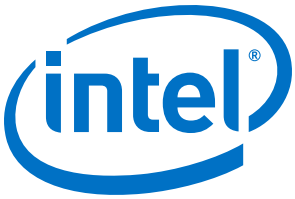


Simulating Intel® FPGA Designs

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	Parallel Simulator	18.03	Linux 64-bit only
Mentor Graphics*		10.6d	Windows, Linux, 32-bit only
Mentor Graphics	PE	10.6d	Windows 32-bit only
Mentor Graphics	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

Table 2. Supported Simulators

Vendor	Simulator	Version	Platform
Aldec	Active-HDL	10.3	Windows
Aldec	Riviera-PRO	2015.10	Windows, Linux
Cadence	Incisive Enterprise*	14.20	Linux
Mentor Graphics		10.5b	Windows, Linux
Mentor Graphics	PE	10.4d	Windows
Mentor Graphics	SE	10.4d	Windows, Linux
Mentor Graphics	QuestaSim	10.4d	Windows, Linux
Synopsys	VCS VCS MX	2014,12-SP1	Linux

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 (including), and QuestaSim 10.1d.
- Synopsys and 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 3. 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 • Platform Designer (Standard)-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 • Intel simulation libraries • Post-synthesis or post-fit functional netlist • Intel FPGA IP bus functional models
Gate-level timing	Simulation using a post-fit timing netlist, testing functional and timing performance. Supported only for the Arria® II GX/GZ,Cyclone® IV, MAX® II, MAX V, and Stratix® IV device families.	• Testbench • Intel 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 Arria II GX/GZ,Cyclone IV, MAX II, MAX V, and Stratix IV device families.. Use Timing Analyzer static timing analysis rather than gate-level timing simulation.

1.3. HDL Support

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

Table 4. 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. • To use NativeLink automation, analyze and elaborate your design in the Intel Quartus Prime software, and then use the NativeLink simulator scripts to compile the design files in your simulator. • 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 software, along with precompiled Intel FPGA simulation libraries, to simplify simulation of Intel FPGA designs. Starting in version 15.0, the 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 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 5. 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.
NativeLink Simulation Flow	NativeLink automates Intel Quartus Prime integration with your EDA simulator. Setup NativeLink to generate simulation scripts, compile simulation libraries, and automatically launch your simulator following design compilation. Specify your own compilation, elaboration, and simulation scripts for testbench and simulation model files. Do not use NativeLink if you require direct control over every aspect of simulation. *Note:* The Intel Quartus Prime Pro Edition software does not support NativeLink simulation.
Specialized Simulation Flows	Supports specialized simulation flows for specific design variations, including the following:

continued...

Simulation Flow	Description
	• 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 (Standard)* 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
- Creating a System With Platform Designer (Standard)

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:

- Use the NativeLink feature to automatically compile your design, Intel FPGA IP, simulation model libraries, and testbench.
- 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.

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**.
- To edit parameters and regenerate synthesis or simulation files for an existing IP core variation, click **View ➤ Utility Windows ➤ Project Navigator ➤ IP Components**.

Table 6. 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

continued...

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

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

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

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

To generate IP functional simulation models:

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

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

Related Information

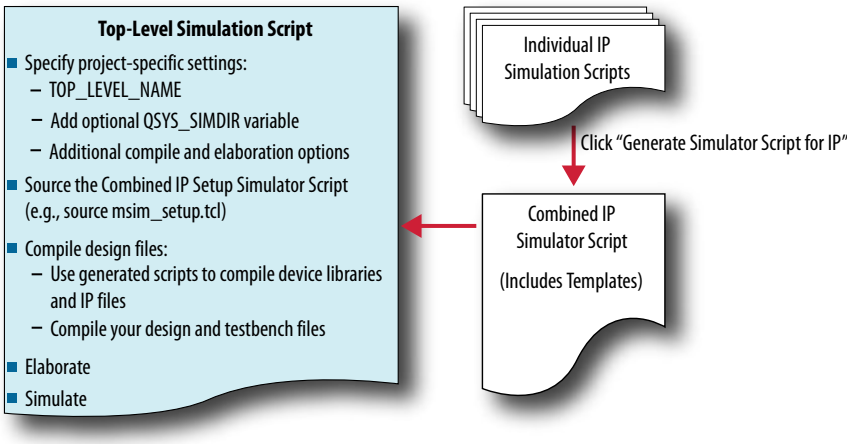
AN 343: Intel FPGA IP Evaluation Mode of AMPP IP

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 7. 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	`ip-setup-simulation`
	` --quartus-project=<my proj>`
	` --output-directory=<my_dir>`
	` --use-relative-paths`
	` --compile-to-work`

continued...

Utility	Syntax
into a single work library if your simulation environment requires. Use the `--use-relative-paths` option to use relative paths whenever possible.	`--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 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. Using NativeLink Simulation (Intel Quartus Prime Standard Edition)

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

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

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

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

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

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

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

Table 8. Execution Paths for EDA Simulators

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

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

1.7.2. Running RTL Simulation (NativeLink Flow)

To run RTL simulation using the NativeLink flow, follow these steps:

1. Set up the simulation environment.
2. Click **Processing > Start > Analysis and Elaboration**.
3. Click **Tools > Run Simulation Tool > RTL Simulation**.

   NativeLink compiles simulation libraries and launches and runs your RTL simulator automatically according to the NativeLink settings.
4. Review and analyze the simulation results in your simulator. Correct any functional errors in your design. If necessary, re-simulate the design to verify correct behavior.

1.7.3. Running Gate-Level Simulation (NativeLink Flow)

To run gate-level simulation with the NativeLink flow, follow these steps:

1. Prepare for simulation.
2. Set up the simulation environment. To generate only a functional (rather than timing) gate-level netlist, click **More EDA Netlist Writer Settings**, and turn on **Generate netlist for functional simulation only**.
3. To synthesize the design, follow one of these steps:
   - To generate a post-fit functional or post-fit timing netlist and then automatically simulate your design according to your NativeLink settings, Click **Processing > Start Compilation**. Skip to step 6.
   - To synthesize the design for post-synthesis functional simulation only, click **Processing > Start > Start Analysis and Synthesis**.
4. To generate the simulation netlist, click **Start EDA Netlist Writer**.
5. Click **Tools > Run Simulation Tool > Gate Level Simulation**.
6. Review and analyze the simulation results in your simulator. Correct any unexpected or incorrect conditions found in your design. Simulate the design again until you verify correct behavior.

1.8. 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:

- NativeLink-generated scripts—use NativeLink only to generate simulation script templates to develop your own custom scripts.
- 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 (Standard) simulation scripts—use the scripts generated for Intel FPGA IP cores and Platform Designer (Standard) systems as templates to create simulation scripts. If your design includes multiple IP cores or Platform Designer (Standard) 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.

Post-synthesis and post-fit gate-level simulations run significantly slower than RTL simulation. Intel FPGA recommends that you verify your design using RTL simulation for functionality and use the Timing Analyzer for timing. Timing simulation is not supported for Arria V, Cyclone V, Stratix V, and newer families.

1.9. 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

1.10. 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	• Removed *Scripting IP Simulation* and *Generating a Combined Simulation Script* topics. These features are supported only for Intel Arria 10 devices in Intel Quartus Prime Standard Edition. • Added link to *Scripting IP Simulation* in the *Introduction to Intel FPGA IP Cores*. • Updated supported simulator versions.
2018.05.07	18.0.0	• Updated list of supported simulation tools to include Cadence Parallel Simulator. • Added `xcelium_setup.sh` to list of simulation setup scripts. • Added *Sourcing 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
2017.11.06	17.1.0	• Added Simulation Library Compiler details to Quick Start Example
2017.05.08	17.0.0	• Updated simulator support table with latest version information.
2017.05.08	17.0.0	• Gate-level timing simulation limited to Arria II GX/GZ,Cyclone IV, MAX II, MAX V, and Stratix IV device families.
2016.10.31	16.1.0	• Updated simulator support table with latest version information. • Clarified license requirements for mixed language simulation with VHDL. • Gate-level timing simulation limited to Stratix IV and Cyclone IV devices.
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.
2016.05.02	16.0.0	• Noted limitations of NativeLink simulation. • 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*.
2015.11.02	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 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.