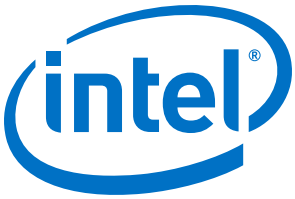

Intel® Quartus® Prime Standard Edition User Guide

Third-party Simulation

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

UG-20180 | 2018.09.24
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*	10.3	Windows
Aldec	Riviera-PRO*	2015.10	Windows, Linux
Cadence	Incisive Enterprise*	14.20	Linux
Mentor Graphics*	ModelSim* - Intel FPGA Edition	10.5b	Windows, Linux
Mentor Graphics	ModelSim PE	10.4d	Windows
Mentor Graphics	ModelSim SE	10.4d	Windows, Linux
Mentor Graphics	QuestaSim*	10.4d	Windows, Linux
Synopsys*	VCS* VCS MX	2014,12-SP1	Linux

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

continued...

Simulation Level	Description	Simulation Input
		• 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 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. • 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 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.
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 tclsh 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

- IP User Guide Documentation
- AN 351: Simulating Nios II Embedded Processors Designs
- 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 creates simulation files, including the functional simulation model, any testbench (or example design), and vendor-specific simulator setup scripts for each IP core. 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 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*>/mentor/msim_setup.tcl <*my_dir*>/synopsys/vcs/vcs_setup.sh <*my_dir*>/synopsys/vcsmx/vcsmx_setup.sh

continued...

File Type	Description	File Name
	Note: For Intel Arria 10 designs, you can use the Intel Quartus Prime software to automatically create a combined simulator setup script. Refer to *Scripting IP Simulation* in the *Introduction to Intel FPGA IP Cores* for more information.	
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

1.7. 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 6. 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)

continued...

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

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

1.7.2. 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. Simulating Intel FPGA Designs Revision History

This document has the following revision history.

Document Version	Intel Quartus Prime Version	Changes
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*.

Date	Version	Changes
2017.11.06	17.1.0	• Added Simulation Library Compiler details to Quick Start Example
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.

continued...

Date	Version	Changes
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.
2016.05.02	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*.
2015.05.04	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.

Related Information

Documentation Archive

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

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 theModelSim - 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:
   - Run NativeLink RTL simulation to compile required design files, simulation models, and run your simulator. Verify results in your simulator. If you complete this step you can ignore the remaining steps.
   - 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 simulatation 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. Disabling Timing Violation on Registers

In certain situations, you may want to ignore timing violations on registers and disable the "X" propagation that occurs. For example, this technique may be helpful to eliminate timing violations in internal synchronization registers in asynchronous clock-domain crossing. Intel Arria 10 devices do not support timing simulation. Intel Arria 10 devices do not support timing simulation.

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>
```

2.2.3. 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.

2.2.4. 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
```

2.2.5. Simulating Transport Delays

By default, the ModelSim and QuestaSim software filter out all pulses that are shorter than the propagation delay between primitives.

Turning on the **transport delay** options in the ModelSim and QuestaSim software prevents the simulator from filtering out these pulses. Intel Arria 10 devices do not support timing simulation.

Table 7. Transport Delay Simulation Options (ModelSim and QuestaSim)

Option	Description
`+transport_path_delays`	Use when simulation pulses are shorter than the delay in a gate-level primitive. You must include the `+pulse_e/number` and `+pulse_r/number` options.
`+transport_int_delays`	Use when simulation pulses are shorter than the interconnect delay between gate-level primitives. You must include the `+pulse_int_e/number` and `+pulse_int_r/number` options.

Note: The `+transport_path_delays` and `+transport_path_delays` options apply automatically during NativeLink gate-level timing simulation. For more information about either of these options, refer to the ModelSim-Intel FPGA Edition Command Reference installed with the ModelSim and QuestaSim software.

The following ModelSim and QuestaSim software command shows the command line syntax to perform a gate-level timing simulation with the device family library:

```
vsim -t 1ps -L stratixii -sdftyp /i1=filtref_vhd.sdo work.filtref_vhd_vec_tst
\
+transport_int_delays +transport_path_delays
```

2.2.6. 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 acces this option, click **Assignments ➤ Settings ➤ EDA Tool Settings ➤ More Settings**.

2.2.7. Generating Power Analysis Files

To generate a timing Value Change Dump File (`.vcd`) for power analysis, you must first generate a `<filename>_dump_all_vcd_nodes.tcl` script file in the Intel Quartus Prime software. You can then run the script from the ModelSim, QuestaSim, or ModelSim-Intel FPGA Edition software to generate a timing `.vcd` for use in the Intel Quartus Prime power analyzer.

To generate and use a `.vcd` for power analysis, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings**.
2. Under **EDA Tool Settings**, click **Simulation**.
3. Turn on **Generate Value Change Dump file script**, specify the type of output signals to include, and specify the top-level design instance name in your testbench. For example, if your top level design name is `Top`, and your testbench wrapper calls `Top` as instance `Top_inst`, specify the top level design instance name as `Top_inst`.
4. Click **Processing ➤ Start Compilation**. The Compiler creates the `<filename>_dump_all_vcd_nodes.tcl` file, the ModelSim simulation `<filename>_run_msim_gate_vhdl/verilog.do` file (including the **.vcd** and **.tcl** execution lines). Use the `<filename>_dump_all_vcd_nodes.tcl` to dump all of the signals that you expect for input back into the Power Analysis.
5. Elaborate and compile the design in your simulator.
6. Source the `<filename>_run_msim_gate_vhdl/verilog.do` file, and then run the simulation. The simulator opens the `.vcd` file that contains the dumped signal file transition information.
7. Stop the simulation if your testbench does not have a break point.

2.2.8. 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 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.

2.2.9. 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

Related Information

ModelSim-Intel FPGA Edition Software Web Page

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

Date	Version	Changes
2017.11.06	17.1.0	• Changed title to ModelSim - Intel FPGA Edition, ModelSim, and QuestaSim Support* • Stated no support for Intel Arria 10 timing simulation in Simulating Transport Delays and Disabling Timing Violations on Registers topics. • Added Simulation Library Compiler details and another step to Quick Start Example
2016.10.31	16.1.0	• Implemented Intel rebranding. • Corrected load design syntax error.
2016.05.02	16.0.0	• Noted limitations of NativeLink simulation. • 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.

Related Information

Documentation Archive

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

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:
   - Run NativeLink RTL simulation to compile required design files, simulation models, and run your simulator. Verify results in your simulator. If you complete this step you can ignore the remaining steps.
   - 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 simulatation 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.

3.2.1. Simulating Transport Delays

By default, the VCS and VCS MX software filter out all pulses that are shorter than the propagation delay between primitives. Turning on the **transport delay** options in the VCS and VCS MX software prevents the simulator from filtering out these pulses. Intel Arria 10 devices do not support timing simulation.

Table 8. Transport Delay Simulation Options (VCS and VCS MX)

Option	Description
`+transport_path_delays`	Use when simulation pulses are shorter than the delay in a gate-level primitive. You must include the `+pulse_e/number` and `+pulse_r/number` options.
`+transport_int_delays`	Use when simulation pulses are shorter than the interconnect delay between gate-level primitives. You must include the `+pulse_int_e/number` and `+pulse_int_r/number` options.

Note: The `+transport_path_delays` and `+transport_path_delays` options apply automatically during NativeLink gate-level timing simulation.

The following VCS and VCS MX software command runs a post-synthesis simulation:

```
vcs -R <testbench>.v <gate-level netlist>.v -v <Intel FPGA device family \
library>.v +transport_int_delays +pulse_int_e/0 +pulse_int_r/0 \
+transport_path_delays +pulse_e/0 +pulse_r/0
```

3.2.2. Disabling Timing Violation on Registers

In certain situations, you may want to ignore timing violations on registers and disable the "X" propagation that occurs. For example, this technique may be helpful to eliminate timing violations in internal synchronization registers in asynchronous clock-domain crossing. Intel Arria 10 devices do not support timing simulation. Intel Arria 10 devices do not support timing simulation.

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>
```

3.2.3. Generating Power Analysis Files

You can generate a Verilog Value Change Dump File (.vcd) for power analysis in the Intel Quartus Prime software, and then run the **.vcd** from the VCS software. Use this **.vcd** for power analysis in the Intel Quartus Prime power analyzer.

To generate and use a **.vcd** for power analysis, follow these steps:

1. In the Intel Quartus Prime software, click **Assignments ➤ Settings**.
2. Under **EDA Tool Settings**, click **Simulation**.
3. Turn on **Generate Value Change Dump file script**, specify the type of output signals to include, and specify the top-level design instance name in your testbench.
4. Click **Processing ➤ Start Compilation**.
5. Use the following command to include the script in your testbench where the design under test (DUT) is instantiated:
   `include <revision_name>_dump_all_vcd_nodes.v`

   Note: Include the script within the testbench module block. If you include the script outside of the testbench module block, syntax errors occur during compilation.
6. Run the simulation with the VCS command. Exit the VCS software when the simulation is finished and the *<revision_name>***.vcd** file is generated in the simulation directory.

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

Date	Version	Changes
2017.11.06	17.1.0	• Stated no support for Intel Arria 10 timing simulation in Simulating Transport Delays and Disabling Timing Violations on Registers topics. • Added Simulation Library Compiler details and another step to Quick Start Example
2016.05.02	16.0.0	• Noted limitations of NativeLink simulation.
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. Cadence Simulator Support

You can include your supported EDA simulator in the Intel Quartus Primedesign flow. This chapter provides specific guidelines for simulation of Intel Quartus Prime designs with the Cadence Incisive Enterprise (IES) software.

4.1. 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:
   - Run NativeLink RTL simulation to compile required design files, simulation models, and run your simulator. Verify results in your simulator. If you complete this step you can ignore the remaining steps.
   - 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 simulat libraries.

   Use the compiled simulation model libraries during simulatation 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>
   ```

4.2. Using GUI or Command-Line Interfaces

Intel FPGA supports both the IES GUI and command-line simulator interfaces

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

Table 9. 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.

4.3. 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.

4.3.1. Elaborating Your Design

The simulator automatically reads the `.sdo` file during elaboration of the Intel Quartus Prime-generated Verilog HDL or SystemVerilog HDL netlist file. The ncelab command recognizes the embedded system task `$sdf_annotate` and automatically compiles and annotates the `.sdo` file by running `ncsdfc` automatically.

VHDL netlist files do not contain system task calls to locate your `.sdf` file; therefore, you must compile the standard `.sdo` file manually. Locate the `.sdo` file in the same directory where you run elaboration or simulation. Otherwise, the `$sdf_annotate` task cannot reference the `.sdo` file correctly. If you are starting an elaboration or simulation from a different directory, you can either comment out the `$sdf_annotate` and annotate the `.sdo` file with the GUI, or add the full path of the `.sdo` file.

Note: If you use NC-Sim for post-fit VHDL functional simulation of a Stratix V design that includes RAM, an elaboration error might occur if the component declaration parameters are not in the same order as the architecture parameters. Use the `-namemap_mixgen` option with the `ncelab` command to match the component declaration parameter and architecture parameter names.

4.3.2. Back-Annotating Simulation Timing Data (VHDL Only)

You can back annotate timing information in a Standard Delay Output File (.sdo) for VHDL simulators. To back annotate the .sdo timing data at the command line, follow these steps:

1. To compile the **.sdo** with the ncsdfc program, type the following command at the command prompt. The ncsdfc program generates an <*output name*>**.sdf.X** compiled **.sdo** file

   ```
   ncsdfc <project name>_vhd.sdo –output <output name>
   ```

   Note: If you do not specify an output name, ncsdfc uses <*project name*>**.sdo.X**
2. Specify the compiled **.sdf** file for the project by adding the following command to an ASCII SDF command file for the project:

   ```
   COMPILED_SDF_FILE = "<project name>.sdf.X" SCOPE = <instance path>
   ```
3. After compiling the **.sdf** file, type the following command to elaborate the design:

   ```
   ncelab worklib.<project name>:entity –SDF_CMD_FILE <SDF Command File>
   ```

Example 5. Example SDF Command File

```
// SDF command file sdf_file
COMPILED_SDF_FILE = "lpm_ram_dp_test_vhd.sdo.X",
SCOPE = :tb,
MTM_CONTROL = "TYPICAL",
SCALE_FACTORS = "1.0:1.0:1.0",
SCALE_TYPE = "FROM_MTM";
```

4.3.3. Disabling Timing Violation on Registers

In certain situations, you may want to ignore timing violations on registers and disable the "X" propagation that occurs. For example, this technique may be helpful to eliminate timing violations in internal synchronization registers in asynchronous clock-domain crossing. Intel Arria 10 devices do not support timing simulation.

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>
```

4.3.4. 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 10. 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

4.3.5. 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.

4.4. 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 (Standard) 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 6. 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
```

4.5. Cadence Simulator Support Revision History

Table 11. Document Revision History

Date	Version	Changes
2017.11.06	17.1.0	• Stated no support for Intel Arria 10 timing simulation in Simulating Transport Delays and Disabling Timing Violations on Registers topics. • Added Simulation Library Compiler details and another step to Quick Start Example
2016.05.02	16.0.0	• Noted limitations of NativeLink simulation.
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.

5. 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.

5.1. 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:
   - Run NativeLink RTL simulation to compile required design files, simulation models, and run your simulator. Verify results in your simulator. If you complete this step you can ignore the remaining steps.
   - 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 simulatation 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.

5.2. Aldec Active-HDL and Riviera-PRO Guidelines

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

5.2.1. 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.

5.2.2. Simulating Transport Delays

By default, the Active-HDL or Riviera-PRO software filters out all pulses that are shorter than the propagation delay between primitives. Turning on the **transport delay** options in the in the Active-HDL or Riviera-PRO software prevents the simulator from filtering out these pulses. Intel Arria 10 devices do not support timing simulation.

Table 12. Transport Delay Simulation Options

Option	Description
`+transport_path_delays`	Use when simulation pulses are shorter than the delay in a gate-level primitive. You must include the `+pulse_e/number` and `+pulse_r/number` options.
`+transport_int_delays`	Use when simulation pulses are shorter than the interconnect delay between gate-level primitives. You must include the `+pulse_int_e/number` and `+pulse_int_r/number` options.

Note: The `+transport_path_delays` and `+transport_path_delays` options apply automatically during NativeLink gate-level timing simulation.

To perform a gate-level timing simulation with the device family library, type the Active-HDL command:

```
vsim -t 1ps -L stratixii -sdftyp /i1=filtref_vhd.sdo \
work.filtref_vhd_vec_tst +transport_int_delays +transport_path_delays
```

5.2.3. Disabling Timing Violation on Registers

In certain situations, you may want to ignore timing violations on registers and disable the "X" propagation that occurs. For example, this technique may be helpful to eliminate timing violations in internal synchronization registers in asynchronous clock-domain crossing. Intel Arria 10 devices do not support timing simulation.

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>
```

5.3. 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

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

Date	Version	Changes
2017.11.06	17.1.0	• Stated no support for Intel Arria 10 timing simulation in Simulating Transport Delays and Disabling Timing Violations on Registers topics. • Added Simulation Library Compiler details to Quick Start Example
2016.05.02	16.0.0	• Noted limitations of NativeLink simulation.
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.

A. Intel Quartus Prime Standard Edition User Guides

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

Related Information

- Intel Quartus Prime Standard Edition User Guide: Getting Started
  Introduces the basic features, files, and design flow of the Intel Quartus Prime Standard Edition software, including managing Intel Quartus Prime Standard Edition projects and IP, initial design planning considerations, and project migration from previous software versions.
- Intel Quartus Prime Standard Edition User Guide: Platform Designer
  Describes creating and optimizing systems using Platform Designer (Standard), a system integration tool that simplifies integrating customized IP cores in your project. Platform Designer (Standard) automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Intel Quartus Prime Standard Edition User Guide: Design Recommendations
  Describes best design practices for designing FPGAs with the Intel Quartus Prime Standard Edition software. HDL coding styles and synchronous design practices can significantly impact design performance. Following recommended HDL coding styles ensures that Intel Quartus Prime Standard Edition synthesis optimally implements your design in hardware.
- Intel Quartus Prime Standard Edition User Guide: Design Compilation
  Describes set up, running, and optimization for all stages of the Intel Quartus Prime Standard Edition Compiler. The Compiler synthesizes, places, and routes your design before generating a device programming file.
- Intel Quartus Prime Standard Edition User Guide: Design Optimization
  Describes Intel Quartus Prime Standard 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, and optimization of device resource usage.
- Intel Quartus Prime Standard Edition User Guide: Programmer
  Describes operation of the Intel Quartus Prime Standard 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 Standard 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 Standard 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 Standard 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 Standard Edition User Guide: Debug Tools
  Describes a portfolio of Intel Quartus Prime Standard 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, Transceiver Toolkit, In-System Memory Content Editor, and In-System Sources and Probes Editor.
- Intel Quartus Prime Standard Edition User Guide: Timing Analyzer
  Explains basic static timing analysis principals and use of the Intel Quartus Prime Standard 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 Standard Edition User Guide: Power Analysis and Optimization
  Describes the Intel Quartus Prime Standard 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 Standard 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 Pin Planner to visualize, modify, and validate all I/O assignments in a graphical representation of the target device.
- Intel Quartus Prime Standard 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 Standard Edition User Guide: Scripting
  Describes use of Tcl and command line scripts to control the Intel Quartus Prime Standard Edition software and to perform a wide range of functions, such as managing projects, specifying constraints, running compilation or timing analysis, or generating reports.