

This chapter describes how to use the Synopsys VCS and VCS MX software to simulate designs that target Altera® FPGAs. This chapter provides instructions about how to perform functional simulations, post-synthesis simulations, and gate-level timing simulations. This chapter also describes the location of the simulation libraries and how to automate simulations.

This chapter includes the following topics:

- “Software Requirements”
- “Using the VCS or VCS MX Software in the Quartus II Design Flow”
- “Common VCS and VCS MX Software Compiler Options” on page 3–8
- “Using DVE” on page 3–8
- “Debugging Support Command-Line Interface” on page 3–9
- “Simulating Designs that Include Transceivers” on page 3–9
- “Transport Delays” on page 3–13
- “Using NativeLink with the VCS or VCS MX Software” on page 3–13
- “Generating a Timing .vcd File for the PowerPlay Power Analyzer” on page 3–13
- “Viewing a Waveform from a .vpd or .vcd File” on page 3–14
- “Scripting Support” on page 3–15

Software Requirements

To simulate your design with the Synopsys VCS or VCS MX software, you must first set up the Altera libraries. These libraries are installed with the Quartus® II software.

- For more information about installing the software and the directories created during the Quartus II software installation, refer to the *Altera Software Installation and Licensing* manual.

Using the VCS or VCS MX Software in the Quartus II Design Flow

You can perform the following types of simulations with the VCS and VCS MX software:

- Functional Simulation
- Post-Synthesis Simulation
- Gate-Level Timing Simulation

- Refer to the “PLD Design Flow” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook* for the Quartus II software design flow.

Compiling Libraries Using the EDA Simulation Library Compiler

The EDA Simulation Library Compiler compiles Verilog HDL, SystemVerilog HDL, and VHDL simulation libraries for all Altera devices and supported third-party simulators. You can compile all libraries required by functional and gate-level timing simulations.

If the compilation targets the VCS simulator, the VCS options file `simlib_comp.vcs` is generated after compilation.

- For more information, refer to the “EDA Simulation Library Compiler” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Functional Simulation

A functional simulation verifies the functionality of the design before synthesis, placement, and routing. A functional simulation is independent of any Altera FPGA architecture implementation. After the HDL designs are verified to be functionally correct, the next step is to synthesize the design and use the Quartus II software to place-and-route the design in an Altera device.

To perform a functional simulation of an Altera FPGA design that uses Altera intellectual property (IP) megafunctions or a library of parameterized modules (LPM) functions, you must include certain libraries during the compilation.

- For a list of the functional simulation library files in the Quartus II directory, refer to *Altera Functional Simulation Libraries* in Quartus II Help.

Functional Simulation for Verilog HDL and SystemVerilog HDL Designs

Use the following VCS commands to perform a functional simulation for Verilog HDL and SystemVerilog HDL designs with functional simulation libraries:

For Verilog HDL, type the following command:

```
vcs -R <testbench>.v <design name>.v -v <altera_library1>.v -v \
<altera_library2>.v ↵
```

For SystemVerilog HDL, type the following command:

```
vcs -R <testbench>.v <design name>.v -v <altera_library1>.v -v \
<altera_library2>.v +systemverilogext+.sv+.svo ↵
```

If you have already generated the option file (`simlib_comp.vcs`) from “[Compiling Libraries Using the EDA Simulation Library Compiler](#)”, type the following command:

```
vcs -file simlib_comp.vcs ↵
```

Be sure to include the design files and testbench files in `simlib_comp.vcs`.


Alternatively, you can use the following commands to perform functional simulation for Verilog HDL and SystemVerilog HDL designs.

- To create library directories, type the following commands:

```
mkdir <Directory_to_store_compiled_altera_library1> ←  
mkdir <Directory_to_store_compiled_altera_library2> ←
```

2. To create the work directory, type the following command:

```
mkdir <Directory_to_store_compiled_design_and_testbench_files> ←
```


 Before performing the following step, make sure the mapped file **synopsys_sim.setup** was created.

3. To compile libraries, type the following commands:

```
vlogan -work <altera_library1_name> <altera_library1>.v ←  
vlogan -work <altera_library2_name> <altera_library2>.v ←
```

4. To compile the design and testbench, type the following command:

```
vlogan -work <work_library_name> <design>.v <testbench>.v ←
```

 For SystemVerilog, type the following commands:

```
vlogan -sverilog <design>.sv ←  
vlogan -sverilog <design>.svo ←
```

5. To elaborate your design, type the following command:


```
vcs -debug_all <work_library_name>.<testbench_top-level_module> ←
```


6. To run the simulation, type the following command

```
simv -gui ←
```

The **synopsys_sim.setup** file contains the following mapping commands to map the libraries:

```
<altera_library1_name> : <Directory_to_store_compiled_altera_library1>  
<altera_library2_name> : <Directory_to_store_compiled_altera_library2>  
<work_library_name> : <Directory_to_store_compiled_design  
and_testbench_files>
```

 The **altera_mf.v** model files should be compiled into the **altera_mf_ver** library. The **220model.v** model files should be compiled into the **lpm_ver** library.

-  If you are compiling Stratix® V libraries, refer to [Guidelines for Compiling Stratix V Libraries](#) in Quartus II Help.

Functional Simulation for VHDL Designs


For VHDL designs, you need to use VCS MX software to perform all three types of simulations. Use the following commands to perform a functional simulation for VHDL designs with the libraries listed in [Altera Functional Simulation Libraries](#) in Quartus II Help.

1. To create library directories, type the following commands:

```
mkdir <Directory_to_store_compiled_altera_library1> ←  
mkdir <Directory_to_store_compiled_altera_library2> ←
```

2. To create the work directory, type the following command:

```
mkdir <Directory_to_store_compiled_design_and_testbench_files> ←
```

 Before performing the following step, make sure the mapped file `synopsys_sim.setup` was created.

3. To compile libraries, type the following commands:

```
vhdlan -work <altera_library1_name> <altera_library1>.vhd ←
vhdlan -work <altera_library2_name> <altera_library2>.vhd ←
```

4. To compile the design and testbench, type the following command:

```
vhdlan -work <work_library_name> <design>.vhd <testbench>.vhd ←
```

5. To elaborate your design, type the following command:


```
vcs -debug_all <work_library_name>.<testbench_top_level_module> ←
```


6. To run the simulation, type the following command:


```
simv -gui ←
```

The `synopsys_sim.setup` file contains the following mapping commands to map the libraries:

```
<altera_library1_name> : <Directory_to_store_compiled_altera_library1>
<altera_library2_name> : <Directory_to_store_compiled_altera_library2>
<work_library_name> : <Directory_to_store_compiled_design \
and_testbench_files>
```


 The `altera_mf.v` model files should be compiled into the `altera_mf_ver` library. The `220model.v` model files should be compiled into the `lpm_ver` library.

 If you have generated the Altera libraries with the EDA Simulation Library Compiler, ignore steps 1 and 3.


 If you are compiling Stratix V libraries, refer to [Guidelines for Compiling Stratix V Libraries](#) in Quartus II Help.

Post-Synthesis Simulation

A post-synthesis simulation verifies the functionality of a design after synthesis is performed. You can create a post-synthesis netlist file in the Quartus II software and use this netlist file to perform a post-synthesis simulation in the VCS or VCS MX software. When the post-synthesis version of the design is verified, the next step is to place-and-route the design in the target architecture with the Quartus II software.

 For information about how to generate a post-synthesis simulation netlist file, refer to [Generating Simulation Netlist Files](#) in Quartus II Help.

Post-Synthesis Simulation for Verilog HDL and SystemVerilog HDL Designs

 You cannot perform post-synthesis or post-fit simulation if you are targeting the Stratix V device family.

To perform a post-synthesis simulation with the appropriate device family library, type the following VCS command:

```
vcs -R <testbench> <post-synthesis netlist> -v <altera_library1> \
+systemverilogext+.sv+.svo ←
```

- ❓ For more information about *Altera Post-Fit Libraries*, refer to the Quartus II Help.

If you have already generated the option file (**simlib_comp.vcs**) as described in “[Compiling Libraries Using the EDA Simulation Library Compiler](#)” on page 3-2, modify the **simlib_comp.vcs** file to add the testbench and post-synthesis netlist file, and then type the following command:

```
vcs -file simlib_comp.vcs ↵
```

Be sure to include the post-synthesis netlist file and testbench files in **simlib_comp.vcs**.

Alternatively, you can use the following commands to perform post-synthesis simulation for Verilog HDL and SystemVerilog HDL designs:

1. To create library directories, type the following commands:

```
mkdir <Directory_to_store_compiled_altera_library1> ↵  
mkdir <Directory_to_store_compiled_altera_library2> ↵
```

2. To create the work directory, type the following command:

```
mkdir <Directory_to_store_compiled_design_and_testbench_files> ↵
```

👉 Before performing the following step, make sure the mapped file **synopsys_sim.setup** was created.

3. To compile libraries, type the following commands:

```
vlogan -work <altera_library1_name> <altera_library1>.v ↵  
vlogan -work <altera_library2_name> <altera_library2>.v ↵
```

4. To compile the design and testbench, type the following command:

```
vlogan -work <work_library_name> <design>.v <testbench>.v ↵
```

For Verilog HDL, include the following command:

```
vlogan -work <work_library_name> <post-synthesis_netlist>.vo \  
<testbench>.v ↵
```

For SystemVerilog HDL, include the following command:

```
vlogan -sverilog -work <work_library_name> <post-synthesis \  
_netlist>.svo <testbench>.v ↵
```

5. To elaborate your design, type the following command:

```
vcs -debug_all <work_library_name>.<testbench_top-level_module> ↵
```

6. To run the simulation, type the following command

```
simv -gui ↵
```

The **synopsys_sim.setup** file contains the following commands to map the libraries:

```
<altera_library1_name> : <Directory_to_store_compiled_altera_library1>  
<altera_library2_name> : <Directory_to_store_compiled_altera_library2>  
<work_library_name> : <Directory_to_store_compiled_post-synthesis \  
netlist and_testbench_files>
```

Post-Synthesis Simulation for VHDL Designs

Use the following VCS MX commands to perform a post-synthesis simulation with the appropriate device family library:

1. To create library directories, type the following commands:

```
mkdir <Directory_to_store_compiled_altera_library1> ←
mkdir <Directory_to_store_compiled_altera_library2> ←
```

2. To create the work directory, type the following command:

```
mkdir <Directory_to_store_compiled_post-synthesis_netlist \
and_testbench_files> ←
```



Before performing the following step, make sure the mapped file **synopsys_sim.setup** was created.

3. To compile libraries, type the following commands:

```
vhdlan -work <altera_library1_name> <altera_library1>.vhd ←
vhdlan -work <altera_library2_name> <altera_library2>.vhd ←
```

4. To compile the design and testbench, type the following command:

```
vhdlan -work <work_library_name> <post-synthesis_netlist>.vho \
<testbench>.vhd ←
```

5. To elaborate your design, type the following command:

```
vcs -debug_all <work_library_name>.<testbench_top_level_module> ←
```

6. To run the simulation, type the following command:

```
simv -gui ←
```

The **synopsys_sim.setup** file contains the following commands to map the libraries:

```
<altera_library1_name> : <Directory_to_store_compiled_altera_library1>
<altera_library2_name> : <Directory_to_store_compiled_altera_library2>
<work_library_name> : <Directory_to_store_compiled_post_synthesis \
netlist and_testbench_files>
```

- ② For more information about *Altera Post-Fit Libraries*, refer to the Quartus II Help.




If you have generated the Altera libraries with the EDA Simulation Library Compiler, ignore steps 1 and 3.

Gate-Level Timing Simulation

A gate-level timing simulation verifies the functionality and timing of the design after place-and-route. You can create a gate-level simulation netlist file in the Quartus II software and use this netlist file to perform a gate-level timing simulation in the VCS or VCS MX software.

- ② For information about how to generate a gate-level simulation netlist file, refer to *Generating Simulation Netlist Files* in Quartus II Help.
- ② For a list of the gate-level timing simulation library files in the Quartus II directory, refer to *Altera Post-Fit Libraries* in Quartus II Help.

 You cannot perform post-synthesis or post-fit simulation if you are targeting the Stratix V device family.

Gate-Level Timing Simulation for Verilog HDL and SystemVerilog HDL Designs

For gate-level timing simulation, follow the steps in “[Post-Synthesis Simulation for Verilog HDL and SystemVerilog HDL Designs](#)” on page 3-4.


You do not have to specify the Standard Delay Output File (.sdo) file because it is already specified in the Verilog Output File (.vo) file or SystemVerilog Output File (.svo). However, the .sdo must be in the same directory as the simulator executable file `simv` generated by VCS.

Gate-Level Timing Simulation for VHDL Designs

For gate-level timing simulation, follow the steps in “[Post-Synthesis Simulation for VHDL Designs](#)” on page 3-6.

For VHDL, the *.sdo file must be specified in the `simv` command as follows:

```
simv -xlrnm -gui -sdf typ:<testbench module name>/<design instance name>.sdo ←
```

 Adding the `-xlrnm` switch avoids errors that occur when the timing arcs in SDO do not match Altera VHDL simulation models as per the IEEE VITAL ASIC standard. However, adding this switch reduces timing accuracy, as it may cause some SDO delays to be ignored. Therefore, generate the Verilog HDL or SystemVerilog HDL simulation output netlist (.vo or .svo) if you want to perform gate-level timing simulation.

Disabling Timing Violation on Registers

In certain situations, the timing violations can be ignored and you can disable the “X” propagation that happens when there are timing violations on registers (for example, timing violations that occur in internal synchronization registers used for asynchronous clock-domain crossing).

By default, the `x_on_violation_option logic` option that applies to all registers in the design is **On**, which means a register outputs “X” whenever a timing violation occurs. To disable “X” propagation due to a timing violation on certain registers, set the `x_on_violation_option logic` option to **Off** for those registers. The following command is an example from the Quartus II Settings File (.qsf):

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

Performing Functional Simulation Using the Post-Synthesis Netlist

You can perform a functional simulation with the post-synthesis netlist file instead of a gate-level netlist file, and you can generate an .sdo file without running the Fitter. In this case, the .sdo file includes all timing values for the device cells only. Interconnect delays are not included because fitting (placement and routing) has not been performed.

To generate the post-synthesis netlist file and the `.sdo` file, type the following commands at a command prompt:

```
quartus_map <project name> -c <revision name> ↵
quartus_eda <project name> -c <revision name> --simulation \
--functional --tool= <third-party EDA tool> --format=<HDL language> ↵
```

For more information about the `-format` and `-tool` options, type the following command:


```
quartus_eda --help=<options> ↵
```

Common VCS and VCS MX Software Compiler Options

Table 3–1 lists VCS and VCS MX software options that can help you simulate your design.

Table 3–1. VCS Software Compiler Options

Library	Description
<code>-R</code>	Runs the executable file immediately.
<code>-v <library filename></code>	Specifies a Verilog HDL library file (for example, <code>220model.v</code> or <code>altera_mf.v</code>). The VCS software looks in this file for module definitions that are found in the source code. This option is available for VCS only.
<code>-y <library directory></code>	Specifies a Verilog HDL library directory. The VCS software looks for library files in this folder that contain module definitions that are instantiated in the source code. This option is available for VCS only.
<code>+compsdf</code>	Indicates that the VCS software compiler includes the back-annotated Standard Delay File (<code>.sdf</code>) file in the compilation.
<code>+cli</code>	The VCS software enters Command-Line Interface (CLI) mode upon successful compilation completion.
<code>+race</code>	Specifies that the VCS software generate a report that indicates all of the race conditions in the design. The default report name is <code>race.out</code> .
<code>-P</code>	Compiles user-defined Programming Language Interface (PLI) table files.
<code>-q</code>	Indicates the VCS software runs in quiet mode. All messages are suppressed.

 For more information about VCS software options, refer to the [VCS User Guide](#) installed with the VCS software.

Using DVE


Design Viewpoint Editor (DVE) is the graphical debugging system for the VCS and VCS MX software. This tool is included with the VCS MX software. It can be run by adding the `-gui` option when running a simulation.

To run a simulation in DVE, type the following VCS or VCS MX command:

```
simv -gui ↵
```

However, to open the GUI with these commands, you must enable the Unified Command Line Interface (UCLI) and DVE when performing elaboration. To enable UCLI and DVE, type the following command:

```
vcs -debug_all ↵
```

-  For more information about DVE, refer to the *DVE User Guide* installed with the VCS MX software.


Debugging Support Command-Line Interface

The VCS software UCLI is an interactive, non-graphical debugger that can be used to halt simulations at user-defined break points, force registers with values, and display register values.

Enable the debugger by including the `+ucli` run-time option. To use the VCS software UCLI to debug your Altera FPGA design, type the following command:

```
vcs -R <testbench>.v <design name>.vo -v <path to Quartus II \
installation directory> \eda\sim_lib\<device family>_atoms.v +compsdf +ucli
\ +systemverilogext+.sv+.svo ↵
```

The `+ucli` command takes an optional number argument that specifies the level of debugging capability. As the optional debugging capability is increased, there is an increase in simulation time.



-  For more information about the `+ucli` options, refer to the *VCS User Guide* installed with the VCS software.

For the design examples to run gate-level timing simulations in VHDL or Verilog HDL language, refer to the *Synopsys VCS Simulation Design Example* page on the Altera website.

Simulating Designs that Include Transceivers


If your design includes Arria[®], Arria II, Cyclone[®] IV, HardCopy[®] IV, Stratix[®], Stratix II, Stratix IV, or Stratix V transceivers, you must compile additional library files to perform functional or gate-level timing simulations.

For high-speed simulation, you must select **ps** in the **Resolution** list for your simulator resolutions (**Design** tab of the **Start Simulation** dialog box). If you choose slower than **ps**, the high-speed simulation might fail.

-  If your design contains PCI Express hard IP, refer to the “Simulate the Design” section in the *IP Compiler for PCI Express User Guide*.
-  If you are compiling Stratix V libraries, refer to *Guidelines for Compiling Stratix V Libraries* in Quartus II Help.

Functional Simulation for Stratix GX Devices

To perform a functional simulation of your design that instantiates the ALTGXB megafunction, enabling the gigabit transceiver block gigabit transceiver block on Stratix GX devices, compile the `stratixgx_mf` model file into the `altgxb` library.

-  The `stratixgx_mf` model file references the `lpm` and `sgate` libraries. You must create these libraries to perform a simulation.

To compile the libraries necessary for functional simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix GX device, at the VCS command prompt, type the following command:

```
vcs -R <testbench>.v <design files>.v -v stratixgx_mf.v -v sgate.v \
-v 220model.v -v altera_mf.v +systemverilogext+.sv+.svo ←
```

Gate-Level Timing Simulation for Stratix GX Devices

Perform a gate-level timing simulation of your design that includes a Stratix GX transceiver by compiling the **stratixgx_atoms** and **stratixgx_hssi_atoms** model files into the **stratixgx** and **stratixgx_gxb** libraries, respectively.



The **stratixgx_hssi_atoms** model file references the **lpm** and **sgate** libraries. You must create these libraries to perform a simulation.

To compile the libraries necessary for timing simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix GX device, at the VCS command prompt, type the following command:

```
vcs -R <testbench>.v <gate-level netlist>.vo -v stratixgx_atoms.v -v \
stratixgx_hssi_atoms.v -v sgate.v -v 220model.v -v altera_mf.v \
+transport_int_delays +pulse_int_e/0 +pulse_int_r/0 \
+transport_path_delays +pulse_e/0 +pulse_r/0 \
+systemverilogext+.sv+.svo ←
```

Functional Simulation for Stratix II GX Devices

Functional simulation for Stratix II GX devices is similar to functional simulation for Arria GX devices. To simulate the transceiver in Arria GX devices, you only have to replace the **stratixiigx_hssi** model file with the **arriagx_hssi** model file.

To perform a functional simulation of your design that instantiates the ALT2GXB megafunction, enabling the gigabit transceiver block on Stratix II GX devices, compile the **stratixiigx_hssi** model file into the **stratixiigx_hssi** library.



The **stratixiigx_hssi_atoms** model file references the **lpm** and **sgate** libraries. You must create these libraries to perform a simulation.

Generate a functional simulation netlist file by turning on **Generate Simulation Model in the Simulation Library** in the ALT2GXB MegaWizard™ Plug-In Manager. The **<alt2gxb entity name>.vho** file or **<alt2gxb module name>.vo** or **.svo** file is generated in the current project directory.



The ALT2GXB functional simulation library file generated by the Quartus II software references **stratixiigx_hssi wysiwyg atoms**.

To compile the libraries necessary for functional simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix II GX device, type the following command at the VCS command prompt:

```
vcs -R <testbench>.v <alt2gxb simulation netlist>.vo -v \
stratixgx_hssi_atoms.v -v sgate.v -v 220model.v -v altera_mf.v \
+systemverilogext+.sv+.svo ←
```

Gate-Level Timing Simulation for Stratix II GX Devices

Gate-level timing simulation for Stratix II GX devices is similar to gate-level timing simulation for Arria GX devices. You only have to replace the `stratixiigx_hssi` model file with the `arriagx_hssi` model file.

To perform a gate-level timing simulation of your design that includes a Stratix II GX transceiver, compile `stratixiigx_atoms` and `stratixiigx_hssi_atoms` into the `stratixiigx` and `stratixiigx_hssi` libraries, respectively.



The `stratixiigx_hssi_atoms` model file references the `lpm` and `sgate` libraries. You must create these libraries to perform a simulation.

To compile the libraries necessary for timing simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix II GX device, type the following command at the VCS command prompt:

```
vcs -R <testbench>.v <gate-level netlist>.vo -v stratixiigx_atoms.v -v \
stratixiigx_hssi_atoms.v -v sgate.v -v 220model.v -v altera_mf.v \
+transport_int_delays +pulse_int_e/0 +pulse_int_r/0 \
+transport_path_delays +pulse_e/0 +pulse_r/0 +systemverilogext+.sv+.svo ←
```

Functional Simulation for Stratix IV GX Devices

Functional simulation for Stratix IV devices is similar to functional simulation for Arria II, Cyclone IV, and HardCopy IV devices. You only have to replace the `stratixiv_hssi` model file with the `arriaii_hssi`, `cycloneiv_hssi`, and `hardcopyiv_hssi` model files, respectively.

To perform a functional simulation of your design that instantiates the ALTGX megafunction, enabling the gigabit transceiver block on Stratix IV devices, compile the `stratixiv_hssi` model file into the `stratixiv_hssi` library.

The `stratixiv_hssi_atoms` model file references the `lpm` and `sgate` libraries. You must create these libraries to perform a simulation.

To compile the libraries necessary for functional simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix IV device, type the following command at the VCS command prompt:

```
vcs -R <testbench>.v <altgx>.v -v stratixiv_hssi_atoms.v -v sgate.v \
-v 220model.v -v altera_mf.v +systemverilogext+.sv+.svo ←
```

Gate-Level Timing Simulation for Stratix IV GX Devices

Gate-level timing simulation for Stratix IV devices is similar to gate-level timing simulation for Arria II, Cyclone IV, and HardCopy IV devices. You only have to replace the `stratixiv_hssi` model file with the `arriaii_hssi`, `cycloneiv_hssi`, and `hardcopyiv_hssi` model files, respectively.

To perform a gate-level timing simulation of your design that includes a Stratix IV transceiver, compile `stratixiv_atoms` and `stratixiv_hssi_atoms` into the `stratixiv` and `stratixiv_hssi` libraries, respectively.

To perform a gate-level timing simulation of your design that includes a Stratix IV transceiver, compile `stratixiv_atoms` and `stratixiv_hssi_atoms` into the `stratixiv` and `stratixiv_hssi` libraries, respectively.

The `stratixv_hssi_atoms` model file references the `lpm` and `sgate` libraries. You must create these libraries to perform a simulation.

To compile the libraries necessary for timing simulation of a Verilog HDL and SystemVerilog HDL design targeting a Stratix IV device, type the following command at the VCS command prompt:

```
vcs -R <testbench>.v <gate-level netlist>.vo -v stratixiv_atoms.v \
-v stratixiv_hssi_atoms.v -v sgate.v -v 220model.v -v altera_mf.v \
+transport_int_delays +pulse_int_e/0 +pulse_int_r/0 \
+transport_path_delays +pulse_e/0 +pulse_r/0 \
+systemverilogext+.sv+.svo ←
```

Functional Simulation for Stratix V GX Devices

Functional simulation for Stratix V devices is similar to functional simulation for Arria II, Cyclone IV, HardCopy IV, and Stratix IV devices. You only have to replace the `stratixv_hssi` model file with the `arriaii_hssi`, `cycloneiv_hssi`, `hardcopyiv_hssi`, and `stratixiv_hssi` model files, respectively.

The `stratixv_hssi_atoms` model file references the `lpm` and `sgate` libraries. You must compile these libraries to perform a simulation.



The transceiver module from the MegaWizard Plug-In Manager is created in **Interfaces/Transceiver PHY**. Select **Custom PHY**.

To compile the libraries necessary for functional simulation of a Verilog HDL or VHDL design targeting a Stratix V device, type the following commands at the VCS command prompt:

```
mkdir -p ./stratixv ←
mkdir -p ./stratixv_pcie_hip ←
mkdir -p ./stratixv_hssi ←
mkdir -p ./work ←

vlogan +v2k -work stratixv \
$QUARTUS_ROOTDIR/eda/sim_lib/synopsys/stratixv_atoms_ncrypt.v ←

vlogan +v2k -work stratixv_hssi \
$QUARTUS_ROOTDIR/eda/sim_lib/synopsys/stratixv_hssi_atoms_ncrypt.v ←

vlogan -sverilog -work stratixv_pcie_hip \
$QUARTUS_ROOTDIR/eda/sim_lib/synopsys/stratixv_pcie_hip_atoms_ncrypt.v ←

vhdlan -work stratixv_hssi \
$QUARTUS_ROOTDIR/eda/sim_lib/stratixv_hssi_components.vhd ←

vhdlan -work stratixv_hssi \
$QUARTUS_ROOTDIR/eda/sim_lib/stratixv_hssi_atoms.vhd ←

vcs test ←
./simv ←
```



The PCIe files are required only if you are using the PCIe HIP.

For VHDL, you must compile the Verilog HDL files first.

In addition to the top-level variant wrapper, `<variant>.v`, VCS also creates a simulation files subdirectory, `<variant>_sim/`. All Verilog (`.v`) and SystemVerilog (`.sv`) files in the simulation directory must also be compiled into the simulation project.

Transport Delays

By default, the VCS software filters out all pulses that are shorter than the propagation delay between primitives. Turning on the transport delay options in the VCS 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 3–2 describes the transport delay options.

Table 3–2. Transport Delay Options

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



The `+transport_path_delays` and `+transport_int_delays` options are also used by default in the NativeLink feature for gate-level timing simulation.



For more information about either of these options, refer to the *VCS User Guide* installed with the VCS software.

The following VCS software command shows the command-line syntax to perform a post-synthesis simulation with the device family library:

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

Using NativeLink with the VCS or VCS MX Software

The NativeLink feature in the Quartus II software facilitates the seamless transfer of information between the Quartus II software and EDA tools and allows you to run VCS or VCS MX within the Quartus II software.



For more information, refer to the “Using the NativeLink Feature” section in the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

Generating a Timing .vcd File for the PowerPlay Power Analyzer

To generate a timing Verilog Value Change Dump File (.vcd) for PowerPlay, you must first generate a .vcd in the Quartus II software, and then run the .vcd from the VCS software. This .vcd can then be used by PowerPlay for power analysis.

To generate timing .vcd in the Quartus II software, follow these steps:

1. In the Quartus II software, on the Assignments menu, click **Settings**. The **Settings** dialog box appears.

2. In the **Category** list, under **EDA Tool Settings**, click **Simulation**. On the **Simulation** page, in the **Tool name** list, select **VCS** and turn on the **Generate Value Change Dump (VCD) file script** option.
3. To generate the **.vcd**, perform a full compilation.

Perform the following steps to generate a timing **.vcd** file in the VCS software:

1. Before compiling and simulating your design, include the script in your testbench file where the design under test (DUT) is instantiated:

```
include <revision_name>_dump_all_vcd_nodes.v ←
```



Include the script within the testbench module block. If you include the script outside of the testbench module block, syntax errors occur during compilation.

2. Run the simulation with the VCS command as usual. Exit the VCS software when the simulation is finished and the **<revision_name>.vcd** file is generated in the simulation directory.



The **.vcd** file is not supported in the VCS MX software.



For more detailed information about using the timing **.vcd** file for power analysis, refer to the *PowerPlay Power Analysis* chapter in volume 3 of the *Quartus II Handbook*.

Viewing a Waveform from a .vpd or .vcd File

A Virtual Panoramic Display (**.vpd**) file is automatically generated when your simulation is finished. The **.vpd** file is not readable. It is used for generating the waveform view through DVE. You can view your waveform result in DVE if you have created a **.vpd** or **.vcd** file.

To view a waveform from a **.vpd** file through DVE, follow these steps:

1. Type **dve** on a command line. The **DVE** dialog box appears.
2. On the File menu, click **Open Database**. The **Open Database** dialog box appears.
3. Browse to the directory that contains your **.vpd** file (for example, **inter.vpd**).
4. Double-click the **.vpd** file.
5. In the **DVE** dialog box, select the signals that you want to observe from the Hierarchy.
6. On the Signal menu, click **Add To Waves**.
7. Click **New Wave View**. The waveform appears.

You cannot view a waveform from a **.vcd** file in DVE directly. The **.vcd** file must first be converted to a **.vpd** file. To convert the file, follow these steps:

1. Use the **vcd2vpd** command to convert the file. For example, type the following on a command line:

```
vcd2vpd <example>.vcd <example>.vpd ←
```

2. After you convert the **.vcd** file to a **.vpd** file, follow the procedures for viewing a waveform from a **.vpd** file through DVE.

You can also convert your `.vpd` file to a `.vcd` file with the `vpd2vcd` command.

Scripting Support

You can run procedures and create settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt.



For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

For detailed information about scripting command options, refer to the Qhelp utility.

To start the **Qhelp** utility, type the following command:

```
quartus_sh --qhelp ↵
```

Generating a Post-Synthesis Simulation Netlist File for VCS

You can use the Quartus II software to generate a post-synthesis simulation netlist file with Tcl commands or with a command at a command prompt.

Tcl Commands

To generate a post-synthesis simulation netlist file when you compile your design or as part of a Tcl script that compiles your design, type the following Tcl commands:

```
set_global_assignment -name EDA_SIMULATION_TOOL "VCS" ↵  
set_global_assignment -name EDA_GENERATE_FUNCTIONAL_NETLIST ON ↵
```

Command Prompt

To generate a simulation output file for the VCS software simulator, type the following command (specify VHDL or Verilog HDL for the format):

```
quartus_eda <project name> --simulation=on --format=<format> \  
--tool=vcs --functional ↵
```

Generating a Gate-Level Timing Simulation Netlist File for VCS

You can use the Quartus II software to generate a gate-level timing simulation netlist file with Tcl commands or with a command at a command prompt.

Tcl Commands

To generate a gate-level timing simulation netlist file, type the following Tcl command:

```
set_global_assignment -name EDA_SIMULATION_TOOL "VCS" ↵
```

Command Prompt

To generate a simulation output file for the VCS software simulator, type the following command (specify Verilog HDL, SystemVerilog HDL, or VHDL for the format):

```
quartus_eda <project name> --simulation=on --format=<format> --tool=vcs ↵
```

Conclusion



You can use the Synopsys VCS or VCS MX software in your Altera FPGA design flow to easily and accurately perform functional simulations, post-synthesis simulations, and gate-level functional timing simulations. The seamless integration of the Quartus II software and VCS or VCS MX software make this simulation flow an ideal method for fully verifying an FPGA design.

Document Revision History

Table 3-3 shows the revision history for this chapter.

Table 3-3. Document Revision History

Date	Version	Changes
November 2011	11.0.1	Template update. Minor editorial updates.
May 2011	11.0.0	<ul style="list-style-type: none"> ■ Linked to Help for Stratix V Libraries ■ Added SystemVerilog HDL information ■ Editorial updates throughout
December 2010	10.0.1	Changed to new document template. No change to content.
July 2010	10.0.0	<ul style="list-style-type: none"> ■ Linked to Quartus II Help where appropriate ■ Added Stratix V simulation information ■ Minor text edits ■ Removed VirSim references ■ Removed Referenced Documents section
November 2009	9.1.0	<ul style="list-style-type: none"> ■ Removed NativeLink information and referenced new <i>Simulating Designs with EDA Tools</i> chapter in volume 3 of the <i>Quartus II Handbook</i> ■ Added “RTL Functional Simulation for Stratix IV Devices” and “Gate-Level Timing Simulation for Stratix IV Devices” sections ■ Minor text edits
March 2009	9.0.0	<ul style="list-style-type: none"> ■ Added support for Synopsys VCS MX software ■ Changed chapter title to “Synopsys VCS and VCS MX Support” ■ Major revision to “Compiling Libraries Using the EDA Simulation Library Compiler” on page 4-2 ■ Major revision to “RTL Functional Simulations” on page 4-2 ■ Added Table 3-4 on page 3-10 and Table 3-5 on page 3-11 ■ Added new section “Using DVE” on page 4-7 ■ Added new section “Generating a Simulation Script from the EDA Netlist Writer” on page 3-16 ■ Added new section “Viewing a Waveform from a .vpd or .vcd File” on page 4-13
November 2008	8.1.0	<ul style="list-style-type: none"> ■ Added “Compile Libraries Using the EDA Simulation Library Compiler” on page 3-3 ■ Added information about the <code>--simlib_comp</code> utility ■ Updated entire chapter using 8½” × 11” chapter template ■ Minor editorial updates

-  For previous versions of the *Quartus II Handbook*, refer to the [Quartus II Handbook Archive](#).
-  Take an [online survey](#) to provide feedback about this handbook chapter.

