


This chapter describes how to use the Active-HDL and Riviera-PRO 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 Active-HDL or Riviera-PRO Software in Quartus II Design Flows”
- “Simulation Libraries” on page 5–2
- “Performing Simulation with the Active-HDL and Riviera-PRO Software” on page 5–3
- “Functional Simulation” on page 5–4
- “Post-Synthesis Simulation” on page 5–6
- “Gate-Level Timing Simulation” on page 5–9
- “Compiling SystemVerilog Files” on page 5–9
- “Simulating Designs that Include Transceivers” on page 5–10
- “Using the NativeLink Feature in Active-HDL or Riviera-PRO Software” on page 5–17
- “Generating .vcd Files for the PowerPlay Power Analyzer” on page 5–17
- “Scripting Support” on page 5–18

Software Requirements

To simulate your design with the Active-HDL or Riviera-PRO 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 directories created during the Quartus II software installation, refer to the *Altera Software Installation and Licensing* manual.

Using Active-HDL or Riviera-PRO Software in Quartus II Design Flows

You can perform the following types of simulations with the Active-HDL or Riviera-PRO 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.

Simulation Libraries

Simulation model libraries are required to run a simulation whether you are running a functional simulation, post-synthesis simulation, or gate-level timing simulation. In general, running a functional simulation requires the functional simulation model libraries and running a post-synthesis or gate-level timing simulation requires the gate-level timing simulation model libraries. You must compile the necessary library files before you can run the simulation.

However, there are a few exceptions where you are also required to compile gate-level timing simulation library files to perform functional simulation. For example, some of Altera megafunctions require gate-level libraries to perform a functional simulation in third-party simulators.



For each megafunction that you are using, refer to the last page in the Altera MegaWizard™ Plug-In Manager, which lists the simulation library files required to perform a functional simulation for that megafunction.

The transceiver megafunction (for example, ALTGX) also requires the gate-level libraries to perform functional simulation.

For detailed, step-by-step instructions about how to simulate the transceiver megafunction, refer to “[Simulating Designs that Include Transceivers](#)” on page 5–10.

Simulation Library Files in the Quartus II Software

In Active-HDL or Riviera-PRO software, you must compile the necessary libraries to perform functional, post-synthesis functional, or gate-level timing simulations. You can refer to these library files for the particular simulation model that you are looking for.



For a list of all functional simulation library files in the Quartus II directory, refer to *Altera Functional Simulation Libraries* in Quartus II Help. For a list of all gate-level timing simulation and post-fit library files in the Quartus II directory, refer to *Altera Post-Fit Libraries* in Quartus II Help.

Compiling Libraries with the EDA Simulation Library Compiler

The EDA Simulation Library Compiler is used to compile Verilog HDL, SystemVerilog HDL, and VHDL simulation libraries for all Altera devices and supported third-party simulators. You can use this tool to compile all libraries required by gate-level timing simulation.



The `altera_mf_components.vhd` and `altera_mf.vhd` model files should be compiled into the `altera_mf` library. The `220pack.vhd` and `220model.vhd` model files should be compiled into the `lpm` library.



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

Performing Simulation with the Active-HDL and Riviera-PRO Software

Perform simulation of Verilog HDL or VHDL designs with the Active-HDL and Riviera-PRO software at various levels to verify designs from different aspects. There are three types of simulation:

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

Simulation helps you verify your designs and debug any possible errors. The following sections provide instructions to perform the simulations with the GUI and from the command line.

For high-speed simulation, you must select **ps** in the **Resolution** list for your simulator resolutions. If you choose slower than **ps**, the high-speed simulation may fail.

Workspace creation is the mandatory first step to start working in the Active-HDL GUI. You must create a new workspace and add the simulation model files, design files, and testbench file to the workspace before you can compile them. To create and open the workspace, type the following commands:

```
createdesign DELAY_TEST C:/DELAY_TEST/simulation/activehdl ←  
opendesign -a DELAY_TEST.adf ←
```

If you are running Riviera-PRO, you can skip the step above.

In command-line mode, standalone commands, such as `vlib`, `vcom`, and `vsim`, are executed in the system shell (for example, `cygwin`). These standalone commands can be grouped into script files (**tcl**, **perl**, **windows batch**) that are run from the system shell.

Before running Active-HDL or Riviera-PRO from the command line, ensure that the **Active-HDL/bin** or **Riviera-PRO/bin** directory is located in `PATH` environment variables.

Functional Simulation

This section describes performing functional simulation of VHDL and Verilog HDL designs with the Active-HDL and Riviera-PRO software with the GUI and from the command line.

Simulating VHDL Designs with the Active-HDL GUI

When you simulate VHDL designs with the Active-HDL GUI, you do not have to remember the commands to compile the libraries or load and simulate the VHDL design files. You can use the Active-HDL GUI to perform a functional simulation, post-synthesis simulation, or a gate-level timing simulation.

Functional simulation is typically performed to verify the syntax of the code and to check the functionality of the design.

- ② For detailed information about how to perform functional simulation in the Active-HDL software for VHDL designs, refer to *Performing a Simulation of a VHDL Design with the Active-HDL Software* in Quartus II Help.
- ② If you are compiling Stratix® V libraries, refer to *Guidelines for Compiling Stratix V Libraries* in Quartus II Help.

Simulating Verilog HDL Designs with the Active-HDL GUI

When you simulate Verilog HDL designs with the Active-HDL GUI, you do not have to remember the commands to compile the libraries or load and simulate the Verilog HDL design files. You can use the Active-HDL GUI to perform functional simulation, post-synthesis simulation, and gate-level timing simulation.

Functional simulation is performed to verify the syntax of the code and to check the functionality of the design.

- ② For detailed information about how to perform functional simulation in the Active-HDL software for Verilog HDL designs, refer to *Performing a Simulation of a Verilog HDL Design with the Active-HDL Software* in Quartus II Help.
- ② If you are compiling Stratix V libraries, refer to *Guidelines for Compiling Stratix V Libraries* in Quartus II Help.

Simulating VHDL Designs with Active-HDL from the Command Line

To perform a functional simulation for VHDL designs, follow these steps:

1. To create and compile Altera libraries, type the following commands:

```
vlib <lib1> ←
vcom -strict93 -dbg -work <lib1> <lib1_component/pack.vhd> \
<lib1.vhd> ←

vlib <lib2> ←
vcom -strict93 -dbg -work <lib2> <lib2_component/pack.vhd> \
<lib2.vhd> ←
```

2. To create the work library and compile the design files and testbench file, type the following commands:

```
vlib work ←  
vcom -strict93 -dbg -work work <design_file1.vhd> <design \  
file2.vhd> <testbench file.vhd> ←
```

3. To load the design, type the following command:

```
vsim +access +r -t lps -L work -L <lib1> -L <lib2> work. \  
<testbench module name> ←
```

4. To add signals at the waveform and run the simulation, type the following commands:

```
add wave * ←  
run ←
```

Example

```
vlib vhd1_libs/lpm  
vcom -strict93 -dbg -work lpm  
c:/altera/91/quartus/eda/sim_lib/220pack.vhd  
vcom -strict93 -dbg -work lpm  
c:/altera/91/quartus/eda/sim_lib/220model.vhd  
  
vlib vhd1_libs/altera_mf  
vcom -strict93 -dbg -work altera_mf  
c:/altera/91/quartus/eda/sim_lib/altera_mf_components.vhd  
vcom -strict93 -dbg -work altera_mf  
c:/altera/91/quartus/eda/sim_lib/altera_mf.vhd  
  
vlib work  
vcom -strict93 -dbg -work work C:/project/adder.vhd  
C:/project/adder.vht  
  
vsim +access +r -t lps -L adder -L work -L lpm -L altera_mf  
work.adder_vhd_vec_tst  
  
#add signals at waveform and run  
add wave *  
run
```

Simulating Verilog HDL Designs with Active-HDL from the Command Line

To perform a functional simulation for Verilog HDL designs with one of the libraries (lib1) listed in *Altera Functional Simulation Libraries* in Quartus II Help, follow these steps:

1. To create and compile Altera libraries, type the following commands:

```
vlib <lib1> ←  
vlog -v2k -dbg -work <lib1> <lib1.v> ←  
  
vlib <lib2> ←  
vlog -v2k -dbg -work <lib2> <lib2.v> ←
```

2. To create work library and compile design files and testbench file, type the following commands:

```
vlib work ←  
vlog -v2k -dbg -work work <design_file1.v> <design file2.v> \  
<testbench \ file.v> ←
```

3. To load the design, type the following command:

```
vsim +access +r -t lps -L work -L <lib1> -L <lib2> work.<testbench \  
module name> ←
```

- To add signals at the waverform and run the simulation, type the following commands:

```
add wave * ←
run ←
```

Example

```
vlib verilog_libs/lpm_ver
vlog -v2k -dbg -work lpm_ver c:/altera/91/quartus/eda/sim_lib/220model.v

vlib verilog_libs/altera_mf_ver
vlog -v2k -dbg -work altera_mf_ver
c:/altera/91/quartus/eda/sim_lib/altera_mf.v

vlib work
vlog -v2k -dbg -work work C:/project/adder.v C:/project/adder.vt

vsim +access +r -t lps -L work -L lpm_ver -L altera_mf_ver
work.adder_vlg_vec_tst

add wave *
run
```

Simulating VHDL and Verilog HDL Designs with the Riviera-PRO GUI



For information about how to perform a functional simulation with the Riviera-PRO GUI, refer to the Riviera-PRO documentation from Aldec, Inc.

- ⊙ If you are compiling Stratix V libraries, refer to *Guidelines for Compiling Stratix V Libraries* in Quartus II Help.

Simulating VHDL and Verilog HDL Designs with Riviera-PRO from the Command Line

- ⊙ For information about how to perform a functional simulation with the Riviera-PRO software from the command line, refer to *Performing an RTL Functional Simulation with the Riviera-PRO Software* in Quartus II Help.

Post-Synthesis Simulation

Before you run post-synthesis simulation, generate post-synthesis simulation netlist files.

- ⊙ For information about how to generate a post-synthesis simulation netlist file, refer to *Generating Simulation Netlist Files* in Quartus II Help.



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

Simulating VHDL Designs with the Active-HDL GUI

- ⊙ For information about how to perform a post-synthesis simulation with the Active-HDL GUI, refer to *Performing a Simulation of a VHDL Design with the Active-HDL Software* in Quartus II Help.

Simulating Verilog Designs with the Active-HDL GUI

- ❓ For information about how to perform a post-synthesis simulation with the Active-HDL GUI, refer to *Performing a Simulation of a Verilog HDL Design with the Active-HDL Software* in Quartus II Help.

Simulating VHDL Designs with Active-HDL from the Command Line

To perform a post-synthesis simulation for VHDL designs, follow these steps:

1. To create and compile Altera libraries, type the following commands:

```
vlib <lib1> ␣  
vcom -strict93 -dbg -work <lib1> <lib1_component/-pack.vhd> \  
<lib1.vhd> ␣  
  
vlib <lib1> ␣  
vcom -strict93 -dbg -work <lib2> <lib2_component/-pack.vhd> \  
<lib2.vhd> ␣
```

2. To create the work library and compile the EDA output netlist files and testbench file, type the following commands:

```
vlib work ␣  
vcom -strict93 -dbg -work work <EDA output netlist.vho> \  
<testbench file.vhd> ␣
```

3. To load the design, type the following command:

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

4. To add signals at the waveform and run the simulation, type the following commands:

```
add wave * ␣  
run ␣
```

Example

```
vlib vhdl_libs/lpm  
vcom -strict93 -dbg -work lpm  
c:/altera/91/quartus/eda/sim_lib/220pack.vhd  
vcom -strict93 -dbg -work lpm  
c:/altera/91/quartus/eda/sim_lib/220model.vhd  
  
vlib vhdl_libs/altera  
vcom -strict93 -dbg -work altera  
c:/altera/91/quartus/eda/sim_lib/altera_primitives_components.vhd  
vcom -strict93 -dbg -work altera  
c:/altera/91/quartus/eda/sim_lib/altera_primitives.vhd  
  
vlib work  
vcom -strict93 -dbg -work work  
C:/project/simulation/activehdl/adder.vho C:/project/adder.vht  
  
vsim +access +r -t lps +transport_int_delays +transport_path_delays -L  
adder -L work -L lpm -L altera_mf work.adder_vhd_vec_tst  
  
add wave *  
run
```

Simulating Verilog HDL Designs with Active-HDL from the Command Line

To perform a post-synthesis simulation for Verilog HDL designs, follow these steps:

1. To create and compile Altera libraries, type the following commands:

```
vlib <lib1> ↵
vlog -v2k -dbg -work <lib1> <lib1.v> ↵

vlib <lib2> ↵
vlog -v2k -dbg -work <lib2> <lib2.v> ↵
```

2. To create the work library and compile the EDA output netlist files and testbench file, type the following commands:

```
vlib work ↵
vlog -v2k -dbg -work work <EDA_output_netlist.vo> <testbench file.v>
↵
```

3. To load the design, type the following command:

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

4. To add signals at the waveform and run the simulation, type the following commands:

```
add wave * ↵
run ↵
```

Example

```
vlib verilog_libs/lpm_ver
vlog -v2k -dbg -work lpm_ver
c:/altera/91/quartus/eda/sim_lib/220model.v

vlib verilog_libs/altera_ver
vlog -v2k -dbg -work altera_ver
c:/altera/91/quartus/eda/sim_lib/altera_primitives.v

vlib verilog_libs/stratixiv_ver
vlog -v2k -dbg -work stratixiv_ver
c:/altera/91/quartus/eda/sim_lib/stratixiv_atoms.v

vlib work

vlog -v2k -dbg -work work C:/project/simulation/activehdl/adder.vo
C:/project/adder.vt

vsim +access +r -t lps +transport_int_delays +transport_path_delays -L
work -L lpm_ver -L altera_mf_ver work.adder_vlg_vec_tst

add wave *
run
```

Simulating VHDL and Verilog HDL Designs with the Riviera-PRO GUI



For information about how to perform a post-synthesis simulation with the Riviera-PRO GUI, refer to the Riviera-PRO documentation from Aldec, Inc.

Simulating VHDL and Verilog HDL Designs with Riviera-PRO from the Command Line

- For information about how to perform a post-synthesis simulation with the Riviera-PRO software from the command line, refer to *Performing a Post-Synthesis Simulation with the Riviera-PRO Software* in Quartus II Help.

Gate-Level Timing Simulation

The steps for gate-level timing simulation are almost same as the steps for post-synthesis simulation. The only difference is that the Standard Delay Output (.sdo) file must be back-annotated for gate level-timing simulation.

For Verilog HDL designs, the back-annotating process is done within the `EDA_output_netlist.vo` script. Therefore, you are not required to back-annotate the .sdo again.

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

Disabling Timing Violation on Registers

In certain situations, timing violation can be ignored and you can disable the timing violation 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 is **On**, which means the simulation shows “x” whenever a timing violation occurs. To disable showing the timing violation on certain registers, set the `x_on_violation_option logic` option to **Off** on those registers. The following command is an example of the QSF file:

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

For VHDL designs, the back-annotating process is done by adding the `-sdftyp` option.

Example

```
vsim +access +r -t lps +transport_int_delays +transport_path_delays  
-sdftyp <instance path to design>= <path to SDO file> -L adder -L work  
-L lpm -L altera_mf work.adder_vhd_vec_tst
```

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, it is recommended that you compile the Verilog files, and then compile only the SystemVerilog files in the single `alog` command.

Simulating Designs that Include Transceivers

If your design includes Arria®, Arria II, Cyclone® IV, HardCopy® IV, Stratix, Stratix II, or Stratix IV transceivers, you must compile additional library files to perform functional or gate-level timing simulations. The following example shows how to perform simulation on designs that include Stratix GX and Stratix II GX transceivers.

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 may fail.

Performing simulation with transceivers in Arria GX or Stratix II GX is very similar. The only requirement is to replace the **stratixiigx_atoms** and **stratixiigx_hssi_atoms** model files with the **arriagx_atoms** and **arriagx_hssi_atoms** model files, respectively.



If your design contains PCI Express Hard IP, refer to the “Simulate the Design” section in the *IP Compiler for PCI Express User Guide*.

Functional Simulation for Stratix II GX Devices

Functional simulation for Stratix II GX devices is similar to functional simulation for Arria GX devices. The following example shows only the functional simulation for designs that include transceivers in Stratix II GX devices. To simulate transceivers in Arria GX devices, replace the **stratixiigx_hssi** model file with the **arriagx_hssi** model file.

To perform an functional simulation of your design that instantiates the ALT2GXB megafunction, which enables the gigabit transceiver blocks on Stratix II GX devices, you must generate a functional simulation netlist and compile the **stratixiigx_hssi** model file into the **stratixiigx_hssi** library.



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

To run the functional simulation, you must generate a functional simulation netlist by turning on **Generate Simulation Model** in the **Simulation Libraries** tab of the ALT2GXB MegaWizard Plug-In Manager.

The `<alt2gxb entity name>.who` or `<alt2gxb module name>.vo` is generated in the current project directory.

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

Performing Functional Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-1](#).

Example 5-1.

```
vcom -work lpm 220pack.vhd 220model.vhd ←
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
vcom -work sgate sgate_pack.vhd sgate.vhd ←
vcom -work stratixiigx_hssi stratixiigx_hssi_components.vhd \
stratixiigx_hssi_atoms.vhd ←
vcom -work work <alt2gxb entity name>.vho ←
vcom -work work <my design>.vhd <my testbench>.vhd ←
vsim -L lpm -L altera_mf -L sgate -L stratixg_hssi work.<my testbench> ←
```

Performing Functional Simulation in Verilog HDL

To compile and simulate the design, Type the commands in [Example 5-2](#).

Example 5-2.

```
vlog -work lpm_ver 220model.v ←
vlog -work altera_mf_ver altera_mf.v ←
vlog -work sgate_ver sgate.v ←
vlog -work stratixiigx_hssi_ver stratixiigx_hssi_atoms.v ←
vlog -work work <alt2gxb module name>.vo ←
vlog -work work <my design>.v <my testbench>.v ←
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver -L stratixg_hssi \
work.<my testbench> ←
```

Gate-Level Timing Simulation for Stratix II GX Devices

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

Performing Gate-Level Timing Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-3](#).

Example 5-3.

```
vcom -work lpm 220pack.vhd 220model.vhd ←
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
vcom -work sgate sgate_pack.vhd sgate.vhd ←
vcom -work stratixiigx stratixiigx_atoms.vhd stratixiigx_components.vhd ←
vcom -work stratixiigx_hssi stratixiigx_hssi_components.vhd stratixiigx_hssi_atoms.vhd ←
vcom -work work <my design>.vho <my testbench>.vhd ←
vsim -L lpm -L altera_mf -L sgate -L stratixiigx -L stratixiigx_hssi \
work.<my testbench> -t ps -sdftyp <design instance>=<path to SDO file>.sdo \
+transport_int_delays +transport_path_delays ←
```

Performing Gate-Level Timing Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-4](#).

Example 5-4.

```
vlog -work lpm_ver 220model.v ←
vlog -work altera_mf_ver altera_mf.v ←
vlog -work sgate_ver sgate.v ←
vlog -work stratixiigx_ver stratixiigx_atoms.v ←
vlog -work stratixiigx_hssi_ver stratixiigx_hssi_atoms.v ←
vlog -work work <my design>.vo <my testbench>.v ←
vsim -L lpm -L altera_mf_ver -L sgate_ver -L stratixiigx_ver -L stratixiigx_hssi_ver \
work.<my testbench> -t ps +transport_int_delays +transport_path_delays ←
```

Functional Simulation for Stratix GX Devices

To perform a functional simulation of your design that instantiates the ALTGXB megafunction, which enables the 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.

Performing Functional Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-5](#).

Example 5-5.

```
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
vcom -work lpm 220pack.vhd 220model.vhd ←
vcom -work sgate sgate_pack.vhd sgate.vhd ←
vcom -work altgxb stratixgx_mf.vhd stratixgx_mf_components.vhd ←
vcom -work work<altgxb entity name>.vhd ←
vsim -L lpm -L altera_mf -L sgate -L altgxb work.<my testbench> ←
```

Performing Functional Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-6](#).

Example 5-6.

```
vlib work ←
vlib lpm_ver ←
vlib altera_mf_ver ←
vlib sgate_ver ←
vlib altgxb_ver ←
vlog -work lpm_ver 220model.v ←
vlog -work altera_mf_ver altera_mf.v ←
vlog -work sgate_ver sgate.v ←
vlog -work altgxb_ver stratixgx_mf.v ←
vlog -work work <altgxb module name>.v ←
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver -L altgxb_ver \
work.<my testbench> ←
```

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.

Performing Gate-Level Timing Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-7](#).

Example 5-7.

```
vcom -work lpm 220pack.vhd 220model.vhd ←  
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←  
vcom -work sgate sgate_pack.vhd sgate.vhd ←  
vcom -work stratixgx stratixgx_atoms.vhd stratixgx_components.vhd ←  
vcom -work stratixgx_gxb stratixgx_hssi_atoms.vhd \  
stratixgx_hssi_components.vhd ←  
vcom -work work <my design>.vho <my testbench>.vhd ←  
vsim -L lpm -L altera_mf -L sgate -L stratixgx -L stratixgx_gxb work. \  
<my testbench> -t ps -sdftyp <design instance>=<path to SDO file>.sdo \  
+transport_int_delays +transport_path_delays ←
```

Performing Gate-Level Timing Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-8](#).

Example 5-8.

```
vlog -work lpm_ver 220model.v ←  
vlog -work altera_mf_ver altera_mf.v ←  
vlog -work sgate_ver sgate.v ←  
vlog -work stratixgx_ver stratixgx_atoms.v ←  
vlog -work stratixgx_gxb_ver stratixgx_hssi_atoms.v ←  
vlog -work work <my design>.vo <my testbench>.v ←  
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver -L stratixgx_ver -L \  
stratixgx_gxb_ver work.<my testbench> -t ps +transport_int_delays  
+transport_path_delays ←
```

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.

The following example shows only the functional simulation for designs that include transceivers in Stratix IV devices. To simulate transceivers in Arria II, Cyclone IV, and HardCopy IV devices, 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, which enables the gigabit transceiver blocks on Stratix IV devices, you must generate a functional simulation netlist and compile the `stratixiv_hssi` model file into the `stratixiv_hssi` library.



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

Performing Functional Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-9](#).

Example 5-9.

```
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
vcom -work lpm 220pack.vhd 220model.vhd ←
vcom -work sgate sgate_pack.vhd sgate.vhd ←
vcom -work stratixiv_hssi stratixiv_hssi.vhd \
stratixiv_hssi_components.vhd ←
vcom -work work <altgxb entity name>.vhd ←
vsim -L lpm -L altera_mf -L sgate -L stratixiv_hssi work.<my testbench> ←
```

Performing Functional Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-10](#).

Example 5-10.

```
vlib work ←
vlib lpm_ver ←
vlib altera_mf_ver ←
vlib sgate_ver ←
vlib stratixiv_hssi_ver ←
vlog -work lpm_ver 220model.v ←
vlog -work altera_mf_ver altera_mf.v ←
vlog -work sgate_ver sgate.v ←
vlog -work stratixiv_hssi_ver stratixiv_hssi.v ←
vlog -work work <altgxb module name>.v ←
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver \
-L stratixiv_hssi_ver work.<my testbench> ←
```

Gate-Level Timing Simulation for Stratix IV GX Devices

Perform a gate-level timing simulation of your design that includes a Stratix IV transceiver by compiling the `stratixiv_atoms` and `stratixiv_hssi_atoms` model files into the `stratixiv` and `stratixiv_hssi` libraries, respectively.

Performing Gate-Level Timing Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-11](#).

Example 5-11.

```
vcom -work lpm 220pack.vhd 220model.vhd ←
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
vcom -work sgate sgate_pack.vhd sgate.vhd ←
vcom -work stratixiv stratixiv_atoms.vhd stratixiv_components.vhd ←
vcom -work stratixiv_hssi stratixiv_hssi_atoms.vhd \
stratixiv_hssi_components.vhd ←
vcom -work work <my design>.vho <my testbench>.vhd ←
vsim -L lpm -L altera_mf -L sgate -L stratixiv -L stratixiv_hssi \
work.<my testbench> -t ps \
-sdftyp <design instance>=<path to SDO file>.sdo \
+transport_int_delays +transport_path_delays ←
```

Performing Gate-Level Timing Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-12](#).

Example 5-12.

```
vlog -work lpm_ver 220model.v ←  
vlog -work altera_mf_ver altera_mf.v ←  
vlog -work sgate_ver sgate.v ←  
vlog -work stratixiv_ver stratixiv_atoms.v ←  
vlog -work stratixiv_hssi_ver stratixiv_hssi_atoms.v ←  
vlog -work work <my design>.vo <my testbench>.v ←  
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver -L stratixiv_ver -L  
stratixiv_hssi_ver \  
work.<my testbench> -t ps +transport_int_delays +transport_path_delays ←
```

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.

The following example shows only the functional simulation for designs that include transceivers in Stratix V devices. To simulate transceivers in Arria II, Cyclone IV, HardCopy IV, and Stratix V devices, replace the `stratixv_hssi` model file with the `arriaii_hssi`, `cycloneiv_hssi`, `hardcopyiv_hssi`, and `stratixiv_hssi` model files, respectively.



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

Performing Functional Simulation in VHDL

To compile and simulate the design, type the commands in [Example 5-13](#).

Example 5-13.


```
vcom -work altera_mf altera_mf_components.vhd altera_mf.vhd ←  
vcom -work lpm 220pack.vhd 220model.vhd ←  
vcom -work sgate sgate_pack.vhd sgate.vhd ←  
vlog +v2k -work stratixv_hssi \  
quartus/eda/sim_lib/aldec/stratixv_hssi_atoms_ncrypt.v ←  
vcom -work stratixv_hssi stratixiv_hssi.vhd \  
stratixiv_hssi_components.vhd  
vcom -work work <my design>.vhd ←  
vsim -L lpm -L altera_mf -L sgate -L stratixv_hssi work.<my testbench> ←
```


Performing Functional Simulation in Verilog HDL

To compile and simulate the design, type the commands in [Example 5-14](#).

Example 5-14.

```
vlib work ←
vlib lpm_ver ←
vlib altera_mf_ver ←
vlib sgate_ver ←
vlib stratixv_hssi_ver ←
vlog -work lpm_ver 220model.v ←
vlog -work altera_mf_ver altera_mf.v ←
vlog -work sgate_ver sgate.v ←
vlog +v2k -work stratixv_hssi \
quartus/eda/sim_lib/aldec/stratixv_hssi_atoms_ncrypt.v ←
vlog -work stratixv_hssi_ver stratixiv_hssi.v ←
vlog -work work <my design>.v ←
vsim -L lpm_ver -L altera_mf_ver -L sgate_ver \
-L stratixv_hssi_ver work.<my testbench> ←
```

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

 In addition to the top-level variant wrapper, *<variant>.v*, you also get 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 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 Active-HDL or Riviera-PRO software prevents the simulation tool from filtering out these pulses.

[Table 5-1](#) describes the transport delay options.

Table 5-1. 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 +pulse_e/number and +pulse_r/number 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 +pulse_int_e/number and +pulse_int_r/number 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 Active-HDL online documentation installed with the Active-HDL software.


To perform a gate-level timing simulation with the device family library, type the Active-HDL command shown in [Example 5-15](#).

Example 5-15.

```
vsim -t lps -L stratixii -sdftyp /il=filtref_vhd.sdo \  
work.filtref_vhd_vec_tst +transport_int_delays +transport_path_delays
```

Using the NativeLink Feature in Active-HDL or Riviera-PRO 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 the Active-HDL or Riviera-PRO software 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 .vcd Files for the PowerPlay Power Analyzer

To generate a Value Change Dump File (.vcd) for the PowerPlay power analyzer, you must first generate a VCD script in the Quartus II software and run the VCD script from the Active-HDL software to generate a .vcd. This .vcd can then be used by PowerPlay for power analysis. The following instructions show you how to generate a .vcd.

To generate VCD scripts 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, select **Simulator Settings**.
3. On the **Simulator Settings** page, in the **Tool name** list, select **Active-HDL** and turn on the **Generate Value Change Dump File Script** option.
4. To generate the VCD script file, perform a full compilation.

To generate a .vcd in the Active-HDL software, follow these steps:

1. In the Active-HDL software, before simulating your design, source the `<revision_name>_dump_all_vcd_nodes.tcl` script. To source the TCL script, type the following command before running the `vsim` command:




```
source <revision_name>_dump_all_vcd_nodes.tcl ↵
```

2. Continue to run the simulation until the simulation is completed. Exit the Active-HDL software. If you do not exit the software, the Active-HDL software may end the writing process of the .vcd files improperly, resulting in a corrupted VCD file.

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

Scripting Support

You can run procedures and create settings described in this chapter in a Tcl script. You can also run some procedures at the command-line 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** command line and Tcl API help browser.

To start the Qhelp help browser, type the following command:

```
quartus_sh -qhelp ←
```

Generating a Post-Synthesis Simulation Netlist for Active-HDL or Riviera-PRO

You can use the Quartus II software to generate a post-synthesis simulation netlist with Tcl commands or with a command at the command-line prompt. The following examples assume you are selecting Active-HDL or Riviera-PRO (Verilog HDL output from the Quartus II software).

Tcl Commands

To set the output format to Verilog HDL, the simulation tool to Active-HDL or Riviera-PRO for Verilog HDL, and to generate a functional netlist, type the following Tcl commands:

```
set_global_assignment-name EDA_SIMULATION_TOOL "Active-HDL (Verilog)" ←
set_global_assignment-name EDA_GENERATE_FUNCTIONAL_NETLIST ON ←
```

or

```
set_global_assignment-name EDA_SIMULATION_TOOL "Riviera-PRO (Verilog)" ←
set_global_assignment-name EDA_GENERATE_FUNCTIONAL_NETLIST ON ←
```

Command Line

To generate a simulation output file for the Active-HDL or Riviera-PRO software, type one of the following commands (specify VHDL or Verilog HDL for the format):

```
quartus_eda <project name> --simulation=on --format=<format> \
--tool=activehdl --functional ←
```

or

```
quartus_eda <project name> --simulation=on --format=<format> \
--tool=rivierapro --functional ←
```

Generating a Gate-Level Timing Simulation Netlist for Active-HDL or Riviera-PRO

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

Tcl Commands

Type one of the following Tcl commands:

```
set_global_assignment -name EDA_SIMULATION_TOOL "Active-HDL (Verilog)"
set_global_assignment -name EDA_SIMULATION_TOOL "Active-HDL (VHDL)"
or
set_global_assignment -name EDA_SIMULATION_TOOL "Riviera-PRO (Verilog)"
set_global_assignment -name EDA_SIMULATION_TOOL "Riviera-PRO (VHDL)"
```

Command Line

To generate a simulation output file for the Active-HDL or Riviera-PRO software by specifying VHDL or Verilog HDL for the format, type the following command at the command prompt:

```
quartus_eda <project name> --simulation=on --format=<format> \
--tool=activehdl
or
quartus_eda <project name> --simulation=on --format=<format> \
--tool=rivierapro
```

Conclusion

Using the Active-HDL or Riviera-PRO simulation software within the Altera FPGA design flow allows you to easily and accurately perform functional simulations, post-synthesis simulations, and gate-level timing simulations on your designs. Proper verification of designs at the functional, post-synthesis, and post place-and-route stages helps ensure your design functions correctly and, ultimately, a quick time-to-market.

Document Revision History


Table 5-2 shows the revision history for this chapter.


Table 5-2. 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. ■ Reorganized and reformatted chapter ■ Other minor changes throughout.
December 2010	10.0.1	<ul style="list-style-type: none"> ■ Changed to new document template. No change to content.

Table 5–2. Document Revision History

Date	Version	Changes
July 2010	10.0.0	<ul style="list-style-type: none"> ■ Linked to Quartus II Help ■ Revised simulation procedures ■ Added Stratix V simulation information ■ Added Riviera-PRO support ■ Minor text edits ■ Removed Referenced Documents section
November 2000	9.1.0	<ul style="list-style-type: none"> ■ Updated Table 6–1 ■ Removed Simulation Library tables and EDA Simulation Library Compiler sections and referenced new <i>Simulating Designs with EDA Tools</i> chapter ■ 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"> ■ Removed “Compile Libraries Using the Altera Simulation Library Compiler” ■ Added “Compile Libraries Using the EDA Simulation Library Compiler” on page 5–10 ■ Added “Generate Simulation Script from EDA Netlist Writer” on page 5–51 ■ Minor editorial updates
November 2008	8.1.0	<p>Added the following sections:</p> <ul style="list-style-type: none"> ■ “Compile Libraries Using the Altera Simulation Library Compiler” on page 5–10 ■ Added steps to the procedure “Performing an RTL Simulation Using NativeLink” on page 5–45 for using the Altera Simulation Library Compilation ■ Added steps to the procedure “Performing a Gate-Level Timing Simulation Using NativeLink” on page 5–47 for using the Altera Simulation Library Compilation ■ Minor editorial updates ■ Updated entire chapter using 8½” × 11” chapter template
May 2008	8.0.0	Initial release

 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.