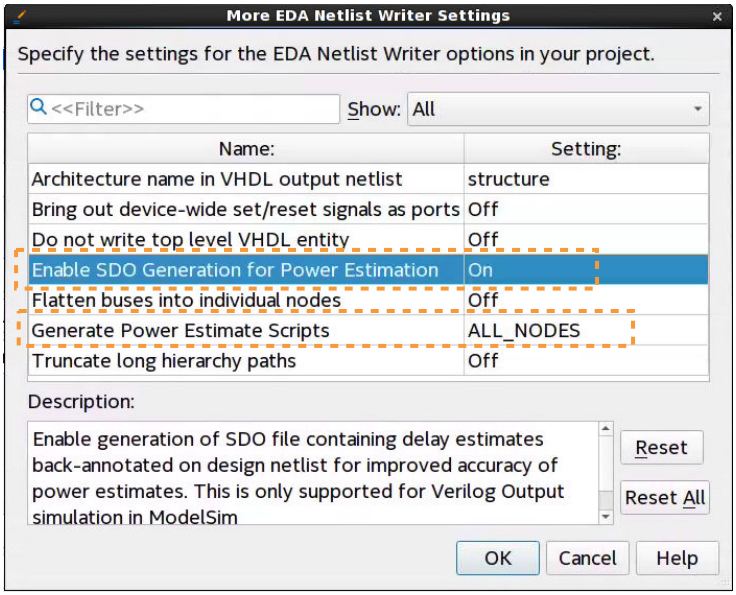
Note: The EDA Netlist Writer currently supports `.sdo` file generation only for Verilog `.vo` simulation in the ModelSim simulator (not ModelSim - Intel FPGA Edition) for Intel Stratix 10 designs. The EDA Netlist Writer does not currently support `.sdo` file generation for any other simulator or device family.

Figure 4. Using an SDO in Power Analysis



1. Click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ Simulation**. In **Tool name** select **ModelSim** and **Verilog** for **Format for output netlist**.
2. Click **More EDA Netlist Writer Settings**. Set **Enable SDO Generation for Power Estimation** to **On**. Set **Generate Power Estimate Scripts** to **ALL_NODES**.

Figure 5. More EDA Netlist Writer Settings



3. To run the Fitter, click **Processing ➤ Start ➤ Start Fitter (Finalize)**.
4. Create a representative testbench (`.vt`) that exercises the design functions appropriately.
5. To specify the appropriate hierarchy level for signals in the output `.vcd`, add the following line to the project `.qsf` file:

```
set_global_assignment -name EDA_TEST_BENCH_DESIGN_INSTANCE_NAME
    <DUT instance path> -section_id eda_simulation
```

(1)

6. After Fitter processing is complete, click **Processing ➤ Start ➤ Start EDA Netlist Writer**. EDA Netlist Writer generates the following files in `/<project>/simulation/modelsim/power/`:
   - `<project>.vo` (contains a reference to the `.sdo` file by default)
   - `<project>_dump_all_vcd_nodes.tcl`—specifies nodes to save in `.vcd`
   - `<project>_v.sdo`—back-annotated delay estimates
7. Create a ModelSim script (`.do`) to load the design and testbench, start ModelSim, and then source the `.do` script.
8. To specify the signals ModelSim includes in the `.vcd` file, source `*_dump_all_vcd_nodes.tcl` in ModelSim.
9. To generate the `.vcd` file, simulate the test bench and netlist in ModelSim. The `.vcd` file generates according to your specifications.
10. Specify the `.vcd` as an input to power analysis, as Generating Signal Activity Data for Power Analysis on page 24 describes.

(1) Specify the full hierarchical path in the testbench, not just the instance name. For example, specify a|b|c, not just c.

2.2.6. Viewing Simulation Waveforms

ModelSim-Intel FPGA Edition, ModelSim, and QuestaSim automatically generate a Wave Log Format File (`.wlf`) following simulation. You can use the `.wlf` to generate a waveform view.

To view a waveform from a `.wlf` through ModelSim-Intel FPGA Edition, ModelSim, or QuestaSim, perform the following steps:

1. Type `vsim` at the command line. The **ModelSim/QuestaSim** or **ModelSim-Intel FPGA Edition** dialog box appears.
2. Click **File ➤ Datasets**. The **Datasets Browser** dialog box appears.
3. Click **Open** and select your `.wlf`.
4. Click **Done**.
5. In the Object browser, select the signals that you want to observe.
6. Click **Add ➤ Wave**, and then click **Selected Signals**.
   You must first convert the `.vcd` to a `.wlf` before you can view a waveform in ModelSim-Intel FPGA Edition, ModelSim, or QuestaSim.
7. To convert the `.vcd` to a `.wlf`, type the following at the command-line:

   ```
   vcd2wlf <example>.vcd <example>.wlf
   ```

8. After conversion, view the `.wlf` waveform in ModelSim or QuestaSim.

2.2.7. Simulating with ModelSim-Intel FPGA Edition Waveform Editor

You can use the ModelSim-Intel FPGA Edition waveform editor as a simple method to create stimulus vectors for simulation. You can create this design stimulus via interactive manipulation of waveforms from the wave window in ModelSim-Intel FPGA Edition. With the ModelSim-Intel FPGA Edition waveform editor, you can create and edit waveforms, drive simulation directly from created waveforms, and save created waveforms into a stimulus file.

Related Information

ModelSim Web Page

2.3. ModelSim Simulation Setup Script Example

The Intel Quartus Prime software can generate a msim_setup.tcl simulation setup script for IP cores in your design. The script compiles the required device library models, compiles the design files, and elaborates the design with or without simulator optimization. To run the script, type source `msim_setup.tcl` in the simulator Transcript window.

Alternatively, if you are using the simulator at the command line, you can type the following command:

```
vsim -c -do msim_setup.tcl
```

In this example the `top-level-simulate.do` custom top-level simulation script sets the hierarchy variable `TOP_LEVEL_NAME` to `top_testbench` for the design, and sets the variable `QSYS_SIMDIR` to the location of the generated simulation files.

```
# Set hierarchy variables used in the IP-generated files
set TOP_LEVEL_NAME "top_testbench"
set QSYS_SIMDIR "./ip_top_sim"
# Source generated simulation script which defines aliases used below
source $QSYS_SIMDIR/mentor/msim_setup.tcl
# dev_com alias compiles simulation libraries for device library files
dev_com
# com alias compiles IP simulation or Qsys model files and/or Qsys model files
in the correct order
com
# Compile top level testbench that instantiates your IP
vlog -sv ./top_testbench.sv
# elab alias elaborates the top-level design and testbench
elab
# Run the full simulation
run - all
```

In this example, the top-level simulation files are stored in the same directory as the original IP core, so this variable is set to the IP-generated directory structure. The `QSYS_SIMDIR` variable provides the relative hierarchy path for the generated IP simulation files. The script calls the generated `msim_setup.tcl` script and uses the alias commands from the script to compile and elaborate the IP files required for simulation along with the top-level simulation testbench. You can specify additional simulator elaboration command options when you run the `elab` command, for example, `elab +nowarnTFMPC`. The last command run in the example starts the simulation.

2.4. Unsupported Features

The Intel Quartus Prime software does not support the following ModelSim simulation features:

- Intel Quartus Prime does not support companion licensing for ModelSim.
- The USB software guard is not supported by versions earlier than ModelSim software version 5.8d.
- For ModelSim software versions prior to 5.5b, use the **PCLS** utility included with the software to set up the license.
- Some versions of ModelSim and QuestaSim support SystemVerilog, PSL assertions, SystemC, and more. For more information about specific feature support, refer to Mentor Graphics literature.
- The ModelSim - Intel FPGA Edition software license does not support Remote Desktop access with node-locked, uncounted licenses.

Related Information

ModelSim-Intel FPGA Edition Software Web Page

2.5. ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim Revision History

Document Version	Intel Quartus Prime Version	Changes
2019.06.19	19.1.0	• Added footnote about ModelSim Remote Desktop limits to "Unsupported Features" topic.
2019.04.01	19.1.0	• Described new support for generation of SDO for use in power analysis.
2017.11.06	17.1.0	• Changed title to ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim Support* • Removed Simulating Transport Delays and Disabling Timing Violations on Registers topics. Intel Quartus Prime Pro Edition does not support timing simulation. • Added Simulation Library Compiler details and another step to Quick Start Example

Date	Version	Changes
2017.05.08	17.0.0	• Removed note about unsupported NativeLink gate level simulation.
2016.10.31	16.1.0	• Implemented Intel rebranding. • Corrected load design syntax error.
2016.05.02	16.0.0	• Removed support for NativeLink simulation in Pro Edition • Added note about avoiding ModelSim - Intel FPGA Edition instance limitations.
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Quartus Prime*.
2015.05.04	15.0.0	• Added mixed language simulation support in the ModelSim - Intel FPGA Edition software.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2012	12.1.0	• Relocated general simulation information to Simulating Altera Designs.
June 2012	12.0.0	• Removed survey link.
November 2011	11.0.1	• Changed to new document template.

3. Synopsys VCS and VCS MX Support

You can include your supported EDA simulator in the *Intel Quartus Prime* design flow. This document provides guidelines for simulation of *Intel Quartus Prime* designs with the Synopsys VCS or VCS MX software.

3.1. Quick Start Example (VCS with Verilog)

You can adapt the following RTL simulation example to get started quickly with VCS:

1. To specify your EDA simulator and executable path, type the following Tcl package command in the Intel Quartus Prime tcl shell window:

   ```
   set_user_option -name EDA_TOOL_PATH_VCS <VCS executable path>
   set_global_assignment -name EDA_SIMULATION_TOOL "VCS"
   ```

2. Compile simulation model libraries using one of the following methods:
   - To automatically compile all required simulation model libraries for your design in your supported simulator, click **Tools ➤ Launch Simulation Library Compiler**. Specify options for your simulation tool, language, target device family, and output location, and then click **OK**.

   Use the compiled simulation model libraries during simulation of your design. Refer to your EDA simulator's documentation for information about running simulation.
3. Modify the `simlib_comp.vcs` file to specify your design and testbench files.
4. Type the following to run the VCS simulator:

   ```
   vcs -R -file simlib_comp.vcs
   ```

3.2. VCS and VCS MX Guidelines

The following guidelines apply to simulation of Intel FPGA designs in the VCS or VCS MX software:

- Do not specify the -v option for `altera_lnsim.sv` because it defines a systemverilog package.
- Add `-verilog` and `+verilog2001ext+.v` options to make sure all **.v** files are compiled as verilog 2001 files, and all other files are compiled as systemverilog files.
- Add the `-lca` option for Stratix V and later families because they include IEEE-encrypted simulation files for VCS and VCS MX.
- Add `-timescale=1ps/1ps` to ensure picosecond resolution.

3.3. VCS Simulation Setup Script Example

The Intel Quartus Prime software can generate a simulation setup script for IP cores in your design. The scripts contain shell commands that compile the required simulation models in the correct order, elaborate the top-level design, and run the simulation for 100 time units by default. You can run these scripts from a Linux command shell.

The scripts for VCS and VCS MX are **vcs_setup.sh** (for Verilog HDL or SystemVerilog) and **vcsmx_setup.sh** (combined Verilog HDL and SystemVerilog with VHDL). Read the generated **.sh** script to see the variables that are available for override when sourcing the script or redefining directly if you edit the script. To set up the simulation for a design, use the command-line to pass variable values to the shell script.

Example 2. Using Command-line to Pass Simulation Variables

```
sh vcsmx_setup.sh\
USER_DEFINED_ELAB_OPTIONS=+rad\
USER_DEFINED_SIM_OPTIONS=+vcs+lic+wait
```

Example 3. Example Top-Level Simulation Shell Script for VCS-MX

```
# Run generated script to compile libraries and IP simulation files
# Skip elaboration and simulation of the IP variation
sh ./ip_top_sim/synopsys/vcsmx/vcsmx_setup.sh SKIP_ELAB=1 SKIP_SIM=1
QSYS_SIMDIR="./ip_top_sim"
#Compile top-level testbench that instantiates IP
vlogan -sverilog ./top_testbench.sv
#Elaborate and simulate the top-level design
vcs –lca –t ps <elaboration control options> top_testbench
simv <simulation control options>
```

Example 4. Example Top-Level Simulation Shell Script for VCS

```
# Run script to compile libraries and IP simulation files
sh ./ip_top_sim/synopsys/vcs/vcs_setup.sh TOP_LEVEL_NAME=”top_testbench”\
# Pass VCS elaboration options to compile files and elaborate top-level
 passed to the script as the TOP_LEVEL_NAME
USER_DEFINED_ELAB_OPTIONS="top_testbench.sv"\
# Pass in simulation options and run the simulation for specified amount of
time.
USER_DEFINED_SIM_OPTIONS=”<simulation control options>
```

3.4. Synopsys VCS and VCS MX Support Revision History

Document Version	Intel Quartus Prime Version	Changes
2017.11.06	17.1.0	• Removed Simulating Transport Delays and Disabling Timing Violations on Registers topics. Intel Quartus Prime Pro Edition does not support timing simulation. • Added Simulation Library Compiler details and another step to Quick Start Example

Date	Version	Changes
2017.05.08	17.0.0	• Removed note about unsupported NativeLink gate level simulation.
2016.10.31	16.1.0	• Implemented Intel rebranding. • Removed support for `.vcd` file generation.
2016.05.02	16.0.0	• Removed support for NativeLink simulation in Pro Edition
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2012	12.1.0	• Relocated general simulation information to Simulating Altera Designs.
June 2012	12.0.0	• Removed survey link.
November 2011	11.0.1	• Changed to new document template.

Related Information

Documentation Archive

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

4. Aldec Active-HDL and Riviera-PRO * Support

You can include your supported EDA simulator in the *Intel Quartus Prime* design flow. This chapter provides specific guidelines for simulation of *Intel Quartus Prime* designs with the Aldec Active-HDL or Riviera-PRO software.

4.1. Quick Start Example (Active-HDL VHDL)

You can adapt the following RTL simulation example to get started quickly with Active-HDL:

1. To specify your EDA simulator and executable path, type the following Tcl package command in the Intel Quartus Prime tcl shell window:

   `set_user_option -name EDA_TOOL_PATH_ACTIVEHDL <Active HDL executable path>`

   `set_global_assignment -name EDA_SIMULATION_TOOL "Active-HDL (VHDL)"`

2. Compile simulation model libraries using one of the following methods:
   - To automatically compile all required simulation model libraries for your design in your supported simulator, click **Tools ➤ Launch Simulation Library Compiler**. Specify options for your simulation tool, language, target device family, and output location, and then click **OK**.
   - Compile Intel FPGA simulation models manually:

     ```
     vlib <library1> <altera_library1>
     vcom -strict93 -dbg -work <library1> <lib1_component/pack.vhd>
     <lib1.vhd>
     ```

   Use the compiled simulation model libraries during simulation of your design. Refer to your EDA simulator's documentation for information about running simulation.
3. Open the Active-HDL simulator.
4. Create and open the workspace:

   ```
   createdesign <workspace name> <workspace path>
   opendesign -a <workspace name>.adf
   ```

5. Create the work library and compile the netlist and testbench files:

   ```
   vlib work
   vcom  -strict93  -dbg -work work <output netlist> <testbench file>
   ```

6. Load the design:

   ```
   vsim +access+r -t 1ps +transport_int_delays +transport_path_delays \
   -L work -L <lib1> -L <lib2> work.<testbench module name>
   ```

7. Run the simulation in the Active-HDL simulator.

4.2. Aldec Active-HDL and Riviera-PRO Guidelines

The following guidelines apply to simulating Intel FPGA designs in the Active-HDL or Riviera-PRO software.

4.2.1. Compiling SystemVerilog Files

If your design includes multiple SystemVerilog files, you must compile the System Verilog files together with a single alog command. If you have Verilog files and SystemVerilog files in your design, you must first compile the Verilog files, and then compile only the SystemVerilog files in the single `alog` command.

4.3. Using Simulation Setup Scripts

The Intel Quartus Prime software generates the `rivierapro_setup.tcl` simulation setup script for IP cores in your design. The use and content of the script file is similar to the `msim_setup.tcl` file used by the ModelSim simulator.

Related Information

Simulating IP Cores

4.4. Aldec Active-HDL and Riviera-PRO * Support Revision History

Document Version	Intel Quartus Prime Version	Changes
2017.11.06	17.1.0	• Removed Simulating Transport Delays and Disabling Timing Violations on Registers topics. Intel Quartus Prime Pro Edition does not support timing simulation. • Added Simulation Library Compiler details and another step to Quick Start Example

Date	Version	Changes
2017.05.08	17.0.0	• Removed note about unsupported NativeLink gate level simulation.
2016.10.31	16.1.0	• Implemented Intel rebranding.
2016.05.02	16.0.0	• Removed support for NativeLink simulation in Pro Edition
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2012	12.1.0	• Relocated general simulation information to Simulating Altera Designs.
June 2012	12.0.0	• Removed survey link.
November 2011	11.0.1	• Changed to new document template.

Related Information

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

5. Cadence Simulator Support

You can include your supported EDA simulator in the *Intel Quartus Prime Pro Edition* design flow. This chapter provides specific guidelines for simulation of *Intel Quartus Prime Pro Edition* designs with the Cadence Incisive Enterprise (IES) and Xcelium Parallel Simulator software.

5.1. Quick Start Example (NC-Verilog)

You can adapt the following RTL simulation example to get started quickly with IES:

1. Click **View ➤ TCL Console** to open the **TCL Console**.
2. To specify your EDA simulator and executable path, type the following Tcl package command in the Intel Quartus Prime tcl shell window:

   `set_user_option -name EDA_TOOL_PATH_NCSIM <ncsim executable path>`

   `set_global_assignment -name EDA_SIMULATION_TOOL "NC-Verilog (Verilog)"`

3. Compile simulation model libraries using one of the following methods:
   - To automatically compile all required simulation model libraries for your design in your supported simulator, click **Tools ➤ Launch Simulation Library Compiler**. Specify options for your simulation tool, language, target device family, and output location, and then click **OK**.
   - You can also compile Intel FPGA simulation libraries from the command-line:

     ```
     quartus_sh --simlib_comp -tool ncsim -family <device family>
       -language <language> -gen_only -cmd_file <sim_script_file_name>
     ```

     This generates the `cds.lib`, `hdl.var` and, `<sim_script_file_name>`, which can be used to compile the simulation libraries.

   Use the compiled simulation model libraries during simulation of your design. Refer to your EDA simulator's documentation for information about running simulation.
4. Elaborate your design and testbench with IES:

   ```
   ncelab <work library>.<top-level entity name>
   ```

5. Run the simulation:

   ```
   ncsim <work library>.<top-level entity name>
   ```

5.2. Using GUI or Command-Line Interfaces

Intel FPGA supports both the IES GUI and command-line simulator interfaces, and command-line support for Xcelium Parallel Simulator.

To start the IES GUI, type `nclaunch` at a command prompt.

Table 7. IES Simulation Executables

Program	Function
`ncvlog`	`ncvlog` compiles your Verilog HDL code and performs syntax and static semantics checks.
`ncvhdl`	`ncvhdl` compiles your VHDL code and performs syntax and static semantics checks.
`ncelab`	Elaborates the design hierarchy and determines signal connectivity.
`ncsdfc`	Performs back-annotation for simulation with VHDL simulators.
`ncsim`	Runs mixed-language simulation. This program is the simulation kernel that performs event scheduling and executes the simulation code.

Table 8. Xcelium Simulation Executables

Program	Function
`xmvlog`	`xmvlog` compiles your Verilog HDL code and performs syntax and static semantics checks.
`xmvhdl`	`xmvhdl` compiles your VHDL code and performs syntax and static semantics checks.
`xmelab`	Elaborates the design hierarchy and determines signal connectivity.
`xmsim`	Runs mixed-language simulation. This program is the simulation kernel that performs event scheduling and executes the simulation code.

5.3. Cadence Incisive Enterprise (IES) Guidelines

The following guidelines apply to simulation of Intel FPGA designs in the IES software:

- Do not specify the -v option for `altera_lnsim.sv` because it defines a systemverilog package.
- Add `-verilog` and `+verilog2001ext+.v` options to make sure all **.v** files are compiled as verilog 2001 files, and all other files are compiled as systemverilog files.
- Add the `-lca` option for Stratix V and later families because they include IEEE-encrypted simulation files for IES.
- Add `-timescale=1ps/1ps` to ensure picosecond resolution.

5.3.1. Simulating Pulse Reject Delays

By default, the IES software filters out all pulses that are shorter than the propagation delay between primitives. Setting the pulse reject delays options in the IES software prevents the simulation tool from filtering out these pulses. Use the following options to ensure that all signal pulses are seen in the simulation results.

Table 9. Pulse Reject Delay Options

Program	Function
`-PULSE_R`	Use when simulation pulses are shorter than the delay in a gate-level primitive. The argument is the percentage of delay for pulse reject limit for the path
`-PULSE_INT_R`	Use when simulation pulses are shorter than the interconnect delay between gate-level primitives. The argument is the percentage of delay for pulse reject limit for the path

5.3.2. Viewing Simulation Waveforms

IES generates a `.trn` file automatically following simulation. You can use the `.trn` for generating the SimVision waveform view.

To view a waveform from a `.trn` file through SimVision, follow these steps:

1. Type `simvision` at the command line. The **Design Browser** dialog box appears.
2. Click **File ➤ Open Database** and click the `.trn` file.
3. In the **Design Browser** dialog box, select the signals that you want to observe from the Hierarchy.
4. Right-click the selected signals and click **Send to Waveform Window**.

   You cannot view a waveform from a `.vcd` file in SimVision, and the `.vcd` file cannot be converted to a `.trn` file.

5.4. IES Simulation Setup Script Example

The Intel Quartus Prime software can generate a `ncsim_setup.sh` simulation setup script for IP cores in your design. The script contains shell commands that compile the required device libraries, IP, or Platform Designer simulation models in the correct order. The script then elaborates the top-level design and runs the simulation for 100 time units by default. You can run these scripts from a Linux command shell. To set up the simulation script for a design, you can use the command-line to pass variable values to the shell script.

Read the generated `.sh` script to see the variables that are available for you to override when you source the script or that you can redefine directly in the generated `.sh` script. For example, you can specify additional elaboration and simulation options with the variables `USER_DEFINED_ELAB_OPTIONS` and `USER_DEFINED_SIM_OPTIONS`.

Example 5. Example Top-Level Simulation Shell Script for Incisive (NCSIM)

```
# Run script to compile libraries and IP simulation files
# Skip elaboration and simulation of the IP variation
sh ./ip_top_sim/cadence/ncsim_setup.sh SKIP_ELAB=1 SKIP_SIM=1 QSYS_SIMDIR="./
ip_top_sim"

#Compile the top-level testbench that instantiates your IP
ncvlog -sv ./top_testbench.sv
#Elaborate and simulate the top-level design
ncelab <elaboration control options> top_testbench
ncsim <simulation control options> top_testbench
```

5.5. Cadence Simulator Support Revision History

Document Version	Intel Quartus Prime Version	Changes
2018.05.07	18.0.0	• Renamed chapter for Xcelium Parallel Simulator support. • Added Xcelium command-line support. • Updated commands in the Quick Start Example.

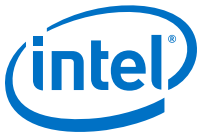


Table 10. Document Revision History

Date	Version	Changes
2017.11.15	17.1.1	• Removed Elaborating Your Design and Back-Annotating Simulation Timing Data topics. These topics apply only to timing simulation in Intel Quartus Prime Standard Edition.
2017.11.06	17.1.0	• Removed Simulating Transport Delays and Disabling Timing Violations on Registers topics. Intel Quartus Prime Pro Edition does not support timing simulation. • Added Simulation Library Compiler details and another step to Quick Start Example
2016.10.31	16.1.0	• Implemented Intel rebranding.
2016.05.02	16.0.0	• Removed support for NativeLink simulation in Pro Edition
2015.11.02	15.1.0	Changed instances of *Quartus II* to *Intel Quartus Prime*.
2014.08.18	14.0.a10.0	• Corrected incorrect references to VCS and VCS MX.
2014.06.30	14.0.0	• Replaced MegaWizard Plug-In Manager information with IP Catalog.
November 2012	12.1.0	• Relocated general simulation information to Simulating Altera Designs.
June 2012	12.0.0	• Removed survey link.
November 2011	11.0.1	• Changed to new document template.

Related Information

Documentation Archive

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

6. Intel Quartus Prime Pro Edition User Guide Third-party Simulation Document Archive

If an Intel Quartus Prime software version is not listed, the user guide for the previous version applies.

Intel Quartus Prime Software Version	User Guide
19.1	Intel Quartus Prime Pro Edition User Guide Third-party Simulation
18.1	Intel Quartus Prime Pro Edition User Guide Third-party Simulation

A. Intel Quartus Prime Pro Edition User Guides

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

Related Information

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

- Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration
  Describes Partial Reconfiguration, an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Define multiple personas for a particular design region, without impacting operation in other areas.
- Intel Quartus Prime Pro Edition User Guide: Third-party Simulation
  Describes RTL- and gate-level design simulation support for third-party simulation tools by Aldec*, Cadence*, Mentor Graphics, and Synopsys that allow you to verify design behavior before device programming. Includes simulator support, simulation flows, and simulating Intel FPGA IP.
- Intel Quartus Prime Pro Edition User Guide: Third-party Synthesis
  Describes support for optional synthesis of your design in third-party synthesis tools by Mentor Graphics, and Synopsys. Includes design flow steps, generated file descriptions, and synthesis guidelines.
- Intel Quartus Prime Pro Edition User Guide: Third-party Logic Equivalence Checking Tools
  Describes support for optional logic equivalence checking (LEC) of your design in third-party LEC tools by OneSpin*.
- Intel Quartus Prime Pro Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Pro Edition in-system design debugging tools for real-time verification of your design. These tools provide visibility by routing (or "tapping") signals in your design to debugging logic. These tools include System Console, Signal Tap logic analyzer, system debugging toolkits, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Pro Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Pro Edition Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology.
- Intel Quartus Prime Pro Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Pro Edition Power Analysis tools that allow accurate estimation of device power consumption. Estimate the power consumption of a device to develop power budgets and design power supplies, voltage regulators, heat sink, and cooling systems.
- Intel Quartus Prime Pro Edition User Guide: Design Constraints
  Describes timing and logic constraints that influence how the Compiler implements your design, such as pin assignments, device options, logic options, and timing constraints. Use the Interface Planner to prototype interface implementations, plan clocks, and quickly define a legal device floorplan. Use the Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Pro Edition User Guide: PCB Design Tools
  Describes support for optional third-party PCB design tools by Mentor Graphics and Cadence*. Also includes information about signal integrity analysis and simulations with HSPICE and IBIS Models.
- Intel Quartus Prime Pro Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Pro Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.