Version 1.2 16/Aug/2004

1 Introduction

ModelSim is a useful tool that allows you to stimulate the inputs of your modules and view both outputs and internal signals. It allows you to do both behavioural and timing simulation, however, this document will focus on behavioural simulation. Keep in mind that these simulations are based on models and thus the results are only as accurate as the constituent models.

Note that simulations involving the ZBT RAM will require using the ZBT behavioural model from the ECE532 course webpage. (This must be compiled in ModelSim as part of the syntax is not supported by XST.)

2 Using ModelSim On the Lab Machines

The ModelSim executable can be found at /nfs/ugsparcs/thesis1/thesis1/modelsim-5.7f/modeltech/win32/ modelsim.exe on the UGSPARC machines. However, you can launch it from within ISE and EDK on the UGSPARC PC's.

3 Using ModelSim at home

You can download MXE II, the free version of ModelSim for Xilinx from http://support.xilinx.com/xlnx /xil_tt_product.jsp?sProduct=MXE+II

Be sure to download it for the language that you intend to use to code your modules (either VHDL or Verilog).

To do mixed language VHDL/Verilog simulations, you will have to connect to school and use the version of ModelSim on the UGSPARCs.

4 ISE

Check that your ISE preferences have been set to include the path to ModelSim on the UGSPARC PC. Go to Edit > Preferences. Select the Integrated Tools tab. If the path has not yet been specified, browse to the location of the ModelSim executable for the Model Tech Simulator entry.

1. Code up your design in HDL. You might include Xilinx primitives, CoreGen modules, etc.
2. Create a testbench for the module/sub-module that you want to simulate. The testbench is an HDL file that drives the inputs at specified times and is essentially your "test case". In ISE you can use HDL bencher to generate one easily.
   - Go to Project > New Source.
   - Select "Test Bench Waveform" and enter a name for it in the File Name box. For easy reference, a good name is test_<name-of-your-module-to-be-tested>.
   - Click Next.
   - Select the source file that corresponds to the top-level file for the module you want to test.
   - Click Next. Click Finish.
   - An Inintialize Timing dialog box will pop up.

- In the Design Type box specify if you have 1 or more clocks. If 1 clock, make sure the correct signal is chosen.
- If you have just 1 clock, specify the clock timing.
- Click OK if 1 clock, otherwise click Next.
- If you have multiple clocks, select the clocks from the list of signals. Click Next. Then associate every remaining signal with a clock. Click Next. Specify the Clock Timing for each clock. Click Finish.
- In HDL bencher, specify your input pattern for all the input signals (blue). (Make sure you are using the correct radix! The radix toolbar buttons are 16=> Hex, 10=> Decimal, 2=> Binary.)
- Scroll horizontally in time to where you want your test to end. Right click on the column and select Set End of Testbench.
- Save your .tbw file. This will generate a .tfw file for Verilog or a .vhw file for VHDL.

3. In the Sources in Project view, click on the .tbw file you just created. In the Processes for Source view, expand the ModelSim Simulator toolbox. You will find the first 2 items useful: Simulate Behavioral Model and Generate Expected Simulation Results.
4. Right Click on Simulate Behavioral Model and select Run. This will generate an .fdo file and a .udo file and launch ModelSim. In ModelSim, it will run the .fdo script. Open the .fdo script to see the ModelSim commands that are called (in sequence).

5 Using ModelSim itself

While there are many things you can do with ModelSim, these are the basic things you will need to get started.

If you did not launch ModelSim from ISE or EDK, follow these directions based on the "A Quicker Start" document by Yan Kiu Chan to compile the HDL:

- If you are logged in on the UGSPARC workstations, set up the necessary environment variables by doing:

```
% source /thesis1/modelsim-5.7f.SUN/CSHRC
```

- Change directory to your project directory, which should include your design files and test bench. Then invoke Modelsim:

```
% vsim
```

- Type the following command to create a working directory called "work". This is where Modelsim will compile your design to (GUI: menu File -> New -> Directory):

```
% vlib work
```

- Before simulating your design, you need to compile the source files and test bench. For hierarchical designs, compile the lower level design blocks before the higher level design blocks. To compile, (GUI: menu Compile -> Compile) type the following commands:

```
% vlog <design_file>.v
% vcom <design_file2>.vhd
% vlog <testbench>.v
```

- To simulate your design, type the following command:

```
% vsim <topmost_module_name>
```

- For example, if your design has topmost module named "top" (GUI: menu Simulate -> Simulate, select the topmost module name from the "Design" tab):

```
% vsim top
```

- If this is a post-synthesis simulation or any Xilinx core macros are instantiated in your Verilog source code, use the following command to simulate your design with the Xilinx Verilog Core Library (GUI: add the library to reference in the "Libraries" tab):

```
% vsim -L /thesis1/modelsim-5.7f/modeltech/xilinx_libs/XilinxCoreLib_ver <topmost_module_name>
```

If you launched ModelSim from ISE, the generated .fdo file will open the Wave window, the Signal window, and the Structure window. Otherwise, you can open them yourself from the main GUI by going to the View menu.

In the Structure window, you can expand the module's hierarchy. The signals in the Signal window will correspond to the level selected in the Structure window. Expanding and selecting a level in the main ModelSim window Workspace "sim" tab has the same effect. The signals can then be dragged and dropped from the Signals window into the Wave window for viewing.

If you are debugging, you will probably wish to use the same set of signals every time you simulate this module. You can save the format of the signals, radix, dividers, and labels by selecting File > Save Format in the Wave window. This will save the format (not the simulation data) to a .do file. Sometimes you may find it useful to modify the .do file by hand instead of manipulating signal names from the Wave window GUI.

Once the signals are in the Wave window, you can Restart the simulation by clicking on the 6th toolbar button from the right. You can then run the testbench by clicking on the Run -All button on the Wave window toolbar (3rd from the right). Alternately, you could type "run -all" at the ModelSim command prompt in the main ModelSim window to run the testbench.

As you make modifications to your HDL during debugging, you will have to re-compile it within ModelSim before re-simulating. If you have an .fdo script, you can simply type "do <filename>.fdo" to recompile and launch all the appropriate windows. Note that, for simulation purposes, re-synthesizing your HDL in the Xilinx tools will have no impact besides helping to find syntax errors.

5.1 Troubleshooting:

In the ISE Sources in Project window, right click on the device (e.g. xc2v2000-4ff896) and select Properties. Make sure that your project properties list the correct HDL for the generated simulation language. If the top-level file of your design is in Verilog, you should specify to generate Verilog. If it is in VHDL, you should specify VHDL. A mismatch will cause weird errors to be generated.

6 EDK

For instructions on using SimGen from within EDK to do system-level simulation, please refer to Section 5 of "Integrating a Verilog design into a MicroBlaze System" by Lorne Applebaum on the ECE532 course webpage.

7 Handy Xilinx Answer Records for ModelSim

2561: How do I compile the Xilinx Simulation Libraries for ModelSim?
10176: "Error: Cannot open library unisim at unisim." (VHDL, Verilog)

12491: How do I save the position of the ModelSim windows?
18016: Does MXE support mixed language VHDL and Verilog simulations? (Note that ModelSim SE/PE do)
18226: Advanced tips for using ModelSim with Project Navigator

Look At Next

Module 9: Using and Modelling OPB Interfaces
Module 12: Using ChipScope

Revision History

1.0 04 Jul 2004 Initial Version - Nathalie Chan King Choy
1.1 12 Jul 2004 Added "A Quicker Start" steps - Nathalie Chan King Choy
1.2 16 Aug 2004 Added Look at Next section - Nathalie Chan King Choy