

This chapter describes how to use the Cadence Incisive Enterprise Simulator (IES) 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”
- “Simulation Flow Overview”
- “Functional Simulation” on page 4–3
- “Post-Synthesis Simulation” on page 4–6
- “Gate-Level Timing Simulation” on page 4–7
- “Simulating Designs that Include Transceivers” on page 4–10
- “Using the NativeLink Feature with IES” on page 4–17
- “Generating a Timing VCD File for the PowerPlay Power Analyzer” on page 4–17
- “Viewing a Waveform from a .trn File” on page 4–18
- “Scripting Support” on page 4–19

Software Requirements

To simulate your design with the IES 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](#).

Simulation Flow Overview

The IES software supports the following simulation flows:

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

Functional simulation verifies the functionality of your design. When you perform a functional simulation with the IES software, you use your design files (Verilog HDL, SystemVerilog HDL, or VHDL) and the models provided with the Quartus II software. These Quartus II models are required if your design uses the library of parameterized modules (LPM) functions or Altera-specific megafunctions. Refer to [“Functional Simulation” on page 4-3](#) for more information about how to perform this simulation.

A post-synthesis simulation verifies the functionality of a design after synthesis has been performed. You can create a post-synthesis netlist (Verilog HDL Output File (.vo), SystemVerilog HDL Output File (.svo), or VHDL Output File (.vho)) in the Quartus II software and use this netlist to perform a post-synthesis simulation with the Incisive simulator. Refer to [“Post-Synthesis Simulation” on page 4-6](#) for more information about how to perform this simulation.

After performing place-and-route, the Quartus II software generates a .vo, .svo, or .vho and a Standard Delay Output file (.sdo) for gate-level timing simulation. The netlist files map your design to architecture-specific primitives. The .sdo contains the delay information of each architecture primitive and routing element specific to your design. Together, these files provide an accurate simulation of your design with the selected Altera FPGA architecture. Refer to [“Gate-Level Timing Simulation” on page 4-7](#) for more information about how to perform this simulation.

Operation Modes

You can use either the GUI mode or the command-line mode to simulate your design in the IES software.

To start the IES software in GUI mode, type the following command at a command prompt:

```
nclaunch ←
```

To simulate in command-line mode, use the programs shown in [Table 4-1](#).

Table 4-1. Command-Line Programs

Program	Function
ncvlog or ncvhdl	ncvlog compiles your Verilog HDL code and performs syntax and static semantics checks. ncvhdl compiles your VHDL code and performs syntax and static semantics checks.
ncelab	ncelab elaborates the design. ncelab constructs the design hierarchy and establishes signal connectivity.
ncsim	ncsim performs mixed-language simulation. This program is the simulation kernel that performs event scheduling and executes the simulation code.

Quartus II Software and IES Flow Overview

This section provides an overview of the Quartus II software and IES simulation flow. More detailed information is provided in [“Functional Simulation” on page 4-3](#), [“Post-Synthesis Simulation” on page 4-6](#), and [“Gate-Level Timing Simulation” on page 4-7](#).

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.

Complete the following tasks:

1. Create user libraries.

Create a file that maps logical library names to their physical locations. These library mappings include your working directory and any design-specific libraries; for example, Altera LPM functions or megafunctions.

2. Compile source code and testbenches.

Compile your design files at the command-line with the **ncvlog** (Verilog HDL files) or **ncvhdl** (VHDL files) command, or, on the Tools menu, click **Verilog Compiler** or **VHDL Compiler** in NCLaunch. During compilation, the IES software performs syntax and static semantic checks. If no errors are found, compilation produces an internal representation for each HDL design unit in the source files. By default, these intermediate objects are stored in a single, packed, library database file in your working directory.

3. Elaborate your design.

Before you can simulate your model, you must define the design hierarchy in a process called “elaboration”. Use **ncelab** in command-line mode or, on the Tools menu in NCLaunch, click **Elaborator**.

4. Add signals to your waveform.

Specify which signals to view in your waveform using a simulation history manager (SHM) database.

5. Simulate your design.

Run the simulator with the **ncsim** program (command-line mode) or by clicking **Run** in the SimVision Console window.

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 simulation with this tool.



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*.

Functional Simulation

The following sections provide detailed instructions for performing a functional simulation with the Quartus II software and the IES software.



For the Altera Behavioral Simulation Models, refer to *Altera Functional Simulation Libraries* in Quartus II Help.

Creating Libraries

To create libraries, follow these steps:

1. To create a directory for the work library and any other libraries you require, type the following command at a command prompt:

```
mkdir <library name> ↵
```

Examples

```
mkdir worklib ↵
mkdir altera_mf ↵
```

2. Using a text editor, create a **cds.lib** file and add the following line to it:

```
DEFINE <library name> <physical directory path>
```

Examples

```
DEFINE worklib ./worklib
DEFINE altera_mf ./altera_mf
```

- ❓ For information about creating a **cds.lib** file in GUI mode, refer to [Performing a Functional Simulation with the Incisive Enterprise Simulator Software](#) in Quartus II Help.
- ❓ If you are compiling Stratix® V libraries, refer to [Guidelines for Compiling Stratix V Libraries](#) in Quartus II Help.

Compiling Source Code

To compile from your source code from the command line, type one of the following commands:

- Verilog HDL:

```
ncvlog <options> -work <library name> <design files> ↵
```

- SystemVerilog HDL:

```
ncvlog -sv <options> -work <library name> <design files> ↵
```

- VHDL:

```
ncvhdl <options> -work <library name> <design files> ↵
```

You must create a work library before compiling your design and testbench. If your design uses LPM, Altera megafunctions, or Altera primitives, you must also compile the Altera-provided functional models. The commands in [Example 4-1](#) and [Example 4-2](#) show an example of each.

Example 4-1. Compile in Verilog HDL

```
ncvlog -WORK lpm 220model.v ↵
ncvlog -WORK altera_mf altera_mf.v ↵
ncvlog -WORK altera altera_primitives.v ↵
ncvlog -WORK altera altera_primitives.v ↵
ncvlog -WORK work toplevel.v testbench.v ↵
```

Example 4-2. Compile in VHDL

```
ncvhd1 -V93 -WORK lpm 220pack.vhd ←  
ncvhd1 -V93 -WORK lpm 220model.vhd ←  
ncvhd1 -V93 -WORK altera_mf altera_mf_components.vhd ←  
ncvhd1 -V93 -WORK altera_mf altera_mf.vhd ←  
ncvhd1 -V93 -WORK altera altera_primitives_components.vhd ←  
ncvhd1 -V93 -WORK altera altera_primitives.vhd ←  
ncvhd1 -V93 -WORK work toplevel.vhd testbench.vhd ←
```

- ❓ For information about compiling in GUI mode, refer to *Performing a Functional Simulation with the Incisive Enterprise Simulator Software* in Quartus II Help.

Elaborating Your Design

Before you can simulate your design, you must define the design hierarchy in a process called elaboration. The IES software elaborates your design with the language-independent **ncelab** program. The **ncelab** program constructs a design hierarchy based on the design's instantiation and configuration information, establishes signal connectivity, and computes initial values for all objects in the design. The elaborated design hierarchy is stored in a simulation snapshot, which is the representation of your design that the simulator uses to run the simulation. The snapshot is stored in the library database file, along with the other intermediate objects generated by the compiler and elaborator.

To elaborate your Verilog HDL, SystemVerilog HDL, or VHDL design from the command line, type the following command:

```
ncelab [options][<library>.<testbench module name>] ←
```

Example

```
ncelab -access +rwc work.testbench_module ←
```

Adding the option `-access +rwc` allows signals to be viewed in the Waveform window.

If your design includes high-speed signals, you might have to add the following pulse reject options with the `ncelab` command:

```
ncelab -access +rwc work.testbench_module -PULSE_R 0 -PULSE_INT_R 0 ←
```

- 🔗 For more information about the pulse reject options, refer to the *SDF Annotate Guide* from Cadence.

- ❓ For information about elaborating your design in GUI mode, refer to *Performing a Functional Simulation with the Incisive Enterprise Simulator Software* in Quartus II Help.

Simulating Your Design

After you have compiled and elaborated your design, you can simulate it with **ncsim**. The **ncsim** program loads the file, or snapshot, generated by **ncelab** as its primary input and then loads other intermediate objects referenced by the snapshot. If you enable interactive debugging, **ncsim** can also load HDL source files and script files. The simulation output is controlled by the model or debugger. The output can include result files generated by the model, the SHM database, or the `.vcd` file.

To perform functional simulation of your Verilog HDL, SystemVerilog HDL, or VHDL design at the command line, type the following command:

```
ncsim [options][<library>.<testbench module name>] ←
```

Example

```
ncsim -gui work.testbench_module ←
```

Adding the option `-gui` opens the SimVision window for running your simulation.

- ② For information about performing a functional simulation in GUI mode, refer to *Performing a Functional Simulation with the Incisive Enterprise Simulator Software* in Quartus II Help.

Post-Synthesis Simulation

The following sections provide detailed instructions for performing post-synthesis simulation with the IES software and output files and simulation files from the Quartus II software.



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

- ② For a list of the gate-level simulation models, refer to *Altera Post-Fit Libraries* in Quartus II Help.

Quartus II Simulation Output Files

After performing synthesis with either a third-party synthesis tool or with Quartus II integrated synthesis, you must generate a simulation netlist for functional simulations.

- ② For information about how to generate a post-synthesis simulation netlist, refer to *Generating Simulation Netlist Files* in Quartus II Help.

Creating Libraries

Create the following libraries for your simulation:

- Work library
- Device family library with the following files in the `<path to Quartus II installation>/eda/sim_lib` directory:
 - `<device_family>_atoms.v`
 - `<device_family>_atoms.vhd`
 - `<device_family>_components.vhd`

Refer to “[Creating Libraries](#)” on page 4-4 for instructions about creating libraries.

Compiling Project Files and Libraries

Compile the project files and libraries into your work directory with the **ncvlog** program, **ncvhdl** program, or the GUI. Include the following files:

- Testbench file
- The Quartus II software functional simulation output netlist file (**.vo** file or **.vho** file)
- Atom library file for the device family *<device family>_atoms.<v|vhd>*
- For VHDL, *<device family>_components.vhd*

Refer to “[Compiling Source Code](#)” on page 4-4 for instructions about compiling.

Elaborating Your Design




Elaborate your design with the **ncelab** program, as described in “[Elaborating Your Design](#)” on page 4-5.

Simulating Your Design

Simulate your design with the **ncsim** program, as described in “[Simulating Your Design](#)” on page 4-5.


Gate-Level Timing Simulation

The following sections provide detailed instructions for performing a gate-level simulation with the Quartus II output files, simulation libraries, and Cadence IES tools.

-  You cannot perform post-synthesis or gate-level simulations if you are targeting the Stratix V device family.
-  For a list of the gate-level simulation models, refer to [Altera Post-Fit Libraries](#) in Quartus II Help.
-  For details about how to perform gate-level timing simulation with the Quartus II software and the IES software, refer to [Performing a Timing Simulation with the Incisive Enterprise Simulator Software](#) in Quartus II Help.

Generating a Gate-Level Timing Simulation Netlist

To perform a gate-level timing simulation, your design should provide the IES software with information about how the design was placed into device-specific architectural blocks. The Quartus II software provides this information in the form of a **.vo** file for Verilog HDL designs, a **.svo** file for SystemVerilog HDL designs, and a **.vho** file for VHDL designs. The accompanying timing information is stored in the **.sdo** file, which annotates the delay for the elements found in the **.vo** file, **.svo** file, or **.vho** file.

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

Disabling Timing Violation on Registers

In certain situations, the timing violations 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 can be ignored and disabled.

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, you can set the `x_on_violation_option logic` option to **Off** for those registers.

To disable timing violation on registers, type the following Quartus II Tcl command:

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

This Tcl command is also stored in the `.qsf` file.

Creating Libraries

Create the following libraries for your simulation:

- Work library
- Device family libraries with the following files in the `<path to Quartus II installation>/eda/sim_lib` directory:
 - `<device_family>_atoms.v`
 - `<device_family>_atoms.vhd`
 - `<device_family>_components.vhd`

For instructions about creating libraries, refer to [“Creating Libraries” on page 4-4](#).

Compiling Project Files and Libraries


Compile the project files and libraries into your work directory with the `ncvlog` program, `ncvhdl` program, or the GUI. Include the following files:

- Testbench file
- The Quartus II software functional output netlist file (`.vo` file, `.svo` file, or `.vho` file)
- Atom library file for the device family `<device family>_atoms.<v | vhd>`
- For VHDL, `<device family>_components.vhd`

For instructions about compiling, refer to [“Compiling Source Code” on page 4-4](#).

Elaborating Your Design

When performing elaboration with the Quartus II-generated Verilog HDL or SystemVerilog HDL netlist file, the `.sdo` file is read automatically. The `ncelab` executable recognizes the embedded system task `$sdf_annotate` and automatically compiles and annotates the `.sdo` file (runs `ncsdhc` automatically).

 The `.sdo` file should be located in the same directory where you perform an elaboration or simulation, because the `$sdf_annotate` task references the `.sdo` file without using a full path. If you are starting an elaboration or simulation from a different directory, you can either comment out the `$sdf_annotate` and annotate the `.sdo` file with the GUI, or add the full path of the `.sdo` file.

Refer to “[Elaborating Your Design](#)” on page 4-5 for elaboration instructions.

VHDL netlist files do not contain system task calls to locate your `.sdf` file; therefore, you must compile the standard `.sdo` file manually. For information about compiling the `.sdo` file, refer to “[Compiling the .sdo File \(VHDL Only\) in Command-Line Mode](#)” and “[Compiling the .sdo File \(VHDL Only\) in GUI Mode](#)”.


Compiling the .sdo File (VHDL Only) in Command-Line Mode

To annotate the `.sdo` file timing data from the command line, follow these steps:

1. To compile the `.sdo` file with the `ncsdfc` program, type the following command at the command prompt:

```
ncsdfc <project name>_vhd.sdo -output <output name> ↵
```

The `ncsdfc` program generates an `<output name>.sdf.X` compiled `.sdo` file.

 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>
```

[Example 4-3](#) shows an example of an SDF command file.


Example 4-3. 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";
```

After you compile the `.sdf` file, type the following command to elaborate the design:

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

Compiling the .sdo File (VHDL Only) in GUI Mode

-  To compile the `.sdo` file in GUI mode, refer to [Performing a Timing Simulation with the Incisive Enterprise Simulator Software](#) in Quartus II Help.

Simulating Your Design

Simulate your design with the `ncsim` program, as described in “[Simulating Your Design](#)” on page 4-5.

- For the design examples to run gate-level timing simulation, refer to the [Cadence NC-Sim Simulation Design Example](#) web page.

Simulating Designs that Include Transceivers

If your design includes Arria®, Arria II, Cyclone IV®, HardCopy IV®, Stratix, Stratix II, or 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](#).

Functional Simulation for Stratix GX Devices

To perform a functional simulation of your design that instantiates the ALTGXB megafunction, enabling the gigabit transceiver block (GXB) 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 VHDL design targeting a Stratix GX device, type the commands shown in [Example 4-4](#) at the IES command prompt.

Example 4-4. Compile Libraries Commands for Functional Simulation in VHDL

```
ncvhdl -work lpm 220pack.vhd 220model.vhd ←
ncvhdl -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
ncvhdl -work sgate sgate_pack.vhd sgate.vhd ←
ncvhdl -work altgxb stratixgx_mf.vhd stratixgx_mf_components.vhd ←
ncsim work.<my design> ←
```

To compile the libraries necessary for a functional simulation of a Verilog HDL design targeting a Stratix GX device, type the commands shown in [Example 4-5](#) at the IES command prompt.

Example 4-5. Compile Libraries Commands for Functional Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ←
ncvlog -work altera_mf altera_mf.v ←
ncvlog -work sgate sgate.v ←
ncvlog -work altgxb stratixgx_mf.v ←
ncsim work.<my design> ←
```

Gate-Level Timing Simulation for Stratix GX Devices

To perform a gate-level timing simulation of your design that includes a Stratix GX transceiver, compile the `stratixgx_atoms` and `stratixgx_hssi_atoms` model files into the `stratixgx` and `stratixgx_gxb` libraries, respectively.



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

To compile the libraries necessary for a timing simulation of a VHDL design targeting a Stratix GX device, type the commands shown in [Example 4-6](#) at the IES command prompt.

Example 4-6. Compile Libraries Commands for Timing Simulation in VHDL

```
ncvhd1 -work lpm 220pack.vhd 220model.vhd ↵
ncvhd1 -work altera_mf altera_mf_components.vhd altera_mf.vhd ↵
ncvhd1 -work sgate sgate_pack.vhd sgate.vhd ↵
ncvhd1 -work stratixgx stratixgx_atoms.vhd stratixgx_components.vhd ↵
ncvhd1 -work stratixgx_gxb stratixgx_hssi_atoms.vhd \
stratixgx_hssi_components.vhd ↵
ncelab work.<my design> -TIMESCALE 1ps/1ps -PULSE_R 0 -PULSE_INT_R 0 ↵
```

To compile the libraries necessary for a timing simulation of a Verilog HDL design targeting a Stratix GX device, type the commands shown in [Example 4-7](#) at the IES command prompt.

Example 4-7. Compile Libraries Commands for Timing Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ↵
ncvlog -work altera_mf altera_mf.v ↵
ncvlog -work sgate sgate.v ↵
ncvlog -work stratixgx stratixgx_atoms.v ↵
ncvlog -work stratixgx_gxb stratixgx_hssi_atoms.v ↵
ncelab work.<my design> -TIMESCALE 1ps/1ps -PULSE_R 0 -PULSE_INT_R 0 ↵
```

Functional Simulation for Stratix II GX Devices

Functional simulation of Stratix II GX devices is similar to functional simulation of Arria GX devices. [Example 4-9 on page 4-12](#) and [“Compile Libraries Commands for Functional Simulation in Verilog HDL” on page 4-12](#) show 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 a functional simulation of your design that instantiates the ALT2GXB megafunction, edit your `cds.lib` file so all of the libraries point to the work library, and compile the `stratixiigx_hssi` model file into the `stratixiigx_hssi` library. When compiling the library files, you can safely ignore the following warning message:

```
"Multiple logical libraries mapped to a single location"
```

Example 4-8 shows the `cds.lib` file.

Example 4-8. `cds.lib` File

```
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvhdl.lib
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvlog.lib
DEFINE work ./ncsim_work
DEFINE stratixiigx_hssi ./ncsim_work
DEFINE stratixiigx ./ncsim_work
DEFINE lpm ./ncsim_work
DEFINE sgate ./ncsim_work
```



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 by turning on **Create a simulation library for this design** in the last page of the ALT2GXB MegaWizard Plug-In Manager. The `<alt2gxb entity name>.vho` 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.

To compile the libraries necessary for functional simulation of a VHDL design targeting a Stratix II GX device, type the commands shown in Example 4-9 at the IES command prompt.

Example 4-9. Compile Libraries Commands for Functional Simulation in VHDL

```
ncvhd1 -work lpm 220pack.vhd 220model.vhd ←
ncvhd1 -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
ncvhd1 -work sgate sgate_pack.vhd sgate.vhd ←
ncvhd1 -work stratixiigx_hssi stratixiigx_hssi_components.vhd \
stratixiigx_hssi_atoms.vhd ←
ncvhd1 -work work <alt2gxb entity name>.vho ←
ncelab work.<my design> ←
```

To compile the libraries necessary for functional simulation of a Verilog HDL design targeting a Stratix II GX device, type the commands shown in Example 4-10 at the IES command prompt.

Example 4-10. Compile Libraries Commands for Functional Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ←
ncvlog -work altera_mf altera_mf.v ←
ncvlog -work sgate sgate.v ←
ncvlog -work stratixiigx_hssi stratixiigx_hssi_atoms.v ←
ncvlog -work work <alt2gxb module name>.vo ←
ncelab work.<my design> ←
```

Gate-Level Timing Simulation for Stratix II GX Devices

Stratix II GX functional simulation is similar to Arria GX functional simulation.

[Example 4-11 on page 4-13](#) and [Example 4-12 on page 4-13](#) show only the gate-level timing simulation for designs that include transceivers in Stratix II GX. To simulate transceivers in Arria GX, replace the `stratixiigx_hssi` model file with the `arriagx_hssi` model file.

To perform a post-fit timing simulation of your design that includes the ALT2GXB megafunction, edit your `cds.lib` file so that all the libraries point to the work library and compile `stratixiigx_atoms` and `stratixiigx_hssi_atoms` into the `stratixiigx` and `stratixiigx_hssi` libraries, respectively. When compiling the library files, you can safely ignore the following warning message:

```
"Multiple logical libraries mapped to a single location"
```

For an example of a `cds.lib` file, refer to [“Functional Simulation for Stratix II GX Devices” on page 4-11](#).



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 VHDL design targeting a Stratix II GX device, type the commands shown in [Example 4-11](#) at the IES command prompt.

Example 4-11. Compile Libraries Commands for Timing Simulation in VHDL

```
ncvhd1 -work lpm 220pack.vhd 220model.vhd ←  
ncvhd1 -work altera_mf altera_mf_components.vhd altera_mf.vhd ←  
ncvhd1 -work sgate sgate_pack.vhd sgate.vhd ←  
ncvhd1 -work stratixiigx stratixiigx_atoms.vhd \  
stratixiigx_components.vhd ←  
ncvhd1 -work stratixiigx_hssi stratixiigx_hssi_components.vhd \  
stratixiigx_hssi_atoms.vhd ←  
ncvhd1 -work work <alt2gxb>.vho ←  
ncelab work.<my design> -TIMESCALE 1ps/1ps -PULSE_R 0 -PULSE_INT_R 0 ←
```

To compile the libraries necessary for timing simulation of a Verilog HDL design targeting a Stratix II GX device, type the commands shown in [Example 4-12](#) at the IES command prompt.

Example 4-12. Compile Libraries Commands for Timing Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ←  
ncvlog -work altera_mf altera_mf.v ←  
ncvlog -work sgate sgate.v ←  
ncvlog -work stratixiigx stratixiigx_atoms.v ←  
ncvlog -work stratixiigx_hssi stratixiigx_hssi_atoms.v ←  
ncelab work.<my design> -TIMESCALE 1ps/1ps -PULSE_R 0 -PULSE_INT_R 0 ←
```

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. [Example 4-14](#) 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, edit your `cds.lib` file so that all of the libraries point to the work library, and compile the `stratixiv_hssi` model file into the `stratixiv_hssi` library.

When compiling the library files, you can safely ignore the following warning message:

```
"Multiple logical libraries mapped to a single location"
```

[Example 4-13](#) shows the `cds.lib` file.

Example 4-13. cds.lib File

```
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvhdl.lib
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvlog.lib
DEFINE work ./ncsim_work
DEFINE stratixiv_hssi ./ncsim_work
DEFINE stratixiv ./ncsim_work
DEFINE lpm ./ncsim_work
DEFINE sgate ./ncsim_work
```

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 VHDL design targeting a Stratix IV device, type the commands shown in [Example 4-14](#) at the IES command prompt.

Example 4-14. Compile Libraries Commands for Functional Simulation in VHDL

```
ncvhdl -work lpm 220pack.vhd 220model.vhd ←
ncvhdl -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
ncvhdl -work sgate sgate_pack.vhd sgate.vhd ←
ncvhdl -work stratixiv_hssi stratixiv_hssi_components.vhd \
stratixiv_hssi_atoms.vhd ←
ncvhdl -work work <altgx entity name>.vhd ←
ncelab work.<my design> ←
```

To compile the libraries necessary for a timing simulation of a Verilog HDL design targeting a Stratix IV device, type the commands shown in [Example 4-15](#) at the IES command prompt.

Example 4-15. Compile Libraries Commands for Gate-Level Timing Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ←
ncvlog -work altera_mf altera_mf.v ←
ncvlog -work sgate sgate.v ←
ncvlog -work stratixiv stratixiv_atoms.v ←
ncvlog -work stratixiv_hssi stratixiv_hssi_atoms.v ←
ncvlog -work work <altgx>.vo ←
ncelab work.<my design> -TIMESCALE lps/lps -PULSE_R 0 -PULSE_INT_R 0 ←
```

Gate-Level Timing Simulation for Stratix IV GX Devices

Stratix IV gate-level timing simulation is similar to Arria II gate-level timing simulation.

[Example 4-16](#) and [Example 4-17](#) show only the gate-level timing 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 post-fit timing simulation of your design that includes the ALTGX megafunction, edit your `cds.lib` file so that all of the libraries point to the work library and compile `stratixiv_atoms` and `stratixiv_hssi_atoms` into the `stratixiv` and `stratixiv_hssi` libraries, respectively. When compiling the library files, you can safely ignore the following warning message:

```
"Multiple logical libraries mapped to a single location"
```

For an example of a `cds.lib` file, refer to [Example 4-13 on page 4-14](#).

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 a timing simulation of a VHDL design targeting a Stratix IV device, type the commands shown in [Example 4-16](#) at the IES command prompt.

Example 4-16. Compile Libraries Commands for Gate-Level Timing Simulation in VHDL

```
ncvhdl -work lpm 220pack.vhd 220model.vhd ↵
ncvhdl -work altera_mf altera_mf_components.vhd altera_mf.vhd ↵
ncvhdl -work sgate sgate_pack.vhd sgate.vhd ↵
ncvhdl -work stratixiv stratixiv_atoms.vhd \
stratixiv_components.vhd ↵
ncvhdl -work stratixiv_hssi stratixiv_hssi_components.vhd \
stratixiv_hssi_atoms.vhd ↵
ncvhdl -work work <altgx>.vho ↵
ncsdfc <project name>_vhd.sdo ↵
ncelab work.<my design> -TIMESCALE lps/lps \
-SDF_CMD_FILE <SDF Command File> -PULSE_R 0 -PULSE_INT_R 0 ↵
```

To compile the libraries necessary for a timing simulation of a Verilog HDL design targeting a Stratix IV device, type the commands shown in [Example 4-17](#) at the IES command prompt.

Example 4-17. Compile Libraries Commands for Gate-Level Timing Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ↵
ncvlog -work altera_mf altera_mf.v ↵
ncvlog -work sgate sgate.v ↵
ncvlog -work stratixiv stratixiv_atoms.v ↵
ncvlog -work stratixiv_hssi stratixiv_hssi_atoms.v ↵
ncvlog -work work <altgx>.vo ↵
ncelab work.<my design> -TIMESCALE lps/lps -PULSE_R 0 -PULSE_INT_R 0 ↵
```

Functional Simulation for Stratix V 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 `stratiiv_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.

To compile the libraries necessary for a timing simulation of a Verilog HDL design targeting a Stratix V device, create the `cds.lib` file with contents as shown in [Example 4-18](#).

Example 4-18. Compile Libraries Commands for Functional Simulation in Verilog HDL

```
ncvlog -work lpm 220model.v ←
ncvlog -work altera_mf altera_mf.v ←
ncvlog -work sgate sgate.v ←
ncvlog -work stratixvgx stratixiigx_atoms.v ←
ncvlog -work stratixvgx_hssi stratixvgx_hssi_atoms.v ←
ncelab work.<my design> -TIMESCALE lps/lps -PULSE_R 0 -PULSE_INT_R 0 ←
```

[Example 4-19](#) shows the `cds.lib` file.

Example 4-19. cds.lib File

```
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvhdl.lib
SOFTINCLUDE ${CDS_INST_DIR}/tools/inca/files/cdsvlog.lib
DEFINE work ./ncsim_work
DEFINE stratixv_hssi ./ncsim_work
DEFINE stratixv ./ncsim_work
DEFINE lpm ./ncsim_work
DEFINE sgate ./ncsim_work
```

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 functional simulation of a VHDL design targeting a Stratix V device, create the `cds.lib` file with contents as shown in [Example 4-20](#).

Example 4-20. Compile Libraries Commands for Functional Simulation in VHDL

```
ncvhdl -work lpm 220pack.vhd 220model.vhd ←
ncvhdl -work altera_mf altera_mf_components.vhd altera_mf.vhd ←
ncvhdl -work sgate sgate_pack.vhd sgate.vhd ←
ncvhdl -work stratixv_hssi stratixv_hssi_components.vhd ←
ncvlog +v2k -work stratixv_hssi \
quartus/eda/sim_lib/cadence/stratixv_hssi_atoms_ncrypt.v ←
stratixv_hssi_atoms.vhd ←
ncvhdl -work work <my design>.vhd ←
ncelab work.<my design> ←
```

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 4-2 describes the pulse reject delay options.

Table 4-2. Pulse Reject Delay Options

Option	Description
-PULSE_R Option	Use this option when the pulses in your simulation are shorter than the delay within a gate-level primitive. The argument is the percentage of delay for pulse reject limit for the path.
-PULSE_INT_R Option	Use this option when the pulses in your simulation are shorter than the interconnect delay between gate-level primitives. The argument is the percentage of delay for pulse reject limit for the path.



The **-PULSE_R** and **-PULSE_INT_R** options are also used by default in the NativeLink feature for gate-level timing simulation.

To perform a gate-level timing simulation with the device family library, type the following IES software command:

```
nclab worklib.<project name>:entity -SDF_CMD_FILE <SDF Command File> \
-TIMESCALE 1ps/1ps -PULSE_R 0 -PULSE_INT_R 0 ↵
```

Using the NativeLink Feature with IES

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 IES 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

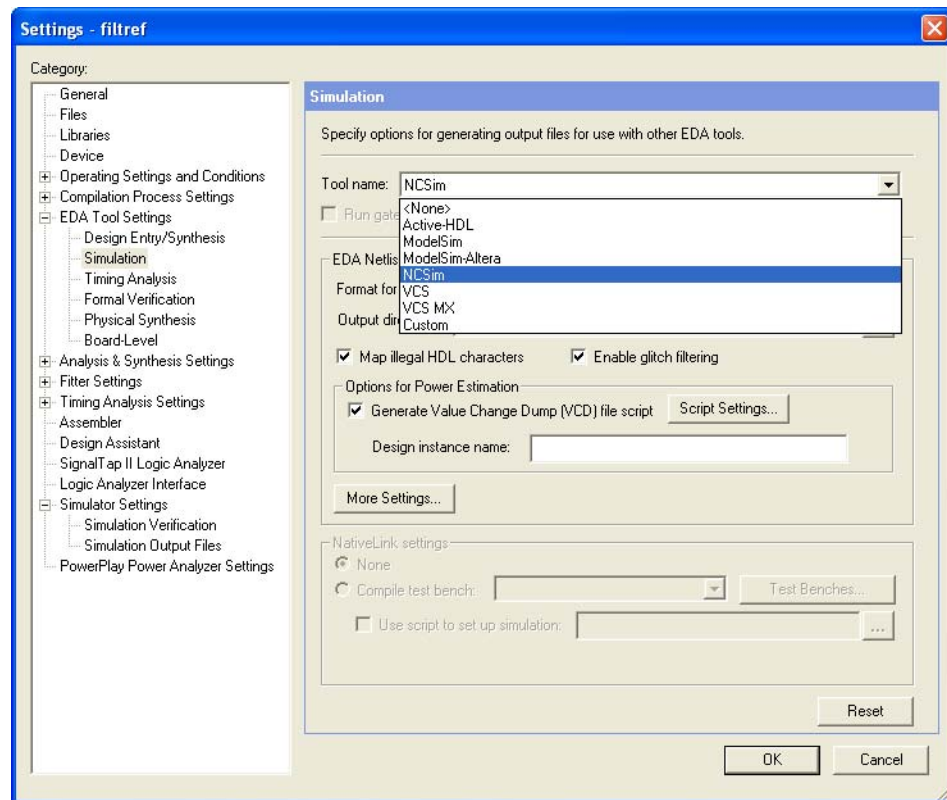
To generate a timing **.vcd** file for PowerPlay, you must first generate a VCD script in the Quartus II software and run the VCD script from the IES software to generate a timing **.vcd** file. This timing **.vcd** file can then be used by the PowerPlay Power Analyzer for power analysis. The following instructions show you how to generate a timing **.vcd** file.

To generate timing 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 (Figure 4-1).
2. In the **Category** list, click the “+” icon to expand **EDA Tool Settings**.
3. Click **Simulation**.
4. In the **Tool name** list, click **NC-Sim**.

- Turn on the **Generate Value Change Dump (VCD) File Script** option.

Figure 4-1. Simulation Settings Dialog Box




- Click **OK**.
- To generate the VCD script file, perform a full compilation.

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

- In the IES software, before simulating your design, source the `<revision_name>_dump_all_vcd_nodes.tcl` script. To source the **.tcl** script, use the **-input** switch while running the **nssim** command. For example:

```
ncsim -input <revision_name>_dump_all_vcd_nodes.tcl <my design>
```

- Continue to run the simulation until it finishes. Exit **ncsim** and the `<revision_name>.vcd` is generated in the simulation directory.

 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 .trn File

A **.trn** file is automatically generated when your simulation is done. The **.trn** file is not readable. It is used for generating the waveform view through SimVision.

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

- Type **simvision** on a command line. The **Design Browser** dialog box appears.

2. On the File menu, click **Open Database**. The **Open File** dialog box appears.
3. In the **Directories** field, browse to the directory that contains your **.trn** file.
4. Double-click the **.trn** file.
5. In the **Design Browser** dialog box, select the signals that you want to observe from the Hierarchy.
6. 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.

Scripting Support


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

For detailed information about scripting command options, refer to the Quartus II Command-Line and Tcl API Help browser.

To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp ←
```

 For more information, refer to *About Quartus II Scripting* in Quartus II Help.

 For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For information about all settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Manual*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

Generating IES Simulation Output Files


You can generate **.vo** and **.svo** files and **.sdo** simulation output files with Tcl commands or at a command prompt.

For more information about generating **.vo** and **.svo** simulation output files and **.sdo** file simulation output files, refer to “*Quartus II Simulation Output Files*” on page 4-6.

Tcl Commands

The following three assignments cause a Verilog HDL or SystemVerilog HDL netlist to be written out when you run the Quartus II netlist writer:

```
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT VERILOG -section_id  
eda_simulation  
set_global_assignment -name EDA_TIME_SCALE "1 ps" -section_id  
eda_simulation  
set_global_assignment -name EDA_SIMULATION_TOOL "NC-Verilog (Verilog)"
```

 For SystemVerilog HDL, the first assignment should be:

```
set_global_assignment -name EDA_OUTPUT_DATA_FORMAT "SYSTEMVERILOG  
HDL" -section_id
```

The netlist has a 1 ps timing resolution for the IES simulation software.

To run the Quartus II Netlist Writer, type the following Tcl command

```
execute_module -tool eda ←
```

Command Prompt

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

```
quartus_eda <project name> --simulation --format=<verilog/vhdl> \  
--tool=ncsim ←
```

Conclusion

The Cadence IES family of simulators work within an Altera FPGA design flow to perform functional, post-synthesis, and gate-level timing simulation, easily and accurately.

Altera provides functional models of LPM and Altera-specific megafunctions that you can compile with your testbench or design. For timing simulation, use the atom netlist file generated by Quartus II compilation.

The seamless integration of the Quartus II software and Cadence IES tools make this simulation flow an ideal method for fully verifying an FPGA design.

Document Revision History


Table 4–3 shows the revision history for this chapter.


Table 4–3. Document Revision History (Part 1 of 2)

Date	Version	Changes
November 2011	11.0.1	Template update. Minor editorial updates.
May 2011	11.0.0	<ul style="list-style-type: none"> ■ Changed chapter title ■ Linked to Help for Stratix V Libraries ■ Added SystemVerilog HDL information ■ Other minor changes 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 Help where appropriate ■ Minor text edits ■ 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

Table 4-3. Document Revision History (Part 2 of 2)

Date	Version	Changes
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 4–5 ■ Added “Generate Simulation Script from EDA Netlist Writer” on page 4–35 ■ Added “Viewing a Waveform from a .trn File” on page 4–36 ■ Minor editorial updates
November 2008	8.1.0	<ul style="list-style-type: none"> ■ Added “Compile Libraries Using the Altera Simulation Library Compiler” on page 4–5. ■ Added information about the <code>--simlib_comp</code> utility. ■ Minor editorial updates. ■ Updated entire chapter using 8½” × 11” chapter template.
May 2008	8.0.0.	<ul style="list-style-type: none"> ■ Updated Table 4–1. ■ Updated Figure 4–1. ■ Updated “Compilation in Command-Line Mode” on page 4–9. ■ Updated “Generating a Timing Netlist with Different Timing Models” on page 4–18. ■ Added “Disable Timing Violation on Registers” on page 4–20. ■ Updated “Simulating Designs that Include Transceivers” on page 4–23. ■ Updated “Performing a Gate Level Simulation Using NativeLink” on page 4–30. ■ Added “Generating a Timing VCD File for PowerPlay” on page 4–33. ■ Added hyperlinks to referenced documents throughout the chapter. ■ 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.

