

ModelSim Installation & Tutorial

Greg Gibeling
UC Berkeley
gdgib@eecs.berkeley.edu

October 13, 2008

1 Introduction

In this document I will cover the basics of installing ModelSim (see section 2), compiling the Xilinx simulation libraries, and simulating a simple example project (see section 3). This document is targeted at both RDL users and, more generally, those new to the simulation of digital logic (EECS150 and CS61C students).

EECS150 and CS61C users needn't worry about installation, as the relevant Berkeley computers (those in 125 Cory for EECS150 and ilinux1, 2 & 3 for CS61C) already have ModelSim properly installed. Students may wish to read the installation section if they are interested in running ModelSim on their home computers. CS61C students, like all people not using Xilinx FPGAs, should ignore the instructions in this document which mention "Xilinx". In general, RDL users will need section 2, but should refer to the CounterExample, distributed with RDLC for use. Section 3 on the other hand is meant for those without ModelSim experience, or those who need a refresher, in particular U.C. Berkeley students in CS61C or EECS150.

Note that as of this writing the current versions of various pieces of software mentioned in this document are: RDLC 2.2007.6.7 and ModelSim SE 6.2g.

2 Installation & Setup

Make sure you have a license.

Step1. Download the installer from Mentor Graphs (<http://www.model.com/>).

- i. You probably want SE (which is not the same as the student edition), assuming you, your company or school has purchased a license.
- ii. Students should consider the http://www.model.com/resources/student_edition/download.asp, which has a limited time license. Otherwise you may wish to use the Xilinx Edition or Altera Edition of ModelSim which are both limited to simulating smaller designs.

Step2. Run the installer.

- i. You will want the full, rather than the evaluation version, assuming you already have a license.
- ii. The installer should create a directory `c:\Modeltech_version`.

Step3. Install the license by setting `MGLS_LICENSE_FILE` or running the licensing wizard.

- i. The former is recommended, if you have a floating license on a server somewhere.
- ii. The license for ModelSim should be of the form `1717@somehost.domain`.
- iii. If you have an actual `license.dat` file, simply set `MGLS_LICENSE_FILE` to the absolute path to `license.dat`.

Step4. Build the simulation libraries

- i. Who needs to do this?
 - *. Not necessary unless you intend to use ModelSim to simulate designs for Xilinx FPGAs.
 - †. RDLC2 as of v2.2007.3.26 will require this.
- ii. Make `c:\Modeltech_version\ModelSim.ini` writable (it is marked read-only by the installer).
- iii. Run `compplib` to compile the Xilinx libraries. Note that it is best to run from `c:\Modeltech_version`.
 - *. This will modify `ModelSim.ini` in the current directory to include paths to the Xilinx libraries.

Step5. Define the `MODELSIM` compiler flag for use with `Const.v`.

- i. Who needs to do this?
 - *. Students in CS61C and EECS150 may skip this step as ModelSim has been installed to do this by default.
 - †. Anyone else not using Verilog files from RDLC2, CS61C or EECS150 may also skip this step.
- ii. Open `c:\Modeltech_version\ModelSim.ini` in a text editor (make sure it is still writable).
- iii. Add the line `OptionFile = C:\Modeltech_version\vlog.opt` under the `[vlog]` header.
- iv. Create the file `C:\Modeltech_version\vlog.opt`.
- v. Add the line `+define+MODELSIM` to `vlog.opt`.
- vi. This will ensure that the preprocessor flags in `Const.v` are properly defined for simulation.

Step6. This should complete your installation of ModelSim.

Step7. You may test your installation by attempting to simulate, as described below, a simple project with a single Verilog file shown in program 1

Program 1 TestVerilog.v

```

1 module TestVerilog;
2     initial begin
3         $display('Hello, World!');
4     end
5 endmodule

```

3 Use

The purpose of an HDL simulator is to compile, and then simulate an HDL (hardware description language: Verilog and VHDL are examples) on a standard computer. While this is very slow compared to a real circuit implementation, it allows complete visibility and can be much less expensive, making it ideal for design and debugging. Note that as a circuit grows in complexity an FPGA will generally be a better platform, as the simulator will start to degrade in performance, and has no true IO connections.

ModelSim is a very powerful HDL simulation environment, and as such can be difficult to master. To correctly simulate many complex test benches, you will need to create and use a ModelSim project manually. Note that throughout this tutorial we assume you are attempting to simulate a purely Verilog based design. The steps are fairly simple:

Step1. Create a directory for your project (section 3.1).

Step2. Start ModelSim and create a new project (section 3.2).

- Step3.** Add all your Verilog files to the project (section 3.3).
- Step4.** Compile your Verilog files (section 3.4).
- Step5.** Start the simulation (section 3.5).
- Step6.** Add signals to the wave window (section 3.6).
- Step7.** Recompile changed verilog files (section 3.7).
- Step8.** Restart/Run the simulation (section 3.8).

3.1 Step1: Create a Directory for your Project

- a. Because ModelSim creates rather large output files you should not save your ModelSim projects. It is a simple matter to recreate the project anyway.
- b. Create a directory for your simulation.
 - i. For CS61C Students: create a sub-directory somewhere in your home directory, perhaps something like `~/simulation/`.
 - ii. For EECS150 students: create a sub-directory in `C:\Users\cs150-xxx\`, perhaps something like `C:\Users\cs150-xxx\Simulation`.
- c. When you are done simulating delete this entire directory, this will remove the ModelSim project and all of its temporary files. Obviously your source code should be elsewhere, so that you do not delete it.

3.2 Step2: Start ModelSim and Create a Project

- a. Start ModelSim.
 - i. On windows there is often a shortcut on the desktop or start menu.
 - ii. On UNIX or Linnux, simply run `vsim &`.
 - iii. For CS61C students you will need to take the extra step of logging into `ilinux1.eecs.berkeley.edu`, `ilinux2` or `ilinux3` where ModelSim is installed. Be sure to login using `ssh -X` to enable X-Windows forwarding or the GUI will not be able to launch.
- b. At the main ModelSim window go to `File` → `New` → `Project`.
 - i. Enter a project name, this is for your reference only.
 - ii. Set the `Project Location` to the directory you created in section 3.1 above.
 - iii. You can leave the `Default Library Name` as `work`.
 - iv. Click `OK`.

3.3 Step3: Add Your Verilog to the Project

- a. Click `Add Existing File` to add your Verilog files to the project.
 - i. Click `Browse` to locate the Verilog files you wish to add.
 - ii. Note: you can add multiple files at a time by using `Shift-Click` or `Control-Click` to select them all at once.
 - iii. Leave `Add File as Type` on default.
 - iv. Leave `Folder as Top Level`.
 - v. You will almost certainly want to select `reference from current location`.
 - *. Otherwise you will end up with multiple copies of the same Verilog file floating around, a sure way to lose something.
 - †. Note that you will want to select `copy to project directory` for `Const.v` and any other include files, otherwise ModelSim will not be able to find them. These files should rarely, if ever, change so making copies presents little danger.

- vi. Click **OK**.
- b. Repeat this until all of the necessary Verilog files have been added to the project.
 - i. All of the Verilog files for the modules you are using must be added this way, not just the top level testbench.
 - ii. If you are simulating a project involving Xilinx library components you will need to add `C:\Xilinx\Verilog\src\glbl.v` or `C:\Xilinx\10.1\ISE\verilog\src\glbl.v` to your project.
- c. Click **Close**.

3.4 Step4: Compile your Verilog Files

- a. The project pane on the left of the main ModelSim window should list all of the files in your project with an icon next to each one.
 - i. A **?** means that the file has not been compiled since the last edit.
 - ii. An **X** means that the file could not be compiled, it has an error.
 - *. Double clicking the **X** will bring up a list of errors with line numbers.
 - †. **Control-G** in Windows notepad lets you jump to a specific line number.
 - iii. A **✓** means that the file has been compiled successfully.
- b. Define the `MODELSIM` compiler flag for use with `Const.v`.
 - i. Who needs to do this?
 - *. Students in CS61C and EECS150 may skip this step as ModelSim has been installed to do this by default.
 - †. Anyone else not using Verilog files from RDLC2, CS61C or EECS150 may also skip this step.
 - ‡. If you followed the installation instructions above, you have already defined this flag and may skip this step.
 - ii. Use **Shift-Click** to select all of your Verilog files, and then **Right-Click** and select **Properties**.
 - iii. Go to the **Verilog** tab in the **Properties** window.
 - iv. Click the **Macro** button.
 - v. Enter `MODELSIM` into the **Macro Name** box, leaving the **Value** box empty.
 - vi. Click **OK** until you get to the ModelSim main window.
- c. **Right-Click** in the **Project** pane and select **Compile** → **Compile Out-of-Date**, this will attempt to compile all of the files with **?** or **X** next to them.
 - i. If you change any Verilog source files you must recompile them, using **Compile Out-of-Date**, before restarting the simulation.
 - ii. You can also use **Compile All**, however on projects with a large number of files this may take a while.

3.5 Step5: Start the Simulation

- a. Go to **Simulate** → **Start Simulation** to bring up the simulation dialog box.
 - i. Alternatively you may use the `vsim` TCL command at the prompt. See the ModelSim manual for more information. This will be much faster and easier for more advanced users.
- b. Go to the **Libraries** tab.
 - i. Skip these steps if you are not using Xilinx library components.

- ii. Click **Add** to add a new search library.
 - iii. Navigate to `C:\ModelTech_version\xilinx\verilog\unisims_ver` and then click **OK**, this will add the `unisims_ver` library to your project.
 - iv. Click **Add** to add a new search library.
 - v. Navigate to `C:\ModelTech_version\xilinx\verilog\XilinxCoreLib_ver` and then click **OK**, this will add the `XilinxCoreLib_ver` library to your project.
- c. Go to the **Design** tab.
- i. Click the plus next to the **work** library.
 - ii. Find your testbench and select it.
 - iii. If you are simulating a project involving Xilinx library components you will need to add a space and then `g1b1` to the text-box listing the modules, design units or simulation elements in the lower left.
 - iv. Ensure that the **Enable Optimizations** box is unchecked, if you want to see all internal signals. EECS150 and CS61C students should make sure this box is unchecked at all times.
 - v. Click **OK**.
- d. Your design should now be loaded and ready to simulate.
- i. If your design does not load, reread the above steps carefully.
 - ii. You should also check for errors and warnings from ModelSim, both are indicative of potential problems with your code.
 - iii. Some warnings are important, whereas some (particularly about the `'timescale` directive) may be ignored. Which are which are beyond this tutorial.

3.6 Step6: Add Signals to the Wave Window

- a. With the simulation running, the **Sim** panel should be visible on the left hand side of the ModelSim main window.
- i. The **Sim** panel shows the hierarchy of all the modules in your project.
 - ii. Clicking the plus next to a module will show the modules instantiated within it.
- b. You should add as many signals (wires) as you might need to the ModelSim wave window before simulation. If you add signals after you have started simulation you will need to restart the simulation as in section 3.8 below.
- c. To add all signals from a module
- i. **Right-Click** on a module in the **Sim** panel and select **Add** → **Add to Wave**. Note that you almost certainly do **not** want **Add All to Wave**, as that will add all the signals in your design making it hard to see anything of value.
 - ii. This will add all of the signals from that module to the **Wave** window.
 - iii. Please, look at signals inside your modules rather than just in the testbench.
 - iv. Remember that when debugging you generally need to strike a balance between too many signals which merely confuses things and too few which making it impossible to see what you need.
 - v. It is common to need to restart a simulation multiple times, each time adding more or different signals to the **Wave** window as needed.
- d. To add individual signals
- i. Go to the **Signals** window, or the **Signals** panel (which tends to be in the middle of the main window). This is the window or panel which lists all the signals (wires) in the module instance currently selected in the **Sim** panel.

- *. If it is not visible, use the **View** → **Signals** window to display it
- ii. Drag the signals you wish to see to the **Wave** window
- e. You can view busses in any number system, rather than just binary and hexadecimal is recommended.
 - i. Select a signal, or group of signals
 - ii. **Right-Click** on them and select **Radix** → **Hexadecimal** to display those signals in hex.
 - iii. This can allow you to see significantly more data on one screen and can make it easier to read.

3.7 Step7: Recompile Changed Verilog Files

- a. When you have fixed bugs, or simply made changes you must recompile your Verilog files.
- b. Near the bottom left of the ModelSim main window are tabs for the **Project** and **Sim** panes.
- c. Navigate to the **Project** pane.
- d. **Right-Click** anywhere in the pane and select **Compile** → **Compile Out-of-Date** to recompile the modified files (Which should have **?** next to them).

3.8 Step8: Restart/Run the Simulation

- a. You must do this any time to make changes such as:
 - i. Changing/recompiling Verilog.
 - ii. Adding new signals to the wave window.
- b. At the ModelSim command prompt type `restart -f; run 100us` this will run your simulation for $100\mu\text{s}$.
 - i. `restart -f` tells ModelSim that it needs to start the simulation over from the beginning, that is to say at time 0.
 - ii. `run 100us` tells ModelSim to run the simulation for 100 micro-seconds (“ μ ” is hard to type, so ModelSim uses “u” instead). `100ns` would be 100 nano-seconds. `100ms` would be 100 milli-seconds.
 - iii. The semicolon simply separates the two commands.
 - iv. You may use the up arrow key to select a previous command, like at most command prompts.
- c. Note that if you cannot see all of the interesting events, you may run the simulation for additional time. $100\mu\text{s}$ may be a very long time or a very short one depending on your clock rate and circuit.
 - i. Example: (total running time: $110\mu\text{s}$)


```
restart -f; run 100us
run 10us
```

4 More Information

For more information on ModelSim, please check the manuals and documentation that accompany it. For more information on RDLC2 and it’s interaction with ModelSim please see the CounterExample in the RDLC2 distribution.