

Introduction to Modelsim on Networked PCs

ModelSim is a single kernel, dual language simulator. You are able to run either Verilog or VHDL separately or mixed in the same design. You can have Verilog modules instantiated in VHDL architectures or VHDL entities instantiated in Verilog Modules. You can even mix languages at any level of abstraction and with any number of hierarchical levels i.e. Verilog module instantiated by a VHDL architecture called from a Verilog module. One simulator, one interface, two languages. In this course we are only focusing on VHDL and we will only learn how to work with VHDL modules in Modelsim.

Before starting to use Modelsim you will have to create a directory structure as depicted in Figure 1 on **H:**\ (or your shared network drive) .

Note: The directory “Work”, is created the first time Modelsim is started, all other directories must be created manually with the Explorer.

This directory structure (see Figure 1) is a common way of managing the files needed for a VHDL project. Table 1 explains the contents of each directory.

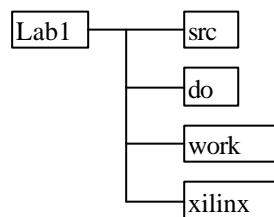


Figure 1: Directory structure in Modelsim.

Directory	Contents
src	In this directory all the VHDL source code is stored. (.vhd)
do	Simulation files needed in Modelsim simulator. (.do)
work	Working directory for Modelsim.
xilinx	Synthesis directory, used in Xilinx ISE.

Table 1: Explanation of each directory in Modelsim.

ModelSim can be used in batch mode, command line, or with the User Interface (UI). Batch mode is the typical method when running regression tests (not issued in this course). Command line mode is very similar to batch mode in the fact that the UI is not displayed, the only interface is a command line console. The user interface mode can accept both command line and UI input, and will be used in this course. Following commands in this document can be used in any mode. Note that it is possible to create a script that is read by Modelsim instead of using the UI input. If you save the commands in a file (*my_file.do* is a common macro file naming convention), you can use it in UI mode, command, or batch modes and thereby automatically execute all the steps needed for a simulation. Below is a short example of how such a script may be created, however this is not mandatory.

```

vlib work          # Creates a design library
vmap work work    # Defines or displays library mappings
vcom my_file.vhd  # Compiles the VHDL design
vsim work.my_file # Loads the design
view *            # View all signals/nets
add wave /*       # Add all signals in the design
run 1000000000    # Set the simulation run time

```

The following procedure explains the above steps and uses the UI in a particular way. Start ModelSim by double clicking the icon or from the Windows Start menu.

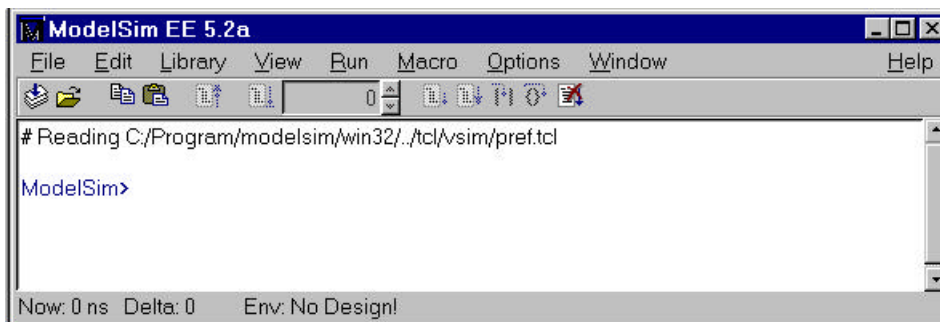


Figure 2: Modelsim main window.

1. Move into the design directory. Select **File > Change Directory**, use the file browser to locate the desired directory. This directory becomes the working directory. Any library that is now created will be placed into this directory by default.

Note: In this example, you will only browse to “Lab1” directory (see Figure 1).

2. Create a working library. Select **Library > Create A New Library**. Ensure that you select *a new library and logical mapping to it* and enter work in the library box as shown in
3. Figure 3: Create a New Library Dialog

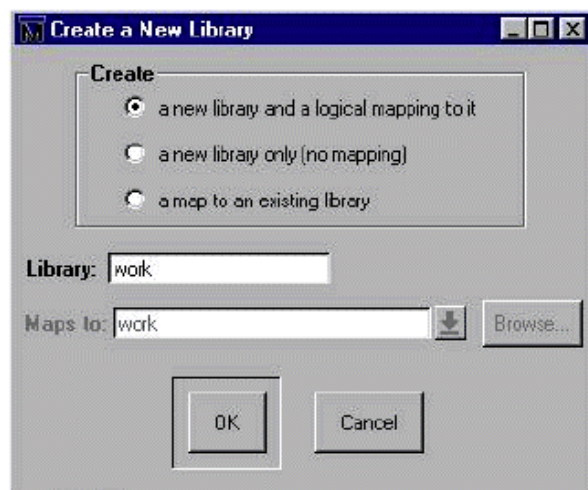


Figure 3: Create a New Library Dialog

4. Select **OK** to except the entry. This executes both the vlib and vmap commands shown in the command line script listing above.
5. Next step is to compile the VHDL (my_file.vhd) design file. This is achieved by entering a Modelsim specific (**vcom**) command in the prompt.

vcom src/my_file.vhd –explicit -93 (compiles according to the VHD93 standard)

This command will automatically selects the correct language compiler based upon the source file selected (**vcom** for VHDL source files and **vlog** for Verilog source files).

You may also compile multiple files by using

vcom src/my_file.vhd src/my_file2.vhd –explicit -93

The files will be compiled in the order stated by the above command.

6. Loading the simulator. Select **File > Load New Design** or select the **Load Design** button. This displays the **Load Design** dialog box (see Figure 4), which displays the design units available in each of the libraries. Ensure that the design tab is highlighted. Ensure that the work library is visible in the library section. Each of the units in the work library is displayed along with a description of the type of unit. The description includes entity, architecture, config or package for VHDL units and module for Verilog units. Each unit type is highlighted with a different color. At the beginning of the line with a VHDL entity there in a '+' sign. Toggling this '+' sign with the mouse shows the architectures that have been compiled for the entity. Once the correct design is selected press the **Load** button. This loads each of the design units needed for the simulation of either the VHDL configuration or the Verilog testbench module.

Note: The cursor line in the ModelSim main window changes to VSIM >, which indicates that simulation mode is active.

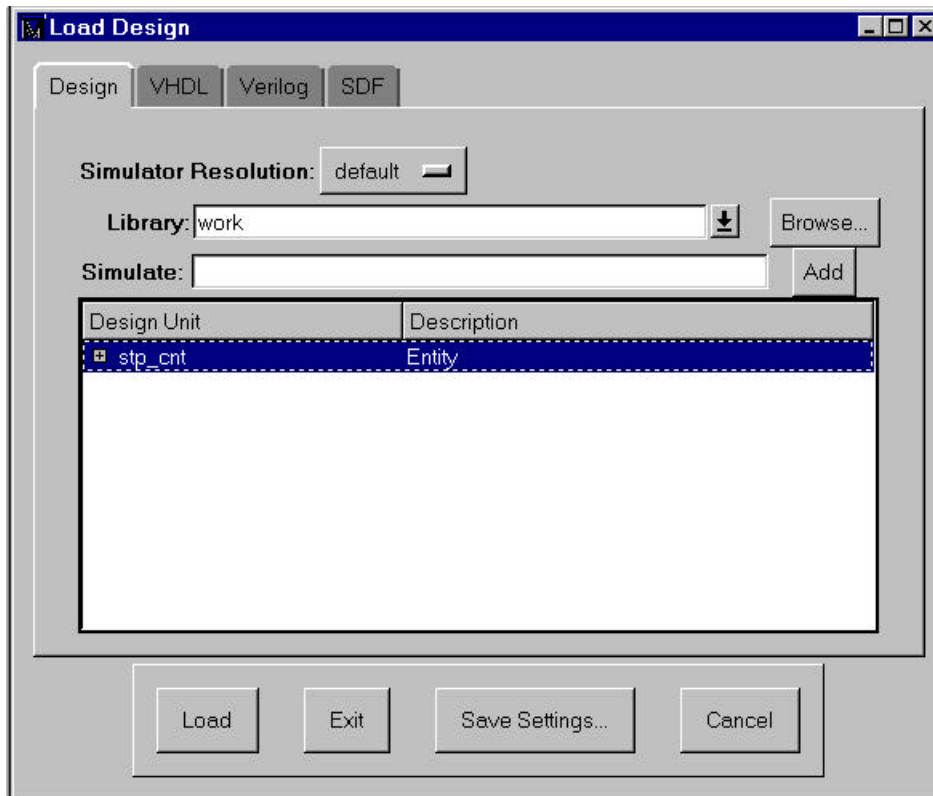


Figure 4: Load Design dialog box.

Creating a simulation file (DO-file)

In order to simulate the VHDL design, a simulation file is needed (my_file.do). This file is created in an ordinary text editor (notepad works fine). The DO-file contains the information of all input signals and their behaviour. For example, in all projects there exists a system clock (clk). The designer will have to generate this clock in the DO-file. Table 2 shows the common commands needed to create a simple DO-file. In Appendix 1 all available commands are found.

restart -force	Restarts the simulation time. Normally included in the beginning of the DO-file.
destroy .wave	Closes the waveform window. This command is also included in the beginning of the DO-file. Normally the same simulation file is used several times and this command will destroy all the previous waveforms, instead of manually closing the waveform windows.
add wave /* or add wave -r /*	Adds the waveform of interest. * will add all the signals used in the design at the top level. You may also specify a specific signal of interest instead of viewing all the signals. -r option is used when several blocks are simulated at the same time and all the signals in each block is needed to be analyzed.
view wave	Opens the graphical user interface window in Modelsim.
force RESET_N 0	The force command sets a specific value of a signal or bus. In this case the signal RESET_N is set to a logically low value, i.e. '0'.
force clk 0 0, 1 50 -repeat 100	The option -repeat is an extension to the force command. This will set the signal to behave in a periodic manner. In this case a system clock (clk) or synchronization signal is generated. In this example the signal clk will be set to '0' at 0 ns and to '1' after 50 ns, and this sequence will be repeated every 100 ns.
#comment	You are able to comment your simulation file by using #.
run 100	The run command specifies the simulation time. In this case the simulation will run in 100 ns.

Table 2: Common simulation commands, used in Modelsim.

Below is an example of a simple DO-file, where only the reset and system clock is generated.

```

destroy .wave                # Closes all existing windows
destroy .source              # Destroys all previous existing sources in the simulation
destroy .variables           # Destroys all previous existing variables in the simulation
restart -force                # Forces the simulator to reset the time
add wave /*                  # Wave all signals/nets in the design
view wave                    # Display wave window
force clk 0 0, 1 100 -repeat 200 # Generate a periodic system clock
force reset_n 0              # Sets reset_n to '0' in 50 ns
run 50
force reset_n 1              # Sets reset_n to '1' after 50 ns
run 1800                      # Sets the simulation time to 1800 ns

```

Once the DO-file is created and the design is compiled it is time for simulating the design with the input stimuli from the DO-file. The command for executing the DO-file is:

do do/my_file.do

Note: The first “do” statement executes the DO-file (my_file.do) , located in the Lab1/do/ directory.

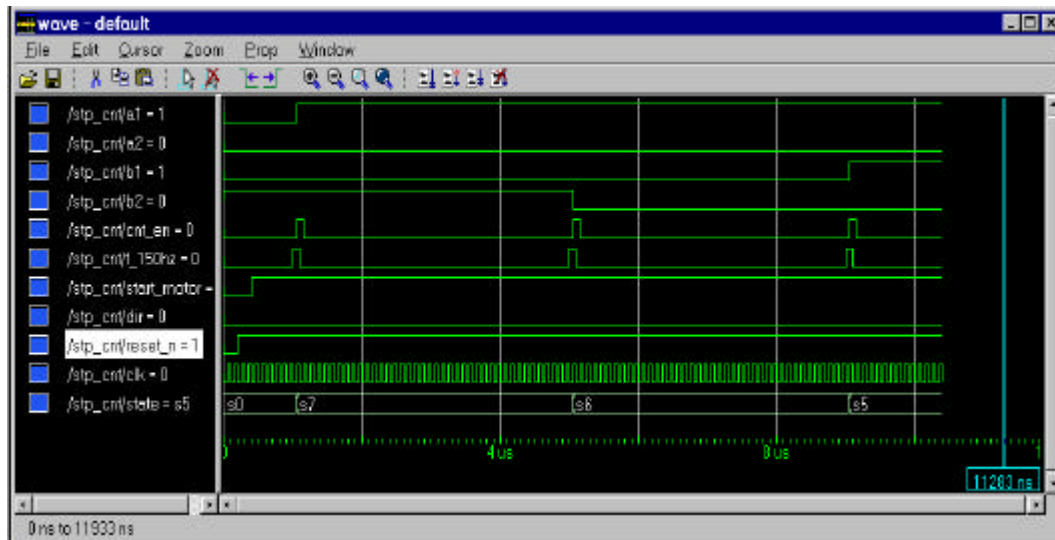


Figure 5: Simulation Wave window.

At the end of the run time, the **Wave** window (see Figure 5) displays the results of the simulation. It is up to the designer to analyse whether the design works correctly or not. Normally simulations are a time consuming step in the design of digital circuits, and it is suggested to simulate the design in smaller parts before implementing the total module (process by process simulations).

Appendix 1

Key ModelSim Commands

Commands may be used in the following locations: (Sh)ell, (M)odelSim> prompt, or (V)SIM> prompt.

<i>vcom</i>	Sh, M, V	VHDL Compiler (see below)
<i>vdel</i>	Sh, M, V	Deletes a design unit from a specific library
<i>vdir</i>	Sh, M, V	Lists the contents of a library
<i>vlib</i>	Sh, M, V	Creates a design library
<i>vlog</i>	Sh, M, V	Verilog Compiler
<i>vmap</i>	Sh, M, V	Defines or displays library mappings
<i>vsim</i>	Sh, M, V	VHDL and/or Verilog Simulator
<i>add list wave</i>	V	Add signals to the List or Wave windows
<i>add log</i>	V	Log signals to <i>vsim.wlf</i> file for analysis later
<i>alias</i>	M, V	Create a user defined alias (e.g., <i>alias h "history"</i>)
<i>bp, bd</i>	V	Set/Clear a breakpoint
<i>cd</i>	Sh, M, V	Change directory
<i>change</i>	V	Modify a VHDL variable or Verilog register
<i>checkpoint</i>	V	Save the state of you simulation (see <i>restore</i>)
<i>compare add</i>	M, V	Compare signals
<i>configure</i>	M, V	Configure List or Wave window attributes
<i>delete</i>	V	Remove HDL item from List or Wave window
<i>do</i>	M, V	Execute a file of commands (e.g., <i>do macro.do</i>)
<i>drivers</i>	V	Display current and future value of signal or net drivers
<i>dumplog64</i>	Sh	Dump the contents of the <i>vsim.wlf</i> file in a readable form
<i>echo</i>	M, V	Display message (e.g., <i>echo "Time is \$now ns."</i>)

<i>edit</i>	M, V	Invoke editor specified by the EDITOR env variable
<i>environment</i>	M, V	Display or change current region/signal environment
<i>examine</i>	M, V	Examine one or more HDL items (e.g., <i>exa /top/clk</i>)
<i>find</i>	V	Display pathnames of matching HDL items
<i>force</i>	V	Force signals or nets (e.g., <i>force clk 1 10, 0 20 -r 100</i>)
<i>history</i>	M, V	List previous commands
<i>noforce</i>	V	Release signals or nets from force commands
<i>notepad</i>	M, V	Simple text editor
<i>printenv</i>	M, V	Display names and values of environment variables
<i>profile on</i>	M, V	Turn on Performance Analyzer
<i>property</i>	V	Change List or Wave signal attributes (color, radix, etc.)
<i>pwd</i>	M, V	Display current path in Main transcript window
<i>radix</i>	M, V	Change the default radix in all windows
<i>report</i>	M, V	Returns all control or state variable values
<i>restart</i>	V	Restart the simulator
<i>restore</i>	M, V	Restore the simulation state from a previous <i>checkpoint</i>
<i>resume</i>	M, V	Resume macro execution after a pause command
<i>right left</i>	V	Search in wave window for next transition or -expr
<i>run</i>	V	Advance simulation time (e.g., <i>run 1000</i>)
<i>search next</i>	V	Search specified window for next item matching pattern
<i>seetime</i> <i>500)</i>	V	Scroll List or Wave window to time(e.g., <i>seetime wave</i> <i>500)</i>
<i>up down</i>	M, V	Toggle thru last commands

Wave Window

<i>add wave <item> <item></i>	Wave specific signals/nets
<i>add wave *</i>	Wave signals/nets in scope
<i>add wave -r /*</i>	Wave all signals/nets in design
<i>add wave -label <name> <item></i>	Wave and rename a signal/net
<i>add wave bus(31:15)</i>	Wave a slice of a bus
<i>view wave</i>	Display wave window
<i>view wave -new</i>	Display additional wave window
<i>write wave</i>	Print wave window to file
<i><left mouse button></i>	Select signal / Place cursor
<i><middle mouse button></i>	Zoom options
<i><right mouse button></i>	Context Menu
<i><ctrl-f></i>	Find next item
<i><tab></i>	(go right) Search forward for next edge
<i><shift-tab></i>	(go left) Search backward for next edge
<i>i or + o or -</i>	Zoom in Zoom out
<i>f l</i>	Zoom full Zoom Last

vcom

<i>[-93] [-87]</i>	Choose VHDL-1993 or 1987
<i>[-check_synthesis]</i>	Turn on synthesis checker
<i>[-debugVA]</i>	Print VITAL opt status
<i>[-05]</i>	Maximum optimization
<i>[-explicit]</i>	Resolve ambiguous overloads
<i>[-help]</i>	Display <i>vcom</i> syntax help
<i>[-f <filename>]</i>	Pass in arguments from file
<i>[-norangecheck]</i>	Disable run time range checks
<i>[-nodebug]</i>	Strip internal names
<i>[-novitalcheck]</i>	Disable VITAL95 checking
<i>[-nowarn <#>]</i>	Disable individual warning msg
<i>[-O0]</i>	Disable optimization
<i>[-quiet]</i>	Disable loading messages
<i>[-refresh]</i>	Regenerate library image
<i>[-version]</i>	Returns vcom version
<i>[-work <libname>]</i>	Specify <i>work</i> library