Verilog
Simulation Guide

Table of Contents

Introduction

The Verilog Simulation Guide contains information about interfacing the FPGA development software with Verilog simulation tools. Refer to the online help for additional information about using the Libero SoC software. This document is intended for use with Libero SoC software v10.0 and above. For earlier software versions, see the Legacy Verilog Simulation Guide.

Refer to the documentation included with your Verilog simulation tool for information about performing simulation.

Document Assumptions

This document assumes the following:

1. If you are using a PC, you have installed the Libero SoC software in the “c:\microsemi” directory.
2. You have installed a VERILOG simulator.
3. You are familiar with UNIX workstations and operating systems.
4. You are familiar with PCs and Windows operating environments.
5. You are familiar with FPGA architecture and FPGA design software.

Document Conventions

This document uses the following variables:

FPGA family libraries are shown as <act_fam>. Substitute the desired FPGA family variable with your family, as needed. For example:

```
vlog -work <act_fam> <act_fam>.v
```

Online Help

Microsemi SoC software comes with online help. Online help specific to each software tool is available from the Help menu.

1 – Setup

This chapter contains information about the directory structure of the Verilog libraries and setting up Verilog libraries for use in simulating Libero SoC designs. Refer to the documentation included with your Verilog simulator for information about setting up your simulation tool.

Software Requirements

The information in this guide applies to the Microsemi Libero SoC Software v10.0 and above and IEEE-1364-compliant Verilog simulators. Additionally, this guides contains information about using the Windows and UNIX ModelSim simulators.

For specific information about which versions this release supports, go to the automated technical support system on the Microsemi website (http://www.actel.com/custsup/search.html) and type the following in the Keyword box:

```
third party
```

If you are using HP-UX, you must also set the following variable:

```
setenv SHLIB_PATH $ALSDIR/lib
```

Refer to the Libero SoC online help and the documentation included with your simulation tool for additional information about setting environment variables.

Compiling Verilog Libraries

Before simulating a design with the ModelSim Verilog simulator, you must compile the Microsemi Verilog libraries. This section describes the procedures. Refer to the documentation included with your simulation tool for additional information about compiling libraries.

ModelSim

Use the following procedure to compile Verilog libraries for the ModelSim simulator. Type UNIX commands at the UNIX prompt. The commands below are for Windows. To compile the Verilog libraries:

Since the installation path varies for each user and each installation, this document uses $ALSDIR to indicate the location where the software is installed. If you are a Unix user, simply create an environment variable called ALSDIR and set its value to the installation path. If you are a Windows user, replace $ALSDIR with the installation path in the commands.

1. Create a directory called **mti** in the directory `$ALSDIR\lib\vlog`
2. Invoke the simulator (Windows only).
3. Change to the $ALSDIR\lib\vlog\mti directory. Type the following command at the prompt:
   ```
   cd $ALSDIR\lib\vlog\mti
   ```
4. Create an <act_fam> family library directory for the simulator. Type the following command:
   ```
   vlib <act_fam>
   ```
5. Compile the library. Type the command at the prompt:
   ```
   vlog -work <act_fam> $ALSDIR\lib\vlog\<act_fam>.v
   ```
6. (Optional) Compile the Migration library. Only perform this step if you are using the migration library. Type the following command:
   ```
   vlog -work <act_fam> $ALSDIR\lib\vlog\<act_fam>_mig.v
   ```

2 – Design Flow

This chapter describes the design flow for creating designs using Verilog simulation and the Libero SoC software.

Verilog Design Flow Overview

The Verilog design flow has four main steps:

1. Create Design
2. Implement Design
3. Programming
4. System Verification

The following sections detail these steps.

Create Design

During design creation/verification, a design is captured as a schematic or as an RTL-level (behavioral) Verilog HDL source file.

If your design is a Verilog HDL source file, you can perform a behavioral simulation to verify that the HDL code is correct. The code is then synthesized into an gate-level (structural) Verilog HDL netlist. After synthesis, you can perform a structural simulation of the design. Finally, you use the EDIF netlist generated in Libero SoC and a structural Verilog netlist for structural and timing simulation.

If your design is a schematic, you generate an EDIF netlist for use in Libero SoC and a structural Verilog netlist for structural and timing simulation. You do not perform behavioral simulation or synthesis.

Design Capture

Enter your schematic using a third-party schematic-capture tool or create your Verilog HDL source file using a text editor or a context-sensitive HDL editor. Your Verilog HDL design source can contain RTL-level constructs, as well as instantiations of structural elements, such as cores from the Libero SoC Catalog. Refer to the documentation included with your design-capture tool for information about design capture.

Behavioral Simulation

Perform a behavioral simulation of your design before synthesis. Behavioral simulation verifies the functionality of your Verilog HDL code. You can use a standard Verilog HDL testbench to drive simulation. Refer to "Behavioral Simulation" on page 9 and the documentation included with your simulation tool for information about performing functional simulation.

Synthesis

After you have created your Verilog HDL source file, you must synthesize it before place-and-route. Synthesis transforms the Verilog HDL source file into a gate-level netlist and optimizes the design for a target technology. Refer to the documentation included with your synthesis tool for information about performing design synthesis.

EDIF Netlist Generation

After you have created, synthesized (if your design is an HDL source file), and verified your design, you must generate an EDIF netlist or import verilog netlists for place-and-route in Libero SoC. If your design is a Verilog HDL source file, use the EDIF netlist to generate a structural Verilog netlist. Refer to "Generating an EDIF Netlist" on page 7 and the documentation included with your schematic-capture or synthesis tool for information about generating an EDIF netlist.

Structural Verilog Netlist Generation

A structural Verilog netlist is generated automatically when you place-and-route your project in Libero SoC. Refer to "Generating a Structural Verilog Netlist" on page 7 for information about generating a structural netlist.

Structural Simulation

Perform a structural simulation of your design before placing-and-routing it. Structural simulation verifies the functionality of your structural Verilog netlist. Use default unit delays included in the Verilog libraries for every gate. Refer to "Structural Simulation" on page 9 and the documentation included with your simulation tool for information about performing structural simulation.

Implement Design

During design implementation, you place-and-route a design using Libero SoC. Additionally, you can perform static-timing analysis on a design with SmartTime. After place-and-route, you can perform postlayout (timing) simulation with a Verilog simulator.

Timing Simulation

Perform a timing simulation of your design after place-and-route. Timing simulation requires information extracted from software, which overrides default unit delays in the Verilog libraries. Refer to "Timing Simulation" on page 10 and the documentation included with your simulation tool for information about performing timing simulation.

Programming

Program a device with programming software and hardware from Microsemi SoC or a supported third-party programming system. Refer to the programmer online help for information about programming a Microsemi SoC device.

System Verification

You can perform system verification on a programmed device using the Silicon Explorer diagnostic tool. Refer to the *Silicon Explorer Quick Start* for information about using the Silicon Explorer.

3 – Generating Netlists

This chapter describes the procedures for generating EDIF and structural Verilog netlists.

Generating an EDIF Netlist

After capturing your schematic or synthesizing your design, generate an EDIF netlist from your schematic-capture or synthesis tool. You can use the EDIF netlist for place-and-route in Libero SoC. Refer to the documentation included with your schematic-capture or synthesis tool for information about generating an EDIF netlist.

Make sure to specify Verilog for the naming style when importing the EDIF netlist into Libero SoC.

Generating a Structural Verilog Netlist

Structural Verilog netlist files are generated automatically as part of your Libero SoC project.

You can find your Verilog netlist files in the /synthesis directory of your Libero project. For example, if your project directory is named project1, then your netlist files are in /project1/synthesis.

Some families enable you to export these files manually for use in external tools. If your device supports this feature you can export netlist files from **Tools > Export > Netlist**.

4 – Interpreted Simulation

This chapter describes the procedures for performing functional (behavioral and structural) and timing simulation on your design by using tools that either interpret library and design files or compile them on-the-fly. Simucad SilosIII and Synopsys VCS are simulators in this category.

This chapter includes information about creating a testbench and information about creating a command file to run a simulation in batch mode. Also, this chapter includes a description of some common Verilog simulation switches. Refer to the documentation included with your simulation tool for additional information about testbenches, command files, switches, and simulation.

Example Testbench

You can use a testbench to apply test vectors or patterns to a design during simulation to compare input and output patterns. The file can instantiate the top-level design, using a Verilog-predefined command, such as $readmemb, $monitor, and $display. To use the testbench, it must be in the current project directory. The following is an example testbench:

```
'timescale 1ns/100ps
module test;
//Inputs and outputs declaration
wire ............
reg ..................
//Instantiate the top module of your design in the test module
<top_module> <instance_name> (.....Pin List...);
...........
//stimulus patterns
initial
begin
.....
end
endmodule
```

Example Command File

You can use a command file for batch simulation. Your command file should include all command variables and Verilog switches you want to set during simulation. The following is an example command file:

```
<test_bench>.v
<design_name>.v
<verilog_switch_1> ... <verilog_switch_n>
```

The <design_name>.v variable is the Verilog top-level design that includes all design sublevels. This variable can represent a behavioral Verilog design for function simulation or gate-level Verilog design for structural or timing simulation.

The <verilog_switch_1> and <verilog_switch_n> variables represent Verilog switches that you can add to your command line during simulation. Refer to "Verilog Switches" on page 11 for information about available Verilog switches.

Behavioral Simulation

Use the following procedure to perform a behavioral simulation of a design. Refer to the documentation included with your simulation tool for additional information about performing behavioral simulation.

1. Create or modify the testbench. Make sure your testbench has a timescale definition added to it. The following is an example timescale definition:

   ```
   'timescale 1ns/100ps
   ```

   Refer to "Example Testbench" on page 8 for information about creating testbenches.
2. Create or modify the command file. A command file is only necessary if you are running batch simulation. Refer to "Example Command File" on page 8 for information.
3. Simulate the design. If your design is a Verilog HDL design, make sure that you simulate your behavioral Verilog HDL source file. Invoke the Verilog simulator by typing the following command:

   ```
   <verilog_executable> <test_bench>.v <design_name>.v -v
   $ALSDIR/lib/vlog/<act_fam>/<act_fam>.v
   ```

   If you are using the migration libraries, type the following command:

   ```
   <verilog_executable> <test_bench>.v <design_name>.v -v
   $ALSDIR/lib/vlog/<act_fam>.v -v $ALSDIR/lib/vlog/<act_fam>_mig.v
   ```

   The “-v” option is a Verilog switch that you can add to your command line during simulation. Refer to "Verilog Switches" on page 11 for information.

To simulate a design using a command file:

invoke the Verilog simulator by typing the following command:

```
<verilog_executable> -f <command_file>
```

The “-f” switch is necessary if you are using a command file to simulate a design.

Structural Simulation

Use the following procedure to perform a structural simulation on a design. Refer to the documentation included with your simulation tool for additional information about performing structural simulation.

1. Create or modify the testbench. Make sure your testbench has a timescale definition added to it. The following is an example timescale definition:

   ```
   'timescale 1ns/100ps
   ```

   Refer to "Example Testbench" on page 8 for information about creating testbenches.
2. Create or modify the command file. A command file is only necessary if you are running batch simulation. Refer to "Example Command File" on page 8 for information.
3. Simulate the design. If your design is a Verilog HDL design, make sure that you simulate the structural Verilog HDL netlist that you generated using Libero SoC. Invoke the Verilog simulator by typing the following command:

   ```
   <verilog_executable> <test_bench>.v <design_name>.v -v
   $ALSDIR/lib/vlog/<act_fam>.v
   ```

   If you are using the migrations libraries, type the following command:

   ```
   <verilog_executable> <test_bench>.v <design_name>.v -v
   $ALSDIR/lib/vlog/<act_fam> -v $ALSDIR/lib/vlog/<act_fam>_mig.v
   ```

   The “-v” option is a Verilog switch that you can add to your command line during simulation. Refer to "Verilog Switches" on page 11 for information.

To simulate a design using a command file:

invoke the Verilog simulator by typing the following command:

```
<verilog_executable> -f <command_file>
```

The -f switch is necessary if you are using a command file to simulate a design.

Timing Simulation

Use the following procedure to perform a timing simulation on a design. Refer to the documentation included with your simulation tool for additional information about performing timing simulation.

1. Place-and-route your design in Libero SoC. Refer to the software online help for information about placing-and-routing a design.

   Timing information files are available from Libero SoC after you place-and-route and backannotate your Libero SoC project. Some device families enable you to export back annotated data directly from the Tools menu. Use **Tools > Export > Back Annotated Data**. Create a <design_name>.sdf file by specifying SDF as the CAE type. Click OK.

2. Create or modify a testbench. Make sure your testbench has an $sdf_annotate construct in it. The following is an example testbench with such construct:

   ```
   'timescale 1ns/100ps
   module test;

   //Inputs and outputs declaration
   wire ............
   reg ..................

   //Instantiate the top module of your design in the test module
   <top_module> <instance_name> (.....Pin List...);
   ...........

   //stimulus patterns
   initial
   begin
   .....
   end

   //Invoke SDF routine to back annotate
   initial
   $sdf_annotate(“<design_name>.sdf”,<instance_name>);
   endmodule
   ```

   The <instance_name> variable is the top-level instance name.

   Refer to "Example Testbench" on page 8 for information about creating testbenches.

   If your design contains a PLL, you must use a 1ps timescale resolution, for example, ‘timescale 1ns/ps.

3. Create or modify the command file. A command file is only necessary if you are running batch simulation. Refer to "Example Command File" on page 8 for information about creating command files.

4. (VCS Only) Create a PLI table. The PLI table is a text file that contains PLI commands for VCS. The following is an example PLI table called “sdf.tab” that uses the module “test” in the testbench example

   ```
   $sdf_annotate call=sdf_annotate_call acc+=tchk,mp,mip,prx:test+
   ```

   Refer to the VCS documentation for information about creating a PLI table.

5. Simulate the design. If your design is a Verilog HDL design, make sure that you simulate the structural Verilog HDL netlist that you generated using Libero SoC. Invoke the Verilog simulator by typing the following command (for VCS):

   ```
   vcs <test_bench>.v <design_name>.v -v $ALSDIR/lib/vlog/<act_fam>.v -M -P sdf.tab
   ```

   If you are using the migrations libraries, type the following command:

   ```
   vcs <test_bench>.v <design_name>.v -v $ALSDIR/lib/vlog/<act_fam>.v -v
   $ALSDIR/lib/vlog/<act_fam>_mig.v -M -P sdf.tab
   ```

   For other simulators:

   ```
   <verilog_executable> <test_bench>.v <design_name>.v -v
   $ALSDIR/lib/vlog/<act_fam>.v
   ```

   If you are using the migrations libraries, type the following command:

```
<verilog_executable> <test_bench>.v <design_name>.v -v
$ALSDIR/lib/vlog/<act_fam>.v -v $ALSDIR/lib/vlog/<act_fam>_mig.v
```

The -v option is a Verilog switch that you can add to your command line during simulation. Refer to "Verilog Switches" on page 11 for information.

To simulate a design using a command file:

Invoke the Verilog simulator by typing the following command:

```
<verilog_executable> -f <command_file>
```

The -f switch is necessary if you are using a command file to simulate a design.

Verilog Switches

This section defines and gives usage examples of some common Verilog switches for simulators that interpret design files or compile the files on-the-fly. Refer to the documentation included with your Verilog simulation tool for additional information about using switches during simulation.

Minus Switches

Table 4-1 defines and gives usage examples of Verilog minus switches.

Table 4-1 • **Minus Switches**

Switch	Definition
-s	Stop option; initiates entry into interactive mode after successful design compilation.
-a	Accelerated option; directs accelerated, declared elements to simulate in accelerated mode (Verilog-XL only).
-c	Compile only option; compiles the text code in a data file and exits the simulation mode.
-d	Decompile option; retargets data files into existing text files.
-f	Command argument file option; reads invocation command from a text file.
-l	Log file option.
-y	Library directory option; specifies a target library directory.
-v	Specifies the library file.

Plus Switches

Table 4-2 defines and gives usage examples of Verilog plus switches.

***Table 4-2* • Plus Switches**

Switch	Definition
+libext+	Used with -y switch.
+delay_mode_path	Specifies the path delay model for simulation.
+delay_mode_unit	Specifies the unit delay model for simulation.
+delay_mode_zero	Functional simulation option; specifies the zero delay model for simulation.
+mindelays	Back-annotation option; selects minimum delay for simulation.
+maxdelays	Back-annotation option; selects maximum delay for simulation.
+typdelays	Back-annotation option; selects typical delay for simulation.
+transport_int_delays	Considers the interconnect delays as transport delays instead of inertial; required for IGLOO and ProASIC3 libraries.

5 – Simulation with ModelSim

This chapter describes the procedures for performing functional (behavioral and structural) and timing simulation on a Libero SoC design using commands for the ModelSim simulator. Refer to the documentation included with your simulation tool for information about simulating a design using the graphical user interface.

Behavioral Simulation

Use the following procedure to perform a behavioral simulation of a design using the ModelSim simulator. Type UNIX commands at the UNIX prompt. Type PC commands on the command line of the ModelSim Transcript window. The commands below are for PC. To make the commands work for UNIX, use forward slashes instead of back slashes.

1. Invoke the simulator (PC only).
2. Change the directory to your project directory. This directory must include your Verilog design files and testbench. Type the following command:

   ```
   cd <project_dir>
   ```
3. Create a work directory. Type the following command:

   ```
   vlib work
   vmap work ./work
   ```
4. Compile your design source and testbench file(s). Before simulating your design, you must compile the source files and testbench. For hierarchical designs, compile the lower-level design blocks before the higher-level design blocks. Type the following commands:

   ```
   vlog <behavioral_design_file>.v
   vlog <test_bench>.v
   ```
5. Simulate your design. Type the following command:

   ```
   vsim <topmost_module_name>
   ```

   For example:

   ```
   vsim test_adder_behave
   ```

   The module test_adder_behave in the testbench will be simulated.

   If any cores are instantiated in your Verilog source, use the following command to simulate your design with the compiled Verilog library.

   ```
   vsim -L $ALSDIR\lib\vlog\mti\<act_fam> <topmost_module_name>
   ```

Structural Simulation

Use the following procedure to perform a structural simulation of a design using the ModelSim simulator. Type UNIX commands at the UNIX prompt. Type PC commands on the command line of the ModelSim Transcript window. The commands below are for PC. To make the commands work for UNIX, use forward slashes instead of back slashes.

1. Invoke the simulator (PC only).
2. Change directory to your project directory. This directory must include your Verilog design files and testbench. Type the following command:

   ```
   cd <project_dir>
   ```
3. Create a “work” directory. You only need to create a work directory if you are using a different project directory than the one you used for behavioral simulation. Type the following command:

   ```
   vlib work
   ```

4. Compile the structural netlist and testbench. If you have not already generated a structural Verilog netlist, go to "Generating a Structural Verilog Netlist" on page 7 for the procedure. Type the following commands:

   ```
   vlog <structural_netlist>.v
   vlog <test_bench>.v
   ```

5. Simulate your design. Type the following commands:

   ```
   vsim -L $ALSDIR\lib\vlog\mti\<act_fam> <topmost_module_name>
   ```

   For example:

   ```
   vsim -L $ALSDIR\lib\vlog\mti\42mx test_adder_structure
   ```

   The module test_adder_structure in the testbench will be simulated using the compiled 42MX Verilog library.

Timing Simulation

Use the following procedure to perform a timing simulation of a design using the ModelSim simulator. Type UNIX commands at the UNIX prompt. Type PC commands on the command line of the ModelSim Transcript window. The commands below are for PC. To make the commands work for UNIX, use forward slashes instead of back slashes.

1. Place-and-route your design in Libero SoC. Refer to the software online help for information about placing-and-routing a design.
2. Timing information files are available from Libero SoC after you place-and-route and backannotate your Libero SoC project. Some device families enable you to export back annotated data directly from the Tools menu. Use **Tools > Export > Back Annotated Data**. Create a <design_name>.sdf file by specifying SDF as the CAE type. Click OK.
3. Invoke the simulator (PC only).
4. Change the directory to your project directory. This directory must include your Verilog design files and testbench. Type the following command:

   ```
   cd <project_dir>
   ```

5. Create a work directory. You only need to create a work directory if you are using a different project directory than the one you used for behavioral and structural simulation. Type the following command:

   ```
   vlib work
   ```

6. Compile the structural netlist and testbench. If you have not already generated a structural Verilog netlist, go to "Generating a Structural Verilog Netlist" on page 7 for the procedure. Type the following commands:

   ```
   vlog <structural_netlist>.v
   vlog <test_bench>.v
   ```

7. Simulate your design using timing information contained in the SDF file. Type the following command:

   ```
   vsim -L $ALSDIR\lib\vlog\mti\<act_fam> -sdf[max|typ|min]
   /<region>=<design name>.sdf -c <topmost_module_name>
   ```

   The <region> option specifies the region (or path) to an instance in a design where back annotation begins. You can use it to specify a particular FPGA instance in a larger system design or testbench that you wish to back annotate. For example:

   ```
   vsim -L $ALSDIR\lib\vlog\mti\42mx -sdfmax /uut=adder.sdf -c
   test_adder_structural
   ```

   In this example, the module adder has been instantiated as instance uut in the testbench. The module named test_adder_structural in the testbench will be simulated using the maximum delays specified in the SDF file.

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world, **650.318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

For Microsemi SoC Products Support, visit http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office.

Visit About Us for sales office listings and corporate contacts.

Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.