

Model*Sim*®

Advanced Verification and Debugging

Xilinx Tutorial

Version 6.0a

Published: September 24, 2004



This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

RESTRICTED RIGHTS LEGEND 03/97

U.S. Government Restricted Rights. The SOFTWARE and documentation have been developed entirely at private expense and are commercial computer software provided with restricted rights. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement provided with the software pursuant to DFARS 227.7202-3(a) or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

Table of Contents

Introduction	T-5
Lesson 1 - ModelSim conceptual overview	T-9
Lesson 2 - Basic simulation	T-17
Lesson 3 - ModelSim projects	T-29
Lesson 4 - Working with multiple libraries	T-39
Lesson 5 - Viewing simulations in the Wave window	T-49
Lesson 6 - Viewing and initializing memories	T-59
Lesson 7 - Automating ModelSim	T-73
T-83	
Index	T-89

Introduction

Topics

The following topics are covered in this chapter:

Assumptions	T-6
Before you begin	T-8
Example designs.	T-8

Assumptions

We assume that you are familiar with the use of your operating system. If you are not familiar with Microsoft Windows, we recommend that you work through the tutorials provided with MS Windows before using *ModelSim*.

We also assume that you have a working knowledge of VHDL and/or Verilog. Although ModelSim is an excellent tool to use while learning HDL concepts and practices, this document is not written to support that goal.

Where to find our documentation

ModelSim documentation is available from our website at www.model.com/support or in the following formats and locations:

Document	Format	How to get it
<i>ModelSim Installation & Licensing Guide</i>	paper	shipped with ModelSim
	PDF	select Help > Documentation ; also available from the Support page of our web site: www.model.com
<i>ModelSim Quick Guide</i> (command and feature quick-reference)	paper	shipped with ModelSim
	PDF	select Help > Documentation , also available from the Support page of our web site: www.model.com
<i>ModelSim Tutorial</i>	PDF, HTML	select Help > Documentation ; also available from the Support page of our web site: www.model.com
<i>ModelSim User's Manual</i>	PDF, HTML	select Help > Documentation
<i>ModelSim Command Reference</i>	PDF, HTML	select Help > Documentation
<i>ModelSim GUI Reference</i>	PDF, HTML	select Help > Documentation
Command Help	ASCII	type <code>help [command name]</code> at the prompt in the Transcript pane
Error message help	ASCII	type <code>verror <msgNum></code> at the Transcript or shell prompt
Tcl Man Pages (Tcl manual)	HTML	select Help > Tcl Man Pages , or find <i>contents.htm</i> in <code>\modeltech\docs\tcl_help_html</code>
Technotes	HTML	select Technotes dropdown on www.model.com/support

Before you begin

Preparation for some of the lessons leaves certain details up to you. You will decide the best way to create directories, copy files, and execute programs within your operating system. (When you are operating the simulator within ModelSim's GUI, the interface is consistent for all platforms.)

Example designs

ModelSim comes with Verilog and VHDL versions of the designs used in these lessons. This allows you to do the tutorial regardless of which license type you have. Though we have tried to minimize the differences between the Verilog and VHDL versions, we could not do so in all cases. In cases where the designs differ (e.g., line numbers or syntax), you will find language-specific instructions. Follow the instructions that are appropriate for the language that you are using.

Lesson 1 - ModelSim conceptual overview

Topics

The following topics are covered in this chapter:

Introduction	T-10
Basic simulation flow	T-11
Creating the working library	T-11
Compiling your design	T-11
Running the simulation	T-11
Debugging your results	T-12
Project flow	T-13
Multiple library flow	T-14
Debugging tools	T-15

Introduction

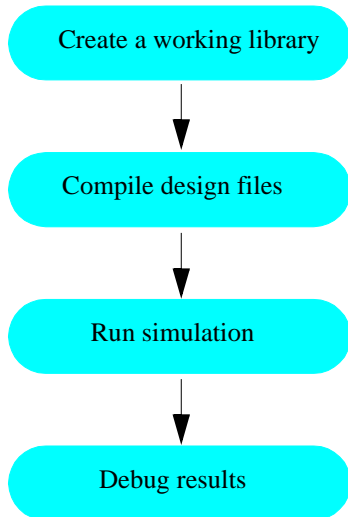
ModelSim is a simulation and debugging tool for VHDL, Verilog, and mixed-language designs.

This lesson provides a brief conceptual overview of the ModelSim simulation environment. It is divided into four topics, which you will learn more about in subsequent lessons:

Topic	Additional information and practice
Basic simulation flow	<i>Lesson 2 - Basic simulation</i>
Project flow	<i>Lesson 3 - ModelSim projects</i>
Multiple library flow	<i>Lesson 4 - Working with multiple libraries</i>
Debugging tools	Remaining lessons

Basic simulation flow

The following diagram shows the basic steps for simulating a design in ModelSim.



Creating the working library

In ModelSim, all designs, be they VHDL, Verilog, or some combination thereof, are compiled into a library. You typically start a new simulation in ModelSim by creating a working library called "work". "Work" is the library name used by the compiler as the default destination for compiled design units.

Compiling your design

After creating the working library, you compile your design units into it. The ModelSim library format is compatible across all supported platforms. You can simulate your design on any platform without having to recompile your design.

Running the simulation

With the design compiled, you invoke the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL). Assuming the design loads successfully, the simulation time is set to zero, and you enter a run command to begin simulation.

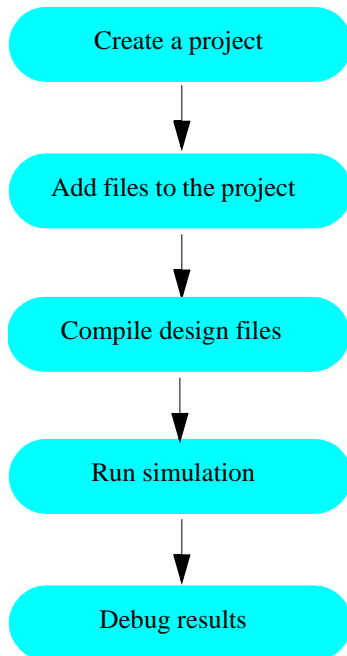
Debugging your results

If you don't get the results you expect, you can use ModelSim's robust debugging environment to track down the cause of the problem.

Project flow

A project is a collection mechanism for an HDL design under specification or test. Even though you don't have to use projects in ModelSim, they may ease interaction with the tool and are useful for organizing files and specifying simulation settings.

The following diagram shows the basic steps for simulating a design within a ModelSim project.



As you can see, the flow is similar to the basic simulation flow. However, there are two important differences:

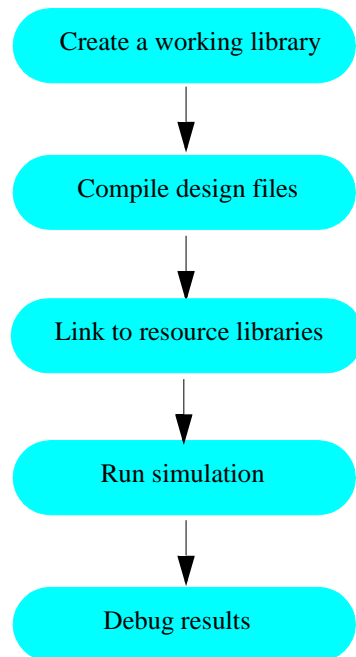
- You do not have to create a working library in the project flow; it is done for you automatically.
- Projects are persistent. In other words, they will open every time you invoke ModelSim unless you specifically close them.

Multiple library flow

ModelSim uses libraries in two ways: 1) as a local working library that contains the compiled version of your design; 2) as a resource library. The contents of your working library will change as you update your design and recompile. A resource library is typically static and serves as a parts source for your design. You can create your own resource libraries, or they may be supplied by another design team or a third party (e.g., a silicon vendor).

You specify which resource libraries will be used when the design is compiled, and there are rules to specify in which order they are searched. A common example of using both a working library and a resource library is one where your gate-level design and testbench are compiled into the working library, and the design references gate-level models in a separate resource library.

The diagram below shows the basic steps for simulating with multiple libraries.



You can also link to resource libraries from within a project. If you are using a project, you would replace the first step above with these two steps: create the project and add the testbench to the project.

Debugging tools

ModelSim offers numerous tools for debugging and analyzing your design. Several of these tools are covered in subsequent lessons, including:

- Setting breakpoints and stepping through the source code
- Viewing waveforms and measuring time
- Viewing and initializing memories

Lesson 2 - Basic simulation

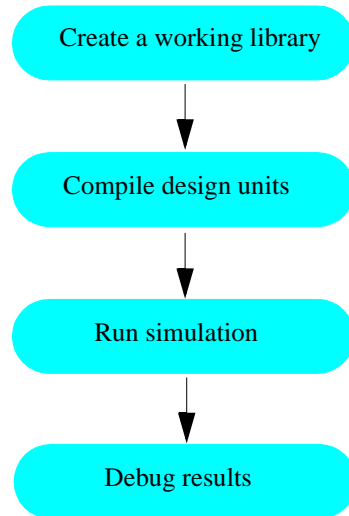
Topics

The following topics are covered in this lesson:

Introduction	T-18
Design files for this lesson	T-18
Related reading	T-18
Creating the working design library	T-19
Compiling the design.	T-21
Loading the design into the simulator	T-22
Running the simulation	T-23
Setting breakpoints and stepping in the Source window.	T-25
Lesson wrap-up	T-27

Introduction

In this lesson you will go step-by-step through the basic simulation flow:



Design files for this lesson

The sample design for this lesson is a simple 8-bit, binary up-counter with an associated testbench. The pathnames are as follows:

Verilog – `<install_dir>/modeltech/examples/counter.v` and `tcounter.v`

VHDL – `<install_dir>/modeltech/examples/counter.vhd` and `tcounter.vhd`

This lesson uses the Verilog files `counter.v` and `tcounter.v` in the examples. If you have a VHDL license, use `counter.vhd` and `tcounter.vhd` instead. Or, if you have a mixed license, feel free to use the Verilog testbench with the VHDL counter or vice versa.

Related reading

ModelSim User's Manual – [Chapter 3 - Design libraries](#) (UM-45), [Chapter 5 - Verilog simulation](#) (UM-91), [Chapter 4 - VHDL simulation](#) (UM-59)

ModelSim Command Reference ([vlib](#) (CR-207), [vmap](#) (CR-217), [vlog](#) (CR-208), [vcom](#) (CR-171), (CR-217), [view](#) (CR-183), and [run](#) (CR-135) commands)

Creating the working design library

Before you can simulate a design, you must first create a library and compile the source code into that library.

- 1 Create a new directory and copy the tutorial files into it.
 Start by creating a new directory for this exercise (in case other users will be working with these lessons).
Verilog: Copy *counter.v* and *tcounter.v* files from */<install_dir>/examples* to the new directory.
VHDL: Copy *counter.vhd* and *tcounter.vhd* files from */<install_dir>/examples* to the new directory.
- 2 Start ModelSim if necessary.
 - a Type **vsim** at a UNIX shell prompt or use the ModelSim icon in Windows.
 Upon opening ModelSim for the first time, you will see the Welcome to ModelSim dialog (Figure 1). Click **Close**.
 - b Select **File > Change Directory** and change to the directory you created in step 1.
- 3 Create the working library.
 - a Select **File > New > Library**.
 This opens a dialog where you specify physical and logical names for the library (Figure 2). You can create a new library or map to an existing library. We'll be doing the former.
 - b Type **work** in the Library Name field if it isn't entered automatically.

Figure 1: The Welcome to ModelSim dialog

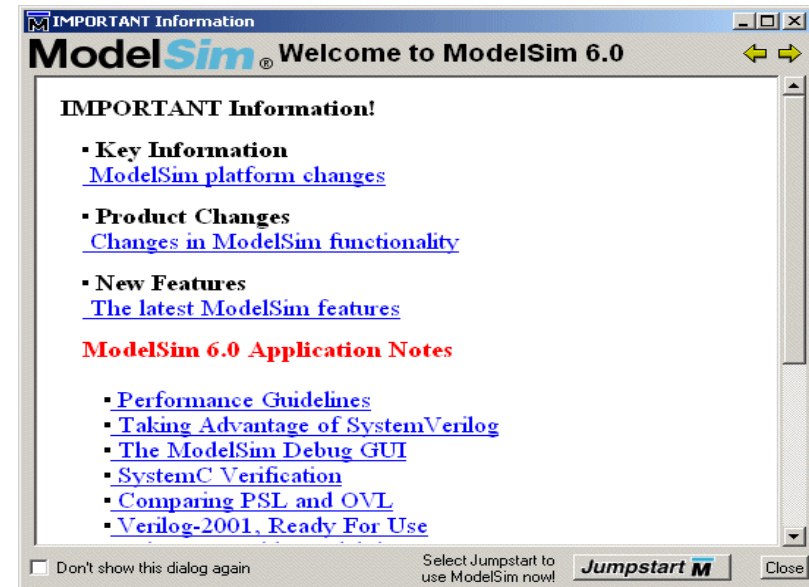
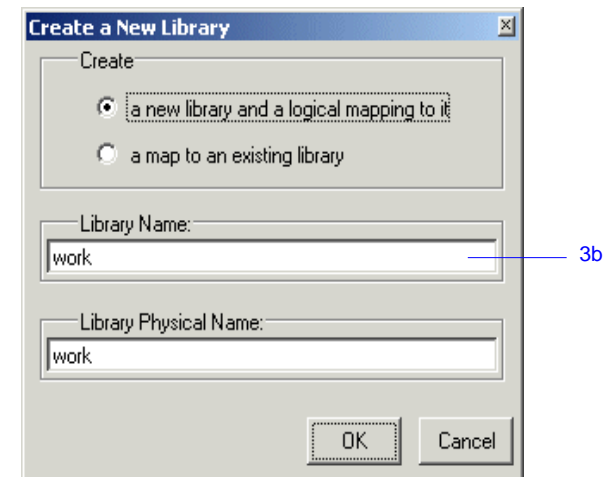


Figure 2: The Create a New Library dialog



c Click **OK**.

ModelSim creates a directory called *work* and writes a specially-formatted file named *_info* into that directory. The *_info* file must remain in the directory to distinguish it as a ModelSim library. Do not edit the folder contents from your operating system; all changes should be made from within ModelSim.

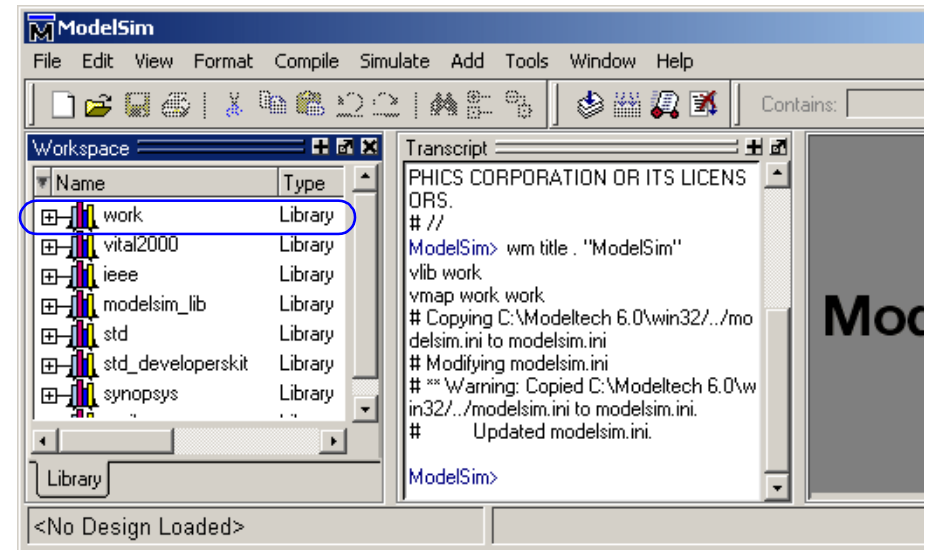
ModelSim also adds the library to the list in the Workspace (Figure 3) and records the library mapping for future reference in the ModelSim initialization file (*modelsim.ini*).

When you pressed OK in step c above, three lines were printed to the Main window Transcript pane:

```
vlib work
vmap work work
# Modifying modelsim.ini
```

The first two lines are the command-line equivalent of the menu commands you invoked. Most menu driven functions will echo their command-line equivalents in this fashion. The third line notifies you that the mapping has been recorded in the ModelSim initialization file.

Figure 3: The newly created work library



Compiling the design

With the working library created, you are ready to compile your source files.

You can compile by using the menus and dialogs of the graphic interface, as in the Verilog example below, or by entering a command at the ModelSim> prompt as in the VHDL example below.

1 Verilog: Compile *counter.v* and *tcounter.v*.

- a Select **Compile > Compile**.

This opens the Compile Source Files dialog (Figure 4).

If the Compile menu option is not available, you probably have a project open. If so, close the project by selecting **File > Close** when the Workspace pane is selected.

- b Select *counter.v*, hold the <Ctrl> key down, and then select *tcounter.v*.

- c With the two files selected, click **Compile**.

The files are compiled into the *work* library.

- d Click **Done**.

VHDL: Compile *counter.vhd* and *tcounter.vhd*.

- a Type **vcom counter.vhd tcounter.vhd** at the ModelSim> prompt and press <Enter> on your keyboard.

2 View the compiled design units.

- a On the Library tab, click the '+' icon next to the *work* library and you will see two design units (Figure 5). You can also see their types (Modules, Entities, etc.) and the path to the underlying source files if you scroll to the right.

Figure 4: The Compile HDL Source Files dialog

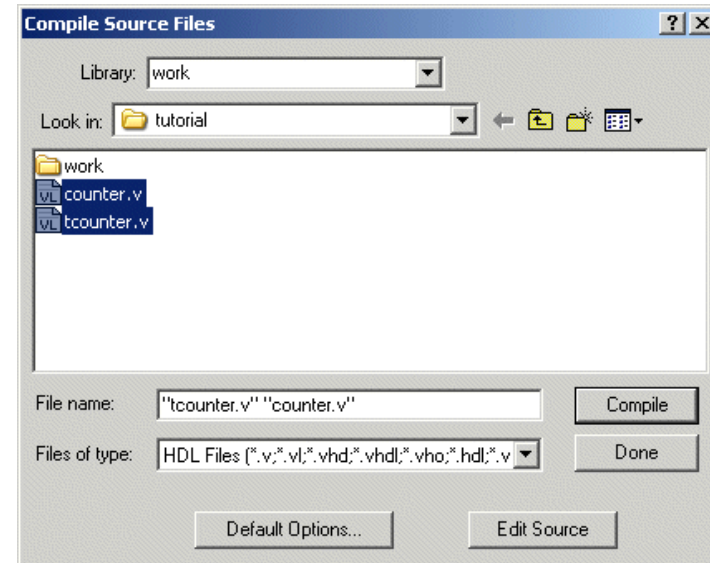
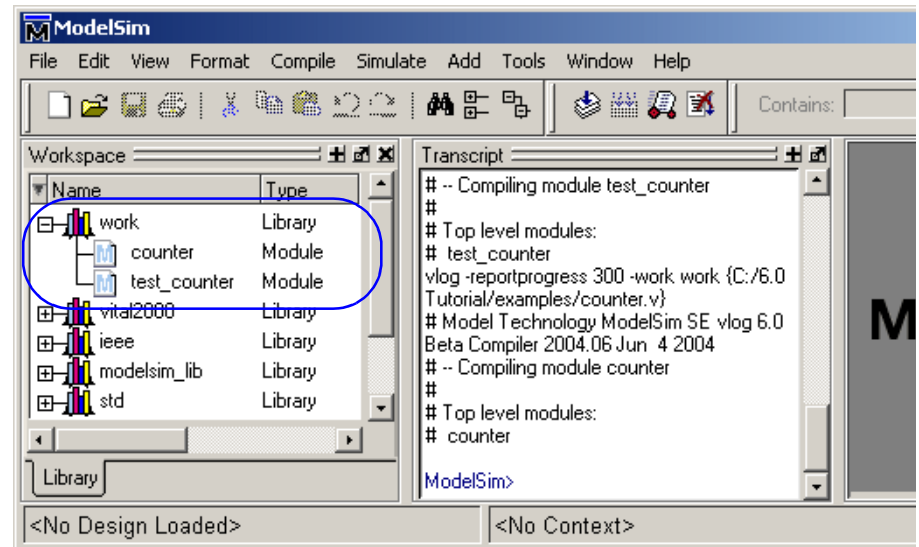


Figure 5: Verilog modules compiled into the work library



Loading the design into the simulator

- 1 Load the *test_counter* module into the simulator.
 - a Double-click *test_counter* in the Main window Workspace to load the design.

You can also load the design by selecting **Simulate > Start Simulation** in the menu bar. This opens the Start Simulation dialog. With the Design tab selected, click the '+' sign next to the work library to see the *counter* and *test_counter* modules. Select the *test_counter* module and click OK (Figure 6).

When the design is loaded, you will see a new tab named *sim* that displays the hierarchical structure of the design (Figure 7). You can navigate within the hierarchy by clicking on any line with a '+' (expand) or '-' (contract) icon. You will also see a tab named *Files* that displays all files included in the design.

Figure 6: Loading the design with the Start Simulation dialog

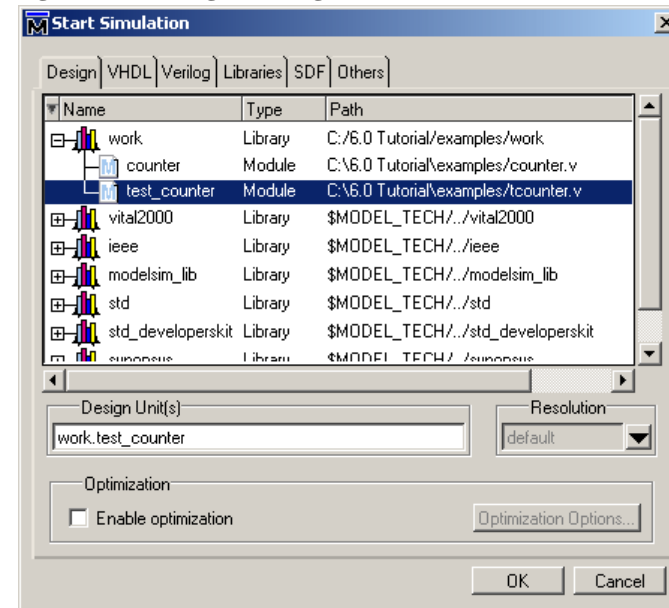
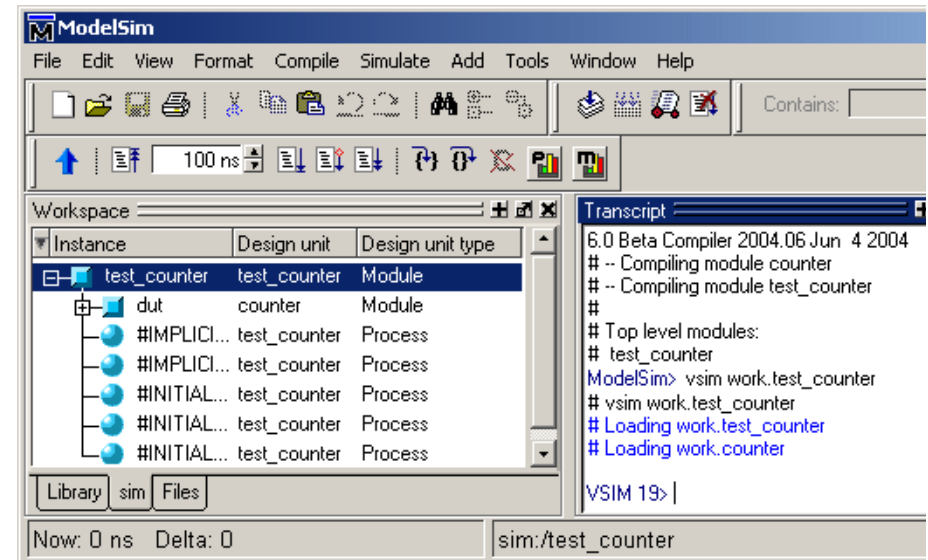


Figure 7: Workspace tab showing a Verilog design



Running the simulation

Now you will run the simulation.

- 1 Set the graphic user interface to view all debugging windows.

- a Select **View > Debug Windows > All Windows**.

This opens all ModelSim windows, giving you different views of your design data and a variety of debugging tools. Most windows will open as panes within the Main window. The Dataflow, List, and Wave windows will open as separate windows. You may need to move or resize the windows to your liking. Panes within the Main window can be undocked to stand alone.

- 2 Add signals to the Wave window.

- a In the Workspace pane, select the **sim** tab.
 - b Right-click *test_counter* to open a popup context menu.
 - c Select **Add > Add to Wave** (Figure 8).

Three signals are added to the Wave window.

- 3 Run the simulation.

- a Click the Run icon in the Main or Wave window toolbar.

The simulation runs for 100 ns (the default simulation length) and waves are drawn in the Wave window.



- b Type **run 500** at the VSIM> prompt in the Main window.

The simulation advances another 500 ns for a total of 600 ns (Figure 9).

Figure 8: Adding signals to the Wave window

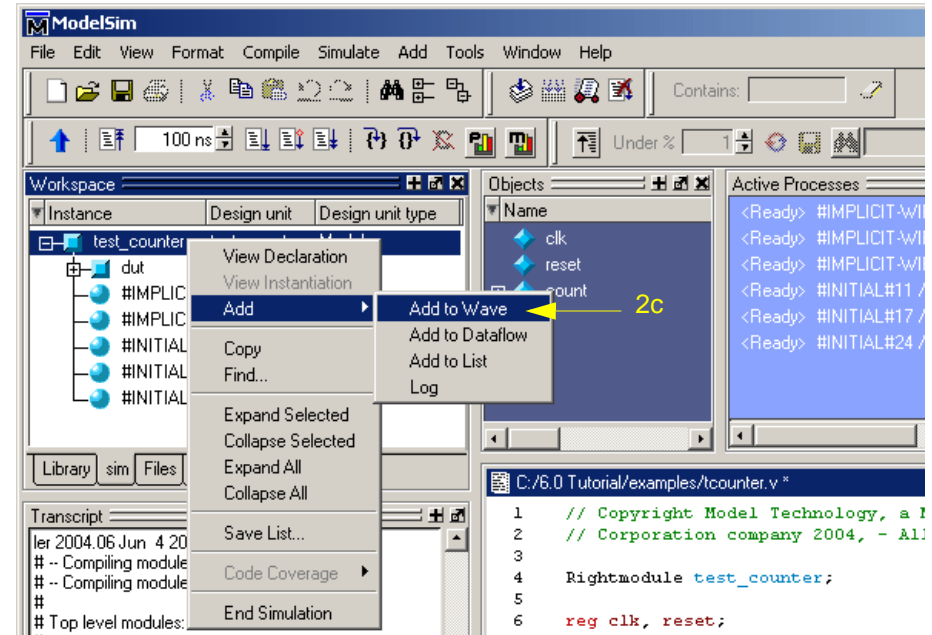
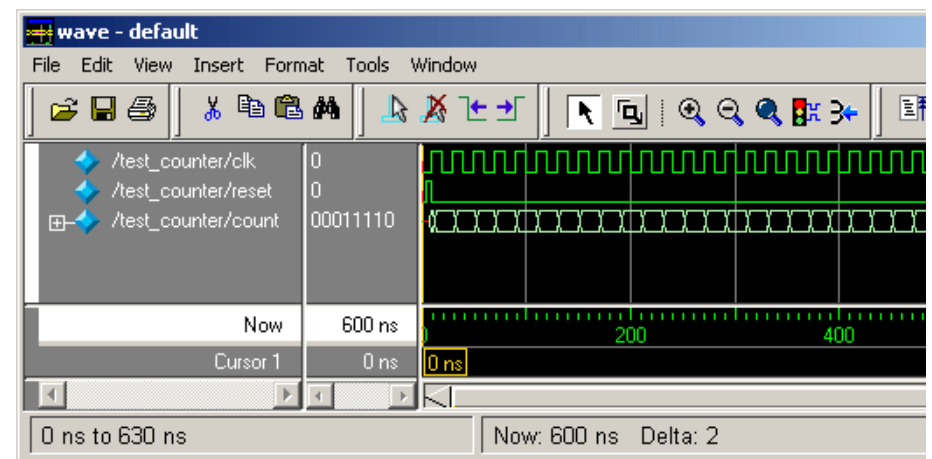


Figure 9: Waves being drawn in the Wave window



T-24 Lesson -

- c Click the Run -All icon on the Main or Wave window toolbar.

The simulation continues running until you execute a break command or it hits a statement in your code (e.g., a Verilog \$stop statement) that halts the simulation.



- d Click the Break icon.

The simulation stops running.



Setting breakpoints and stepping in the Source window

Next you will take a brief look at one interactive debugging feature of the ModelSim environment. You will set a breakpoint in the Source window, run the simulation, and then step through the design under test. Breakpoints can be set only on lines with red line numbers.

- 1 Open *counter.v* in the Source window.
 - a Select the **Files** tab in the Main window Workspace.
 - b Double-click *counter.v* to add it to the Source window.
- 2 Set a breakpoint on line 31 of *counter.v* (if you are simulating the VHDL files, use line 30 instead).
 - a Scroll to line 31 and click on the line number.
A red ball appears next to the line (Figure 10) indicating that a breakpoint has been set.
- 3 Disable, enable, and delete the breakpoint.
 - a Click the red ball to disable the breakpoint. It will become a black circle.
 - b Click the black circle to re-enable the breakpoint. It will become a red ball.
 - c Click the red ball with your right mouse button and select **Remove Breakpoint 31**.
 - d Click on line number 31 again to re-create the breakpoint.
- 4 Restart the simulation.
 - a Click the Restart icon to reload the design elements and reset the simulation time to zero.
The Restart dialog that appears gives you options on what to retain during the restart (Figure 11).
 - b Click **Restart** in the Restart dialog.



Figure 10: A breakpoint in the Source window

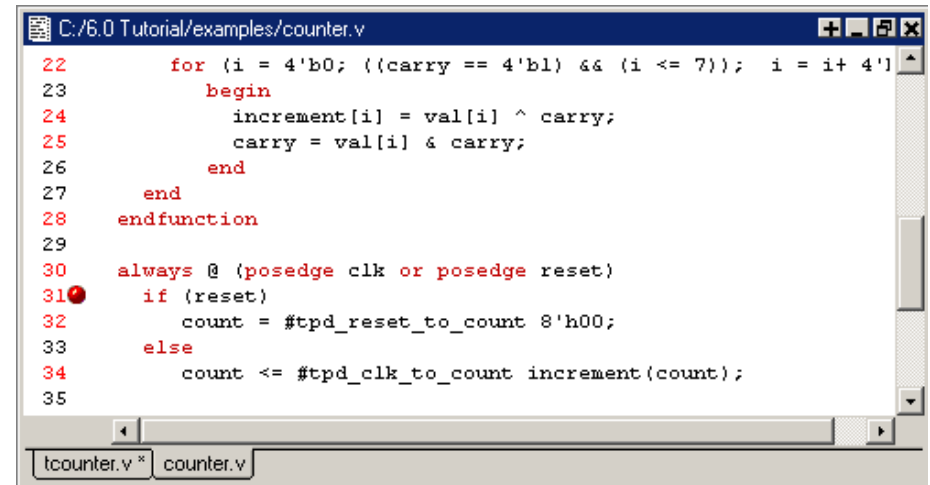
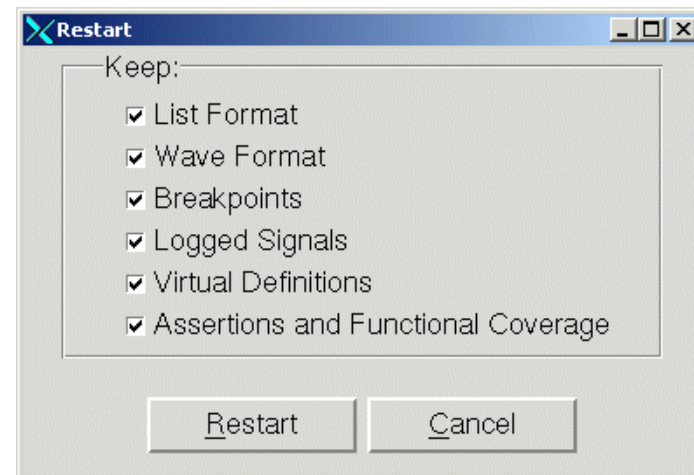


Figure 11: The Restart dialog



- c Click the Run -All icon.

The simulation runs until the breakpoint is hit. When the simulation hits the breakpoint, it stops running, highlights the line with a blue arrow in the Source view (Figure 12), and issues a Break message in the Transcript pane.



When a breakpoint is reached, typically you want to know one or more signal values. You have several options for checking values:

- look at the values shown in the Objects window (Figure 13).
- set your mouse pointer over the *count* variable in the Source window, and a "balloon" will pop up with the value (Figure 12)
- highlight the *count* variable in the Source window, right-click it, and select Examine from the pop-up menu
- use the examine command to output the value to the Main window Transcript (i.e., `examine count`)

- 5 Try out the step commands.

- a Click the Step icon on the Main window toolbar.

This single-steps the debugger.



Experiment on your own. Set and clear breakpoints and use the Step, Step Over, and Continue Run commands until you feel comfortable with their operation.

Figure 12: Resting the mouse pointer on a variable in the Source view

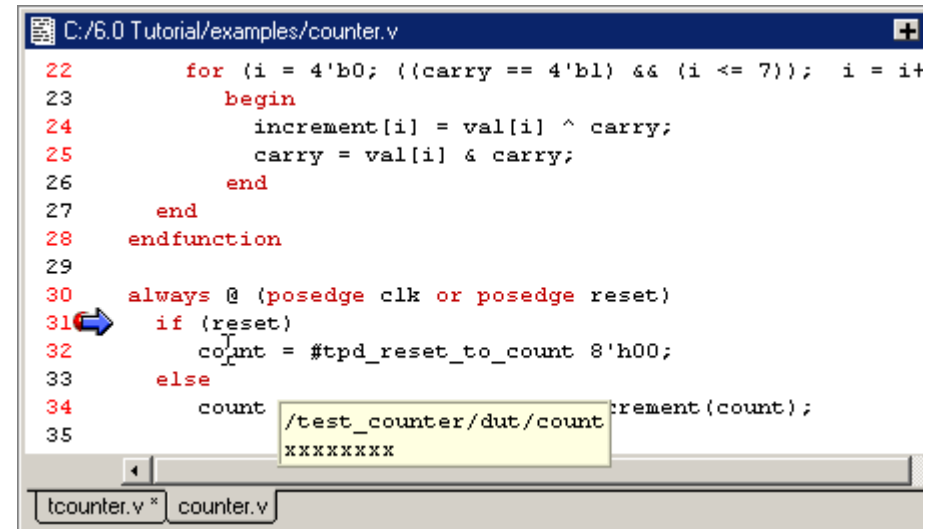


Figure 13: Values shown in the Objects window

Objects			
Name	Value	Kind	Mode
tpd_reset_to_count	3	Parameter	Internal
tpd_clk_to_count	2	Parameter	Internal
count	xxxxxxxx	Reg	Out
clk	St0	Net	In
reset	St1	Net	In

Lesson wrap-up

This concludes this lesson. Before continuing we need to end the current simulation.

- 1 Select **Simulate > End Simulation**.
- 2 Click **Yes** when prompted to confirm that you wish to quit simulating.

Lesson 3 - ModelSim projects

Topics

The following topics are covered in this lesson:

Introduction	T-30
Related reading	T-30
Creating a new project	T-31
Adding objects to the project	T-32
Changing compile order (VHDL)	T-33
Compiling and loading a design	T-34
Organizing projects with folders	T-35
Adding folders	T-35
Moving files to folders	T-36
Simulation Configurations	T-37
Lesson wrap-up	T-38

Introduction

In this lesson you will practice creating a project. At a minimum, projects have a work library and a session state that is stored in a *.mpf* file. A project may also consist of:

- HDL source files or references to source files
- other files such as READMEs or other project documentation
- local libraries
- references to global libraries

This lesson uses the Verilog files *tcounter.v* and *counter.v* in the examples. If you have a VHDL license, use *tcounter.vhd* and *counter.vhd* instead.

Related reading

ModelSim User's Manual, [Chapter 2 - Projects](#) (UM-27)

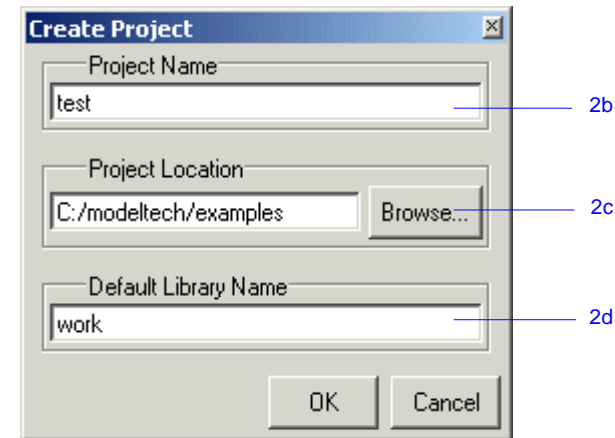
Creating a new project

- 1 If you just finished the previous lesson, ModelSim should already be running. If not, start ModelSim.
 - a Type **vsim** at a UNIX shell prompt or use the ModelSim icon in Windows.
- 2 Create a new project.
 - a Select **Create a Project** from the Welcome dialog *or* **File > New > Project** (Main window) from the menu bar.

This opens a dialog where you enter a Project Name, Project Location (i.e., directory), and Default Library Name (Figure 14). The default library is where compiled design units will reside.
 - b Type **test** in the Project Name field.
 - c Click **Browse** to select a directory where the project file will be stored.
 - d Leave the Default Library Name set to *work*.
 - e Click **OK**.

If you see the Select Initial Ini dialog, asking which *modelsim.ini* file you would like the project to be created from, select the **Use Default Ini** button.

Figure 14: The Create Project dialog

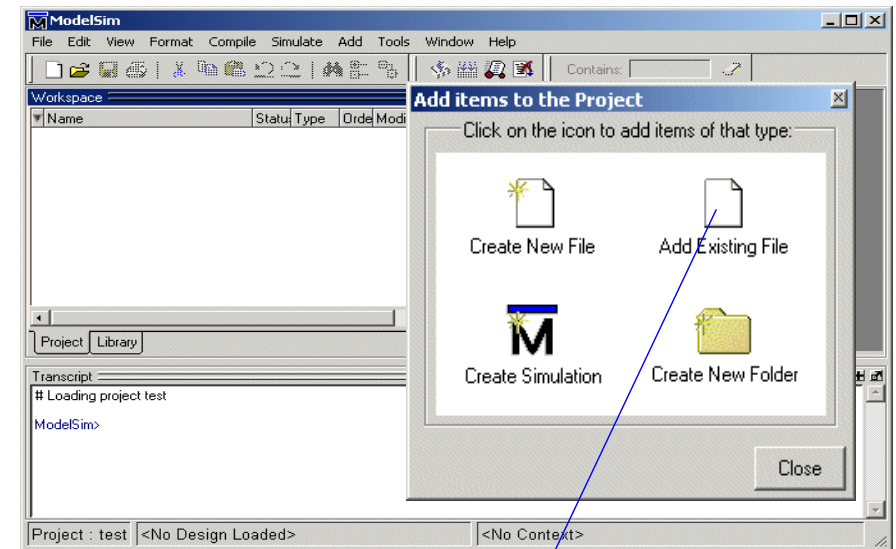


Adding objects to the project

Once you click OK to accept the new project settings, you will see a blank Project tab in the workspace area of the Main window and the Add items to the Project dialog will appear (Figure 15). From this dialog you can create a new design file, add an existing file, add a folder for organization purposes, or create a simulation configuration (discussed below).

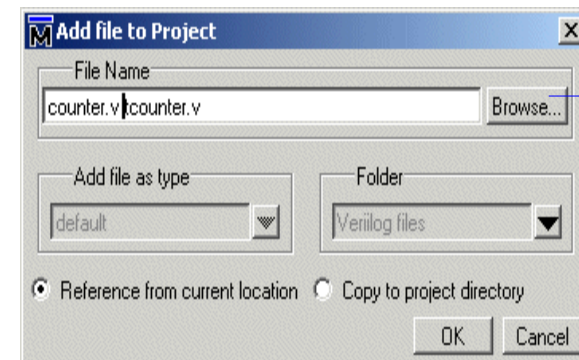
- 1 Add two existing files.
 - a Click **Add Existing File**.
This opens the Add file to Project dialog (Figure 16). This dialog lets you browse to find files, specify the file type, specify which folder to add the file to, and identify whether to leave the file in its current location or to copy it to the project directory.
 - b Click **Browse**.
 - c Open the *examples* directory in your ModelSim installation tree.
 - d **Verilog:** Select *counter.v*, hold the <Ctrl> key down, and then select *tcounter.v*.
VHDL: Select *counter.vhd*, hold the <Ctrl> key down, and then select *tcounter.vhd*.
 - e Click **Open** and then **OK**.
 - f Click **Close** to dismiss the Add items to the Project dialog.

Figure 15: Adding new items to a project



1a

Figure 16: The Add file to Project dialog



1b

You should now see two files listed in the Project tab of the Workspace pane (Figure 17).

Question mark icons (?) in the Status column mean the file hasn't been compiled or the source file has changed since the last successful compile. The other columns identify file type (e.g., Verilog or VHDL), compilation order, and modified date.

Figure 17: Newly added project files display a '?' for status

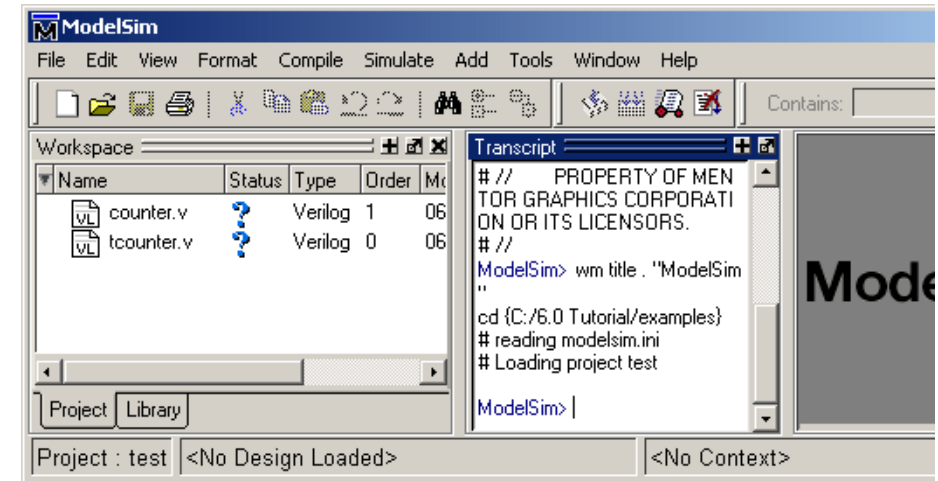
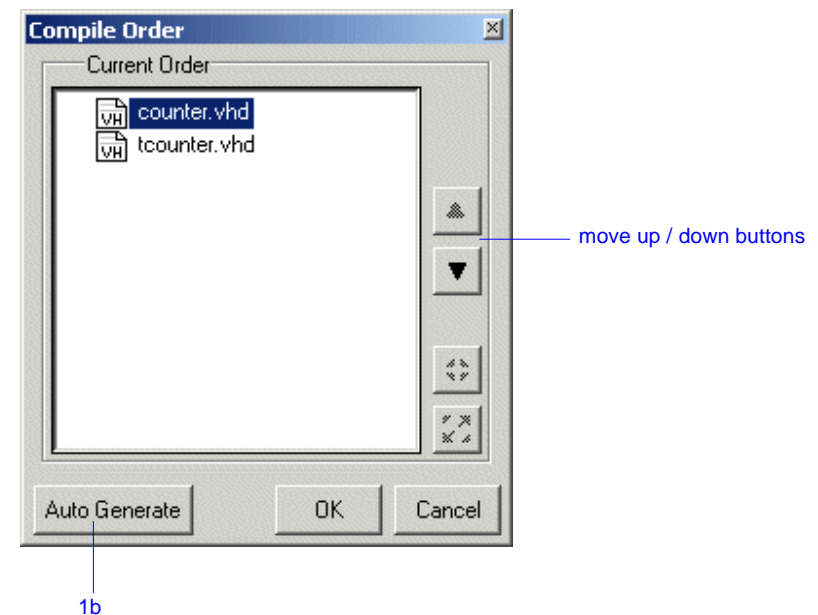


Figure 18: The Compile Order dialog box



Changing compile order (VHDL)

Compilation order is important in VHDL designs. Follow these steps to change compilation order within a project.

1 Change the compile order.

a Select **Compile > Compile Order**.

This opens the Compile Order dialog box (Figure 18).

b Click the **Auto Generate** button.

ModelSim "determines" the compile order by making multiple passes over the files. It starts compiling from the top; if a file fails to compile due to dependencies, it moves that file to the bottom and then recompiles it after compiling the rest of the files. It continues in this manner until all files compile successfully or until a file(s) can't be compiled for reasons other than dependency.

Alternatively, you can select a file and use the Move Up and Move Down buttons to put the files in the correct order.

c Click **OK** to close the Compile Order dialog.

Compiling and loading a design

- 1 Compile the files.
 - a Right-click anywhere in the Project tab and select **Compile > Compile All** from the pop-up menu.

ModelSim compiles both files and changes the symbol in the Status column to a check mark. A check mark means the compile succeeded. If the compile had failed, the symbol would be a red 'X', and you would see an error message in the Transcript pane.

- 2 View the design units.
 - a Click the **Library** tab in the workspace.
 - b Click the "+" icon next to the *work* library.

You should see two compiled design units, their types (modules in this case), and the path to the underlying source files (Figure 19).

- 3 Load the *test_counter* design unit.
 - a Double-click the *test_counter* design unit.

You should see a new tab named *sim* that displays the structure of the *test_counter* design unit (Figure 20). A fourth tab named *Files* contains information about the underlying source files.

At this point you would generally run the simulation and analyze or debug your design like you did in the previous lesson. For now, you'll continue working with the project. However, first you need to end the simulation that started when you loaded *test_counter*.

- 4 End the simulation.
 - a Select **Simulate > End Simulation**.
 - b Click **Yes**.

Figure 19: The Library tab with an expanded library

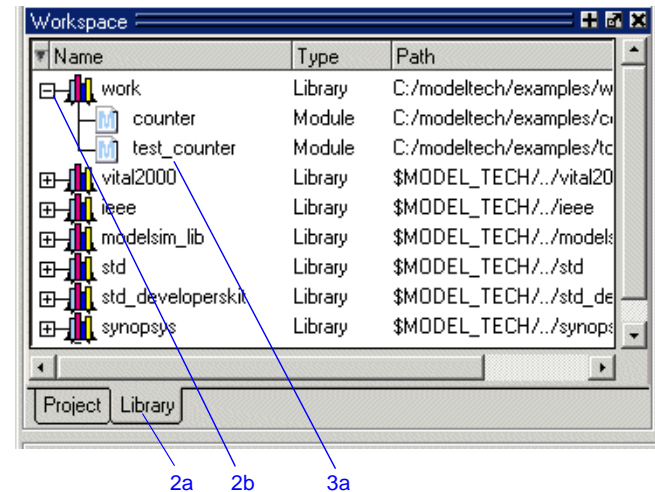
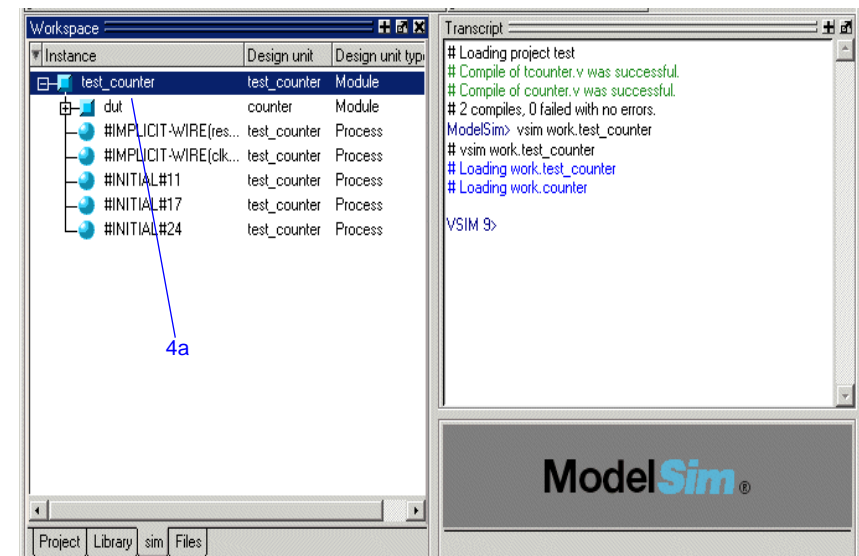


Figure 20: The structure tab for the *counter* design unit



Organizing projects with folders

If you have a lot of files to add to a project, you may want to organize them in folders. You can create folders either before or after adding your files. If you create a folder before adding files, you can specify in which folder you want a file placed at the time you add the file (see Folder field in Figure 16). If you create a folder after adding files, you edit the file properties to move it to that folder.

Adding folders

As shown previously in Figure 15, the Add items to the Project dialog has an option for adding folders. If you have already closed that dialog, you can use a menu command to add a folder.

- 1 Add a new folder.
 - a Select **File > Add to Project > Folder**.
 - b Type **Design Files** in the **Folder Name** field (Figure 21).
 - c Click **OK**.
You'll now see a folder in the Project tab (Figure 22).
- 2 Add a sub-folder.
 - a Right-click anywhere in the Project tab and select **Add to Project > Folder**.
 - b Type **HDL** in the **Folder Name** field (Figure 23).
 - c Click the **Folder Location** drop-down arrow and select *Design Files*.
 - d Click OK.

Figure 21: Adding a new folder to the project

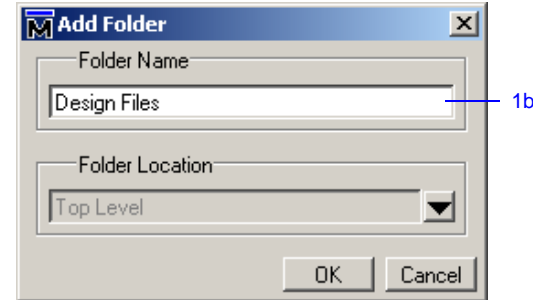


Figure 22: A folder in a project

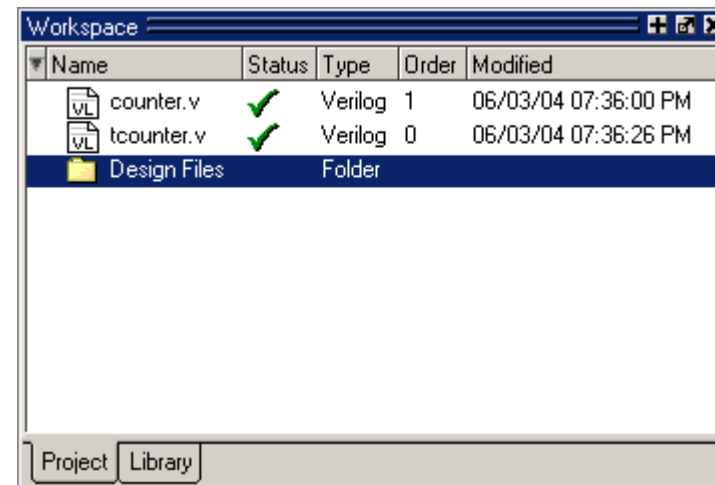
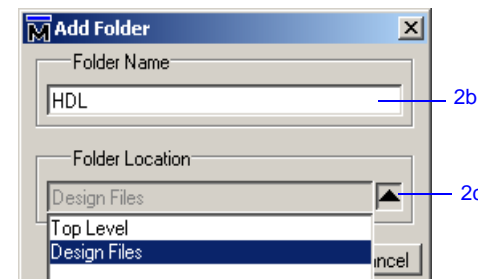


Figure 23: Creating a subfolder



You'll now see a '+' icon next to the *Design Files* folder in the Project tab (Figure 24).

- e Click the '+' icon to see the *HDL* sub-folder.

Moving files to folders

Now that you have folders, you can move the files into them. If you are running on a Windows platform, you can simply drag-and-drop the files into the folder. On Unix platforms, you either have to place the files in a folder when you add the files to the project, or you have to move them using the properties dialog.

- 1 Move *tcounter.v* and *counter.v* to the *HDL* folder.
 - a Select *counter.v*, hold the <Ctrl> key down, and then select *tcounter.v*.
 - b Right-click either file and select **Properties**.

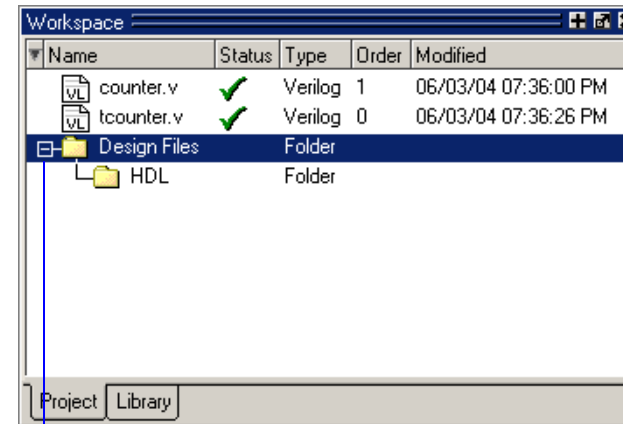
This opens the Project Compiler Settings dialog (Figure 25), which lets you set a variety of options on your design files.

- c Click the **Place In Folder** drop-down arrow and select *HDL*.
- d Click OK.

The two files are moved into the HDL folder. Click the '+' icons on the folders to see the files.

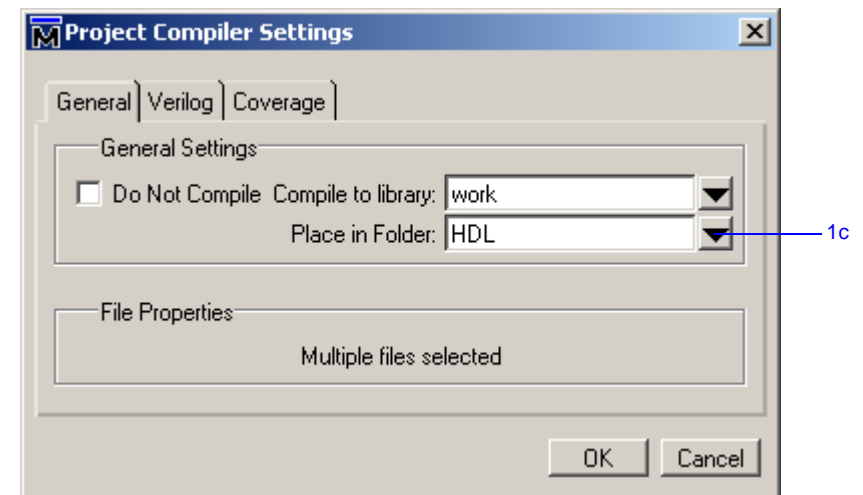
The files are now marked with a '?' icon. Because you moved the files, the project no longer knows if the previous compilation is still valid.

Figure 24: A folder with a sub-folder



2e

Figure 25: Changing file location via the project settings dialog



Simulation Configurations

A Simulation Configuration associates a design unit(s) and its simulation options. For example, say every time you load *tcounter.v* you want to set the simulator resolution to picoseconds (ps) and enable event order hazard checking. Ordinarily you would have to specify those options each time you load the design. With a Simulation Configuration, you specify options for a design and then save a "configuration" that associates the design and its options. The configuration is then listed in the Project tab and you can double-click it to load *counter.v* along with its options.

- 1 Create a new Simulation Configuration.
 - a Select **File > Add to Project > Simulation Configuration**.
This opens the Simulate dialog (Figure 26). The tabs in this dialog present a myriad of simulation options. You may want to explore the tabs to see what's available. You can consult the ModelSim User's Manual to get a description of each option.
 - b Type **counter** in the **Simulation Configuration Name** field.
 - c Select **HDL** from the **Place in Folder** drop-down.
 - d Click the '+' icon next to the *work* library and select *test_counter*.
 - e Click the **Resolution** drop-down and select *ps*.
 - f For Verilog, click the Verilog tab and check **Enable Hazard Checking**.
 - g Click **OK**.

The Project tab now shows a Simulation Configuration named *counter* (Figure 27).

- 2 Load the Simulation Configuration.
 - a Double-click the *counter* Simulation Configuration in the Project tab.
In the Transcript pane of the Main window, the **vsim** (the ModelSim simulator) invocation shows the **-hazards** and **-t ps** switches (Figure 28). These are the command-line equivalents of the options you specified in the Simulate dialog.

Figure 26: The Simulation Configuration dialog

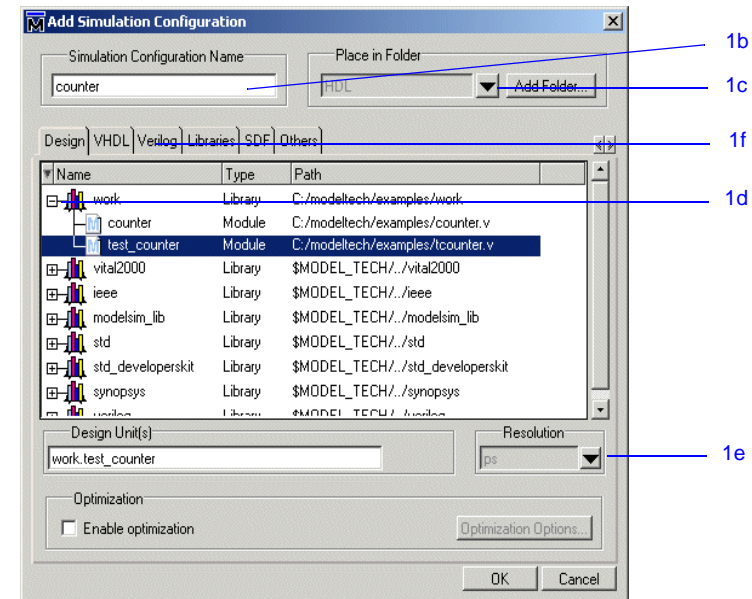
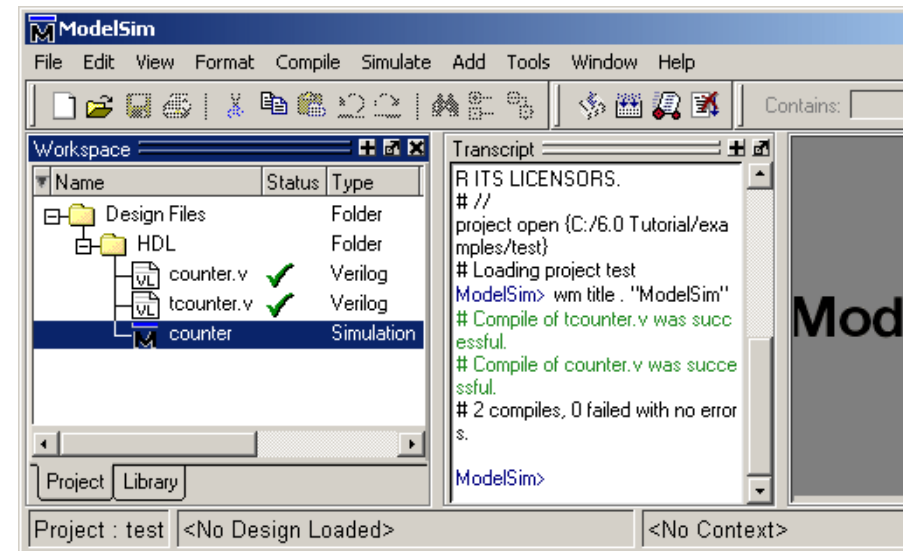


Figure 27: A Simulation Configuration in the Project tab



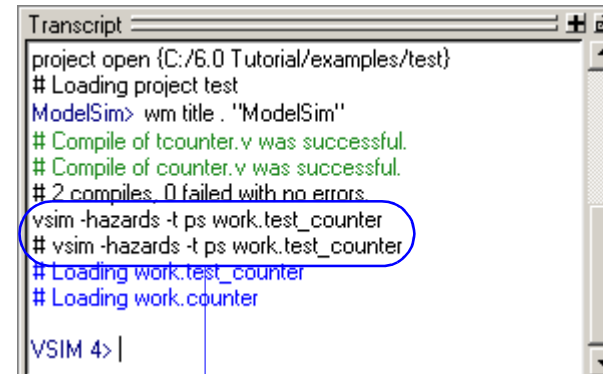
Lesson wrap-up

This concludes this lesson. Before continuing you need to end the current simulation and close the current project.

- 1 Select **Simulate > End Simulation**. Click Yes.
- 2 Select the Project tab in the Main window Workspace.
- 3 Right-click the *test* project to open a context popup menu and select **Close Project**.

If you do not close the project, it will open automatically the next time you start ModelSim.

Figure 28: Transcript shows options used for Simulation Configuration



```
Transcript
project open {C:/6.0 Tutorial/examples/test}
# Loading project test
ModelSim> wm title . "ModelSim"
# Compile of tcounter.v was successful.
# Compile of counter.v was successful.
# 2 compiles, 0 failed with no errors
vsim -hazards -t ps work.test_counter
# vsim -hazards -t ps work.test_counter
# Loading work.test_counter
# Loading work.counter
VSIM 4> |
```

command-line switches

Lesson 4 - Working with multiple libraries

Topics

The following topics are covered in this lesson:

Introduction	T-40
Related reading	T-40
Creating the resource library	T-41
Creating the project	T-43
Linking to the resource library	T-44
Permanently mapping resource libraries	T-47
Lesson wrap-up	T-48

Introduction

In this lesson you will practice working with multiple libraries. As discussed in [Lesson 1 - ModelSim conceptual overview](#), you might have multiple libraries to organize your design, to access IP from a third-party source, or to share common parts between simulations.

You will start the lesson by creating a resource library that contains the *counter* design unit. Next, you will create a project and compile the testbench into it. Finally, you will link to the library containing the counter and then run the simulation.

Design files for this lesson

The sample design for this lesson is a simple 8-bit, binary up-counter with an associated testbench. The pathnames are as follows:

Verilog – *<install_dir>/modeltech/examples/counter.v* and *tcounter.v*

VHDL – *<install_dir>/modeltech/examples/counter.vhd* and *tcounter.vhd*

This lesson uses the Verilog files *tcounter.v* and *counter.v* in the examples. If you have a VHDL license, use *tcounter.vhd* and *counter.vhd* instead.

Related reading

ModelSim User's Manual, [3 - Design libraries](#) (UM-45)

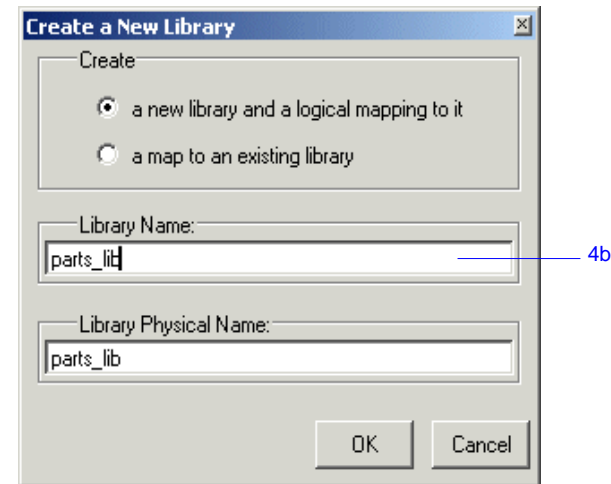
Creating the resource library

- 1 Create a directory for the resource library.
Create a new directory called *resource_library*. Copy *counter.v* from `<install_dir>/modeltech/examples` to the new directory.
- 2 Create a directory for the testbench.
Create a new directory called *testbench* that will hold the testbench and project files. Copy *tcounter.v* from `<install_dir>/modeltech/examples` to the new directory.

You are creating two directories in this lesson to mimic the situation where you receive a resource library from a third-party. As noted earlier, we will link to the resource library in the first directory later in the lesson.
- 3 Start ModelSim and change to the exercise directory.
If you just finished the previous lesson, ModelSim should already be running. If not, start ModelSim.
 - a Type **vsim** at a UNIX shell prompt or use the ModelSim icon in Windows.
If the Welcome to ModelSim dialog appears, click **Close**.
 - b Select **File > Change Directory** and change to the *resource_library* directory you created in step 1.
- 4 Create the resource library.
 - a Select **File > New > Library**.
 - b Type **parts_lib** in the Library Name field (Figure 29).
The Library Physical Name field is filled out automatically.

Once you click OK, ModelSim creates a directory for the library, lists it in the Library tab of the Workspace, and modifies the *modelsim.ini* file to record this new library for the future.

Figure 29: Creating the new resource library



5 Compile the counter into the resource library.

- a Click the Compile icon on the Main window toolbar.
- b Select the *parts_lib* library from the Library list (Figure 30).
- c Double-click *counter.v* to compile it.
- d Click Done.

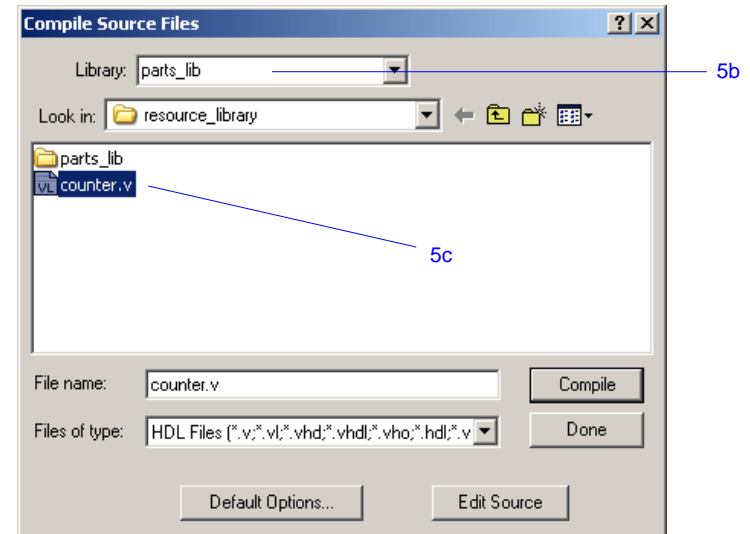


You now have a resource library containing a compiled version of the *counter* design unit.

6 Change to the *testbench* directory.

- a Select **File > Change Directory** and change to the *testbench* directory you created in step 2.

Figure 30: Compiling into the resource library



Creating the project

Now you will create a project that contains *tcounter.v*, the counter's testbench.

- 1 Create the project.
 - a Select **File > New > Project**.
 - b Type **counter** in the Project Name field.
 - c Click **OK**.
 - d If a dialog appears asking about which *modelsim.ini* file to use, click **Use Default Ini**.
- 2 Add the testbench to the project.
 - a Click **Add Existing File** in the Add items to the Project dialog.
 - b Click the Browse button and select *tcounter.v*.
 - c Click Open and then OK.
 - d Click Close to dismiss the Add items to the Project dialog.

The *tcounter.v* file is listed in the Project tab of the Main window.
- 3 Compile the testbench.
 - a Right-click *tcounter.v* and select **Compile > Compile Selected**.

Linking to the resource library

To wrap up this part of the lesson, you will link to the *parts_lib* library you created earlier. But first, try simulating the testbench without the link and see what happens.

ModelSim responds differently for Verilog and VHDL in this situation.

Verilog

- 1 Simulate a Verilog design with a missing resource library.
 - a In the Library tab, click the '+' icon next to the *work* library and double-click *test_counter*.

The Main window Transcript reports an error (Figure 31). When you see a message that contains text like "Error: (vsim-3033)", you can view more detail by using the **verror** command.

- b Type **verror 3033** at the ModelSim> prompt.

The expanded error message tells you that a design unit could not be found for instantiation. It also tells you that the original error message should list which libraries ModelSim searched. In this case, the original message says ModelSim searched only *work*.

VHDL

- 1 Simulate a VHDL design with a missing resource library.
 - a In the Library tab, click the '+' icon next to the *work* library and double-click *test_counter*.

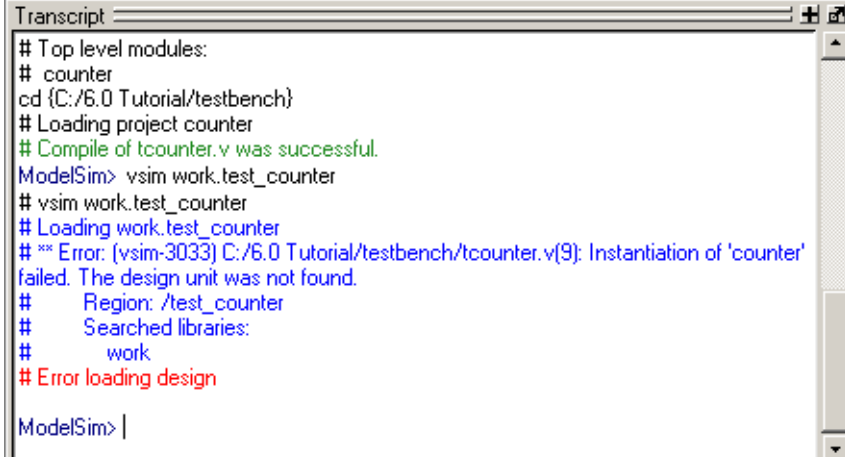
The Main window Transcript reports a warning (Figure 32). When you see a message that contains text like "Warning: (vsim-3473)", you can view more detail by using the **verror** command.

- b Type **verror 3473** at the ModelSim> prompt.

The expanded error message tells you that a component ('dut' in this case) has not been explicitly bound and no default binding can be found.

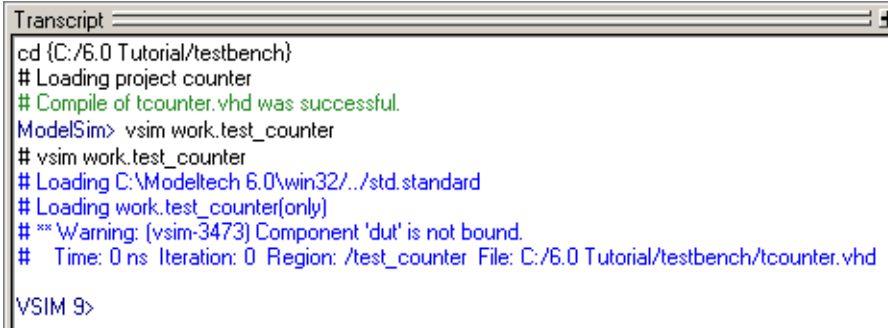
- c Type **quit -sim** to quit the simulation.

Figure 31: Verilog simulation error reported in the Main window



```
Transcript
# Top level modules:
# counter
cd {C:/6.0 Tutorial/testbench}
# Loading project counter
# Compile of tcounter.v was successful.
ModelSim> vsim work.test_counter
# vsim work.test_counter
# Loading work.test_counter
# ** Error: (vsim-3033) C:/6.0 Tutorial/testbench/tcounter.v(9): Instantiation of 'counter'
failed. The design unit was not found.
#   Region: /test_counter
#   Searched libraries:
#     work
# Error loading design
ModelSim> |
```

Figure 32: VHDL simulation warning reported in Main window



```
Transcript
cd {C:/6.0 Tutorial/testbench}
# Loading project counter
# Compile of tcounter.vhd was successful.
ModelSim> vsim work.test_counter
# vsim work.test_counter
# Loading C:\Modeltech 6.0\win32\../std.standard
# Loading work.test_counter(only)
# ** Warning: (vsim-3473) Component 'dut' is not bound.
#   Time: 0 ns Iteration: 0 Region: /test_counter File: C:/6.0 Tutorial/testbench/tcounter.vhd
VSIM 9>
```

The process for linking to a resource library differs between Verilog and VHDL. If you are using Verilog, follow the steps in "[Linking in Verilog](#)" (T-45). If you are using VHDL, follow the steps in "[Linking in VHDL](#)" (T-46) one page later.

Linking in Verilog

Linking in Verilog requires that you specify a "search library" when you invoke the simulator.

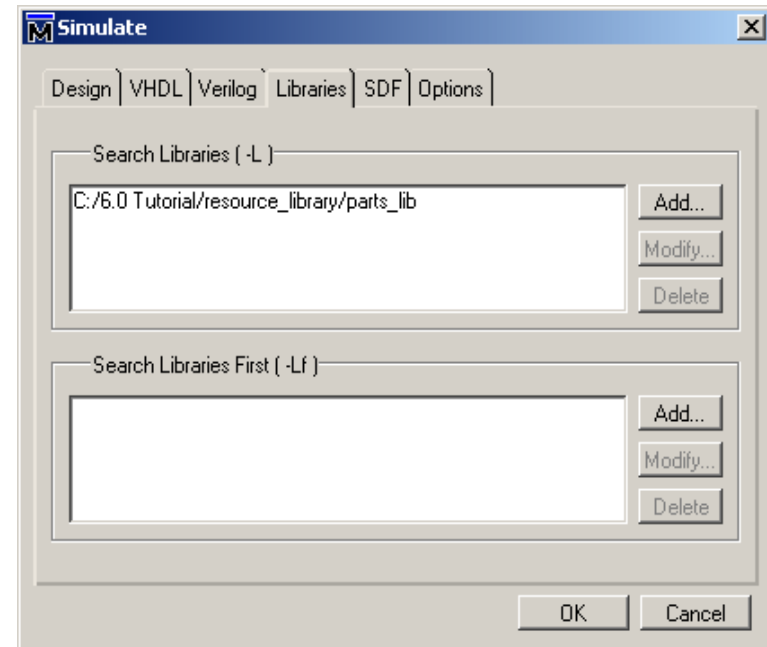
- 1 Specify a search library during simulation.
 - a Click the Simulate icon on the Main window toolbar.
 - b Click the '+' icon next to the *work* library and select *test_counter*.
 - c Click the Libraries tab.
 - d Click the Add button next to the Search Libraries field and browse to *parts_lib* in the first directory you created earlier in the lesson.
 - e Click OK.

The dialog should have *parts_lib* listed in the Search Libraries field ([Figure 33](#)).
 - f Click OK.

The design loads without errors.



Figure 33: Specifying a search library in the Simulate dialog



Linking in VHDL

To link to a resource library in VHDL, you have to create a logical mapping to the physical library and then add LIBRARY and USE statements to the source file.

- 1 Create a logical mapping to *parts_lib*.
 - a Select **File > New > Library**.
 - b In the Create a New Library dialog, select **a map to an existing library**.
 - c Type **parts_lib** in the Library Name field.
 - d Click Browse to open the Select Library dialog and browse to *parts_lib* in the *resource_library* directory you created earlier in the lesson. Click OK to select the library and close the Select Library dialog.
 - e The Create a New Library dialog should look similar to the one shown in [Figure 34](#). Click OK to close the dialog.
- 2 Add LIBRARY and USE statements to *tcounter.vhd*.
 - a In the Library tab of the Main window, click the '+' icon next to the *work* library.
 - b Right-click *test_counter* in the work library and select **Edit**.
This opens the file in the Source window.
 - c Add these two lines to the top of the file:


```
LIBRARY parts_lib;
USE parts_lib.ALL;
```

The testbench source code should now look similar to that shown in [Figure 33](#).
 - d Select **File > Save**.
- 3 Recompile and simulate.
 - a In the Project tab of the Main window, right-click *tcounter.vhd* and select **Compile > Compile Selected**.
 - b In the Library tab, double-click *test_counter* to load the design.
The design loads without errors.

Figure 34: Mapping to the parts_lib library

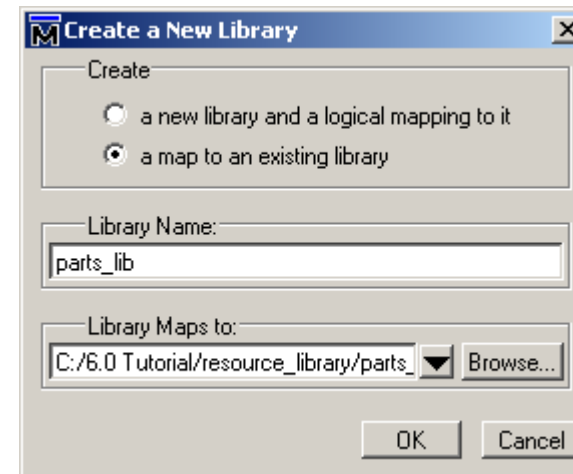
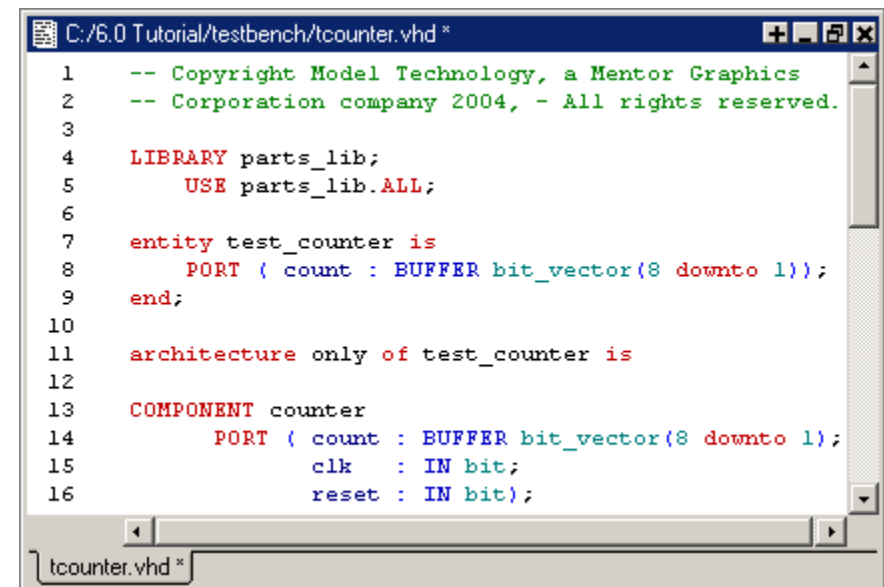


Figure 35: Adding LIBRARY and USE statements to the testbench



Permanently mapping resource libraries

If you reference particular resource libraries in every project or simulation, you may want to permanently map the libraries. Doing this requires that you edit the master *modelsim.ini* file in the installation directory. Though you won't actually practice it in this tutorial, here are the steps for editing the file:

- 1 Locate the *modelsim.ini* file in the ModelSim installation directory (*<install_dir>/modeltech/modelsim.ini*).
- 2 IMPORTANT - Make a backup copy of the file.
- 3 Change the file attributes of *modelsim.ini* so it is no longer "read-only."
- 4 Open the file and enter your library mappings in the [Library] section. For example:

```
parts_lib = C:/libraries/parts_lib
```
- 5 Save the file.
- 6 Change the file attributes so the file is "read-only" again.

Lesson wrap-up

This concludes this lesson. Before continuing we need to end the current simulation and close the project.

- 1 Select **Simulate > End Simulation**. Click Yes.
- 2 Select the Project tab of the Main window Workspace.
- 3 Select **File > Close**. Click OK.

Lesson 5 - Viewing simulations in the Wave window

Topics

The following topics are covered in this lesson:

Introduction	T-50
Related reading	T-50
Loading a design	T-51
Adding objects to the Wave window.	T-52
Using cursors in the Wave window	T-54
Working with a single cursor	T-54
Working with multiple cursors	T-55
Saving the window format	T-57
Lesson wrap-up	T-58

Introduction

The Wave window allows you to view the results of your simulation as HDL waveforms and their values.

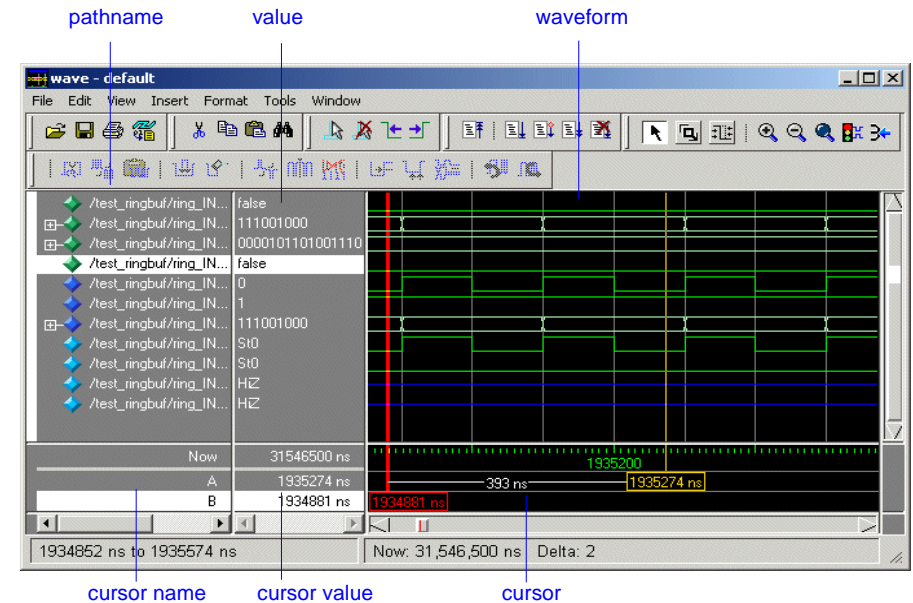
The Wave window is divided into a number of window panes (Figure 36). All window panes in the Wave window can be resized by clicking and dragging the bar between any two panes.

Related reading

ModelSim GUI Reference – "Wave window" (GR-141)

ModelSim User's Manual – [Chapter 6 - WLF files \(datasets\) and virtuals](#) (UM-123)

Figure 36: The Wave window and its many panes



Loading a design

For the examples in this lesson, we have used the design simulated in [Chapter Lesson 2 - Basic simulation](#).

- 1 If you just finished the previous lesson, ModelSim should already be running. If not, start ModelSim.
 - a Type **vsim** at a UNIX shell prompt or use the ModelSim icon in Windows.
If the Welcome to ModelSim dialog appears, click **Close**.
- 2 Load the design.
 - a Select **File > Change Directory** and open the directory you created in Lesson 2.
The *work* library should already exist.
 - b Click the '+' icon next to the *work* library and double-click *test_counter*.
ModelSim loads the design and adds *sim* and *Files* tabs to the Workspace.

Adding objects to the Wave window

ModelSim offers several methods for adding objects to the Wave window. In this exercise, you will try different methods.

1 Add objects from the Objects pane.

- a Select an item in the Objects pane of the Main window, and then select **Add to Wave > Signals in Region**.

ModelSim adds several signals to the Wave window.

- b In the Wave window, select **Edit > Select All** and then **Edit > Delete**.

This deletes all objects in the window.

2 Add objects using drag-and-drop.

You can drag an object to the Wave window from many other windows and panes (e.g., Workspace, Objects, and Locals).

- a Drag an instance from the *sim* tab of the Main window to the Wave window.

ModelSim adds the objects for that instance to the Wave window.

- b Drag a signal from the Objects pane to the Wave window.
- c In the Wave window, select **Edit > Select All** and then **Edit > Delete**.

3 Add objects using a command.

- a Type **add wave *** at the VSIM> prompt.

ModelSim adds all objects from the current region.

- b Run the simulation for awhile so you can see waveforms.

Zooming the waveform display

Zooming lets you change the display range in the waveform pane. There are numerous methods for zooming the display.

1 Zoom the display using various techniques.

- a Click the Zoom Mode icon on the Wave window toolbar.



- b In the waveform pane, click and drag down and to the right.

You should see blue vertical lines and numbers defining an area to zoom in (Figure 37).

- c Select **View > Zoom > Zoom Last**.

The waveform pane returns to the previous display range.

- d Click the Zoom In 2x icon a few times.



- e In the waveform pane, click and drag up and to the right.

You should see a blue line and numbers defining an area to zoom out (Figure 38).

- f Select **View > Zoom > Zoom Full**.

Figure 37: Zooming in with the mouse pointer

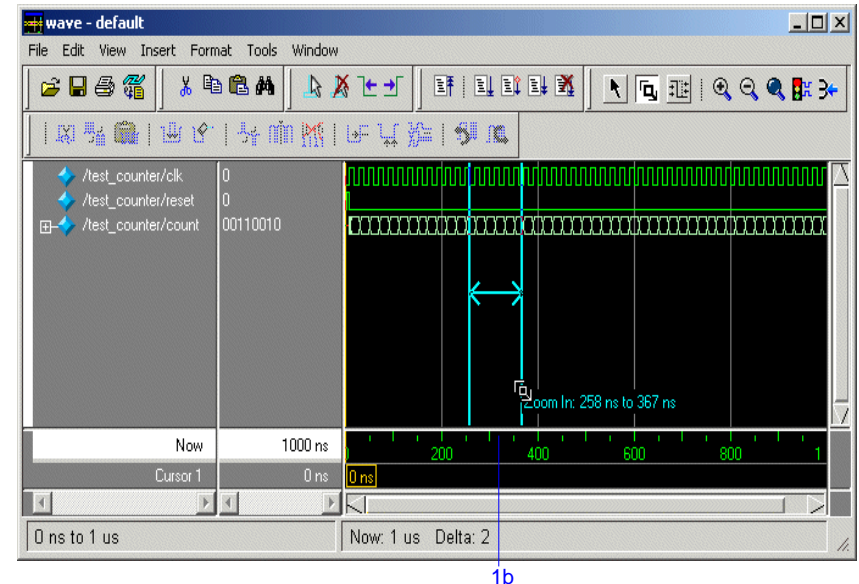
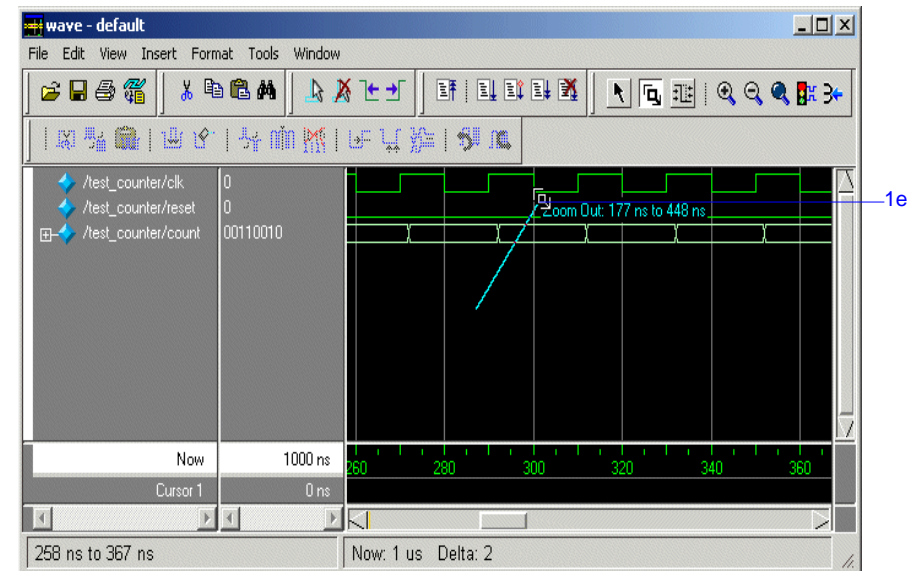


Figure 38: Zooming out with the mouse pointer



Using cursors in the Wave window

Cursors mark simulation time in the Wave window. When ModelSim first draws the Wave window, it places one cursor at time zero. Clicking anywhere in the waveform pane brings that cursor to the mouse location.

You can also add additional cursors; name, lock, and delete cursors; use cursors to measure time intervals; and use cursors to find transitions.

Working with a single cursor

- 1 Position the cursor by clicking and dragging.
 - a Click the Select Mode icon on the Wave window toolbar.
 - b Click anywhere in the waveform pane.

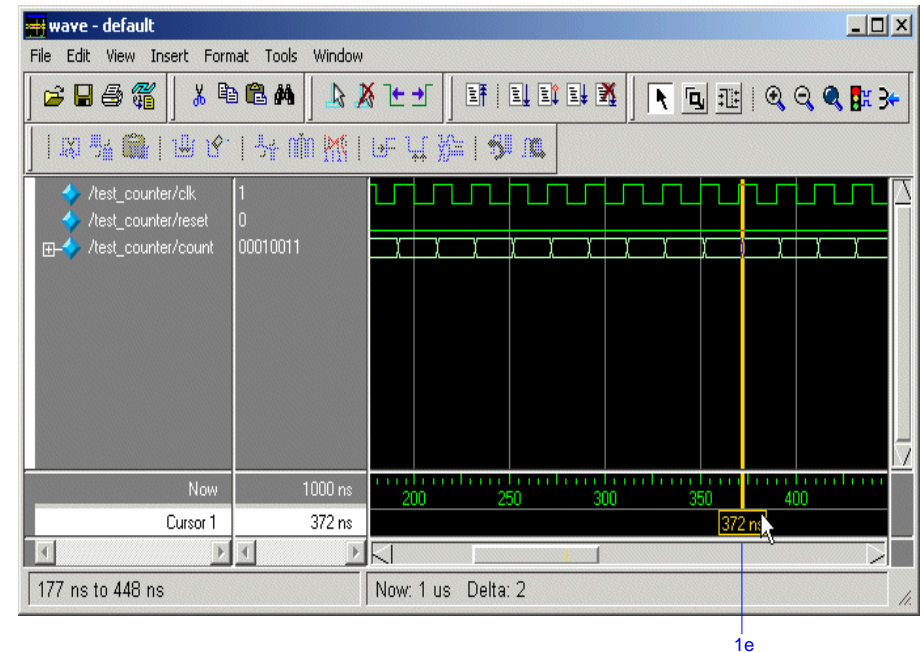
A cursor is inserted at the time where you clicked ([Figure 39](#)).
 - c Drag the cursor and observe the value pane.

The signal values change as you move the cursor. This is perhaps the easiest way to examine the value of a signal at a particular time.
 - d In the waveform pane, drag the cursor to the right of a transition with the mouse positioned over a waveform.

The cursor "snaps" to the transition. Cursors "snap" to a waveform edge if you click or drag a cursor to within ten pixels of a waveform edge. You can set the snap distance in the Window Preferences dialog (select **Tools > Window Preferences**).
 - e In the cursor pane, drag the cursor to the right of a transition ([Figure 39](#)).

The cursor doesn't snap to a transition if you drag in the cursor pane.

Figure 39: Working with a single cursor in the Wave window



- 2 Rename the cursor.
 - a Right-click "Cursor 1" in the cursor name pane, and select and delete the text (Figure 40).
 - b Type **A** and press Enter.
The cursor name changes to "A".
- 3 Jump the cursor to the next or previous transition.
 - a Click signal *count* in the pathname pane.
 - a Click the Find Next Transition icon on the Wave window toolbar.



The cursor jumps to the next transition on the currently selected signal.

- b Click the Find Previous Transition icon on the Wave window toolbar.



The cursor jumps to the previous transition on the currently selected signal.

Working with multiple cursors

- 1 Add a second cursor.
 - a Click the Add Cursor icon on the Wave window toolbar.



- b Right-click the name of the new cursor and delete the text.
- c Type **B** and press Enter.
- d Drag cursor *B* and watch the interval measurement change dynamically (Figure 41).

Figure 40: Renaming a cursor

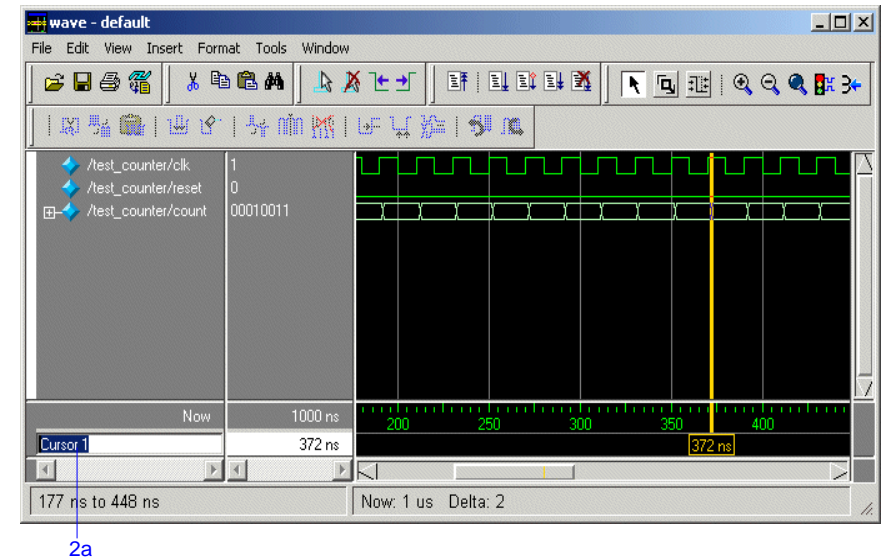
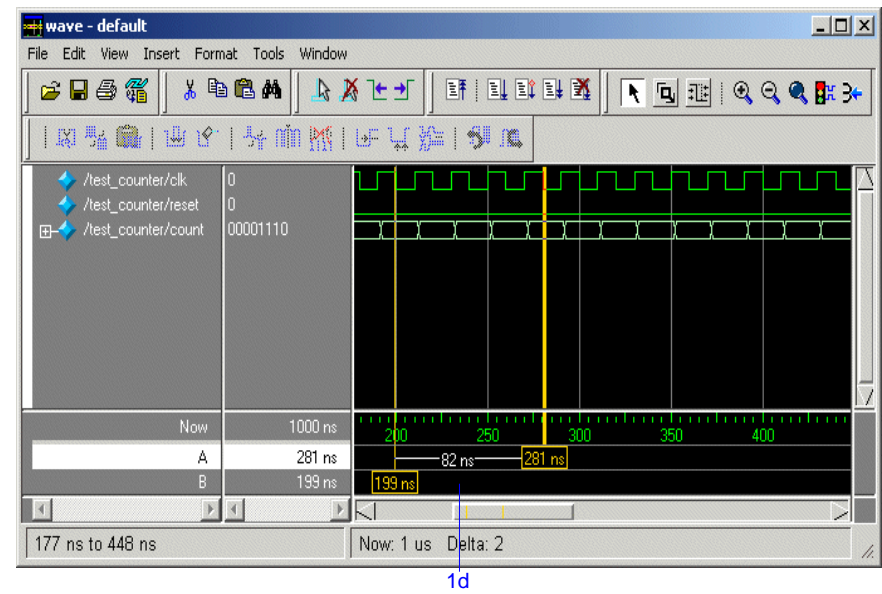


Figure 41: Interval measurement between two cursors

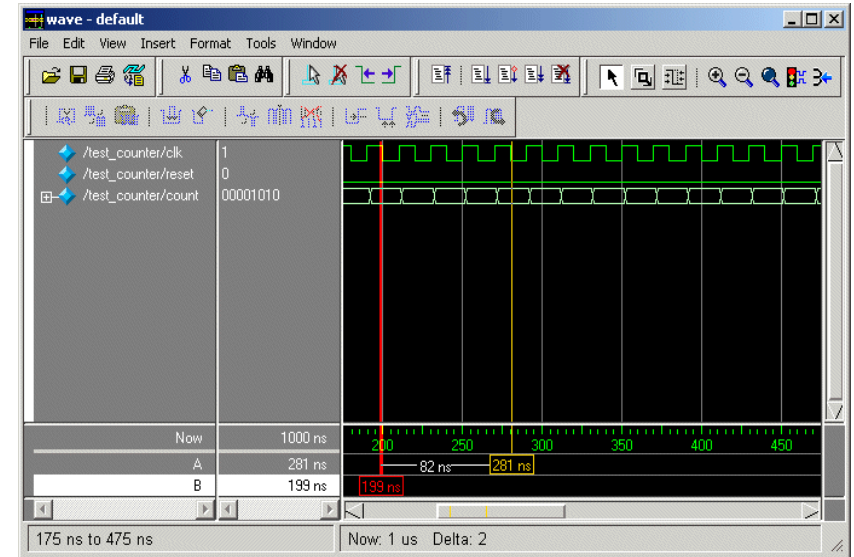


T-56 Lesson -

- 2 Lock cursor *B*.
 - a Right-click cursor *B* in the cursor pane and select **Lock B**.

The cursor color changes to red and you can no longer drag the cursor (Figure 42).
- 3 Delete cursor *B*.
 - a Right-click cursor *B* and select **Delete B**.

Figure 42: A locked cursor in the Wave window



Saving the window format

If you close the Wave window, any configurations you made to the window (e.g., signals added, cursors set, etc.) are discarded. However, you can use the Save Format command to capture the current Wave window display and signal preferences to a DO file. You open the DO file later to recreate the Wave window as it appeared when the file was created.

Format files are design-specific; use them only with the design you were simulating when they were created.

- 1 Save a format file.
 - a Select **File > Save > Format**.
 - b Leave the file name set to *wave.do* and click **Save**.
 - c Close the Wave window.
- 2 Load a format file.
 - a In the Main window, select **View > Debug Windows > Wave**.
All signals and cursor(s) that you had set are gone.
 - b In the Wave window, select **File > Open > Format**.
 - c Select *wave.do* and click **Open**.
ModelSim restores the window to its previous state.
 - d Close the Wave window when you are finished by selecting **File > Close**.

Lesson wrap-up

This concludes this lesson. Before continuing we need to end the current simulation.

- 1 Select **Simulate > End Simulation**. Click Yes.

Lesson 6 - Viewing and initializing memories

Topics

The following topics are covered in this lesson:

Introduction	T-60
Related reading	T-60
Compiling and loading the design	T-61
Viewing a memory	T-62
Navigating within the memory	T-64
Saving memory contents to a file	T-66
Initializing a memory	T-67
Interactive debugging commands	T-69
Lesson Wrap-up	T-71

Introduction

In this lesson you will learn how to view and initialize memories in ModelSim. ModelSim defines and lists as memories any of the following:

- reg, wire, and std_logic arrays
- Integer arrays
- Single dimensional arrays of VHDL enumerated types other than std_logic

Design files for this lesson

The ModelSim installation comes with Verilog and VHDL versions of the example design. The files are located in the following directories:

Verilog – *<install_dir>/modeltech/examples/memory/verilog*

VHDL – *<install_dir>/modeltech/examples/memory/vhdl*

This lesson uses the Verilog version for the exercises. If you have a VHDL license, use the VHDL version instead.

Related reading

ModelSim GUI Reference – "[Memory windows](#)" (GR-109)

ModelSim Command Reference – [mem display](#) (CR-101), [mem load](#) (CR-104), [mem save](#) (CR-107), [radix](#) (CR-128) commands

Compiling and loading the design

- 1 Create a new directory and copy the tutorial files into it.

Start by creating a new directory for this exercise (in case other users will be working with these lessons). Create the directory and copy all files from `<install_dir>/examples/memory/verilog` to the new directory.

If you have a VHDL license, copy the files in `<install_dir>/examples/memory/vhdl` instead.

- 2 Start ModelSim and change to the exercise directory.

If you just finished the previous lesson, ModelSim should already be running. If not, start ModelSim.

- a Type **vsim** at a UNIX shell prompt or use the ModelSim icon in Windows.

If the Welcome to ModelSim dialog appears, click **Close**.

- b Select **File > Change Directory** and change to the directory you created in step 1.

- 3 Create the working library and compile the design.

- a Type **vlib work** at the ModelSim> prompt.

- b **Verilog:**

Type **vlog sp_syn_ram.v dp_syn_ram.v ram_tb.v** at the ModelSim> prompt.

VHDL:

Type **vcom -93 sp_syn_ram.vhd dp_syn_ram.vhd ram_tb.vhd** at the ModelSim> prompt.

Type **set NumericStdNoWarnings 1** at the ModelSim> prompt to suppress NumericStd warnings encountered during simulation.

- 4 Load the design.

- a On the Library tab of the Main window Workspace, click the "+" icon next to the *work* library.

- b Double-click the *ram_tb* design unit to load the design.

Viewing a memory

Memories can be viewed via the ModelSim GUI.

- 1 Open a Memory instance.
 - a Select **View > Debug Windows > Memory**.

The Memories tab opens in the Workspace pane (Figure 43) and lists the memories in the current design context (*ram_tb*) with the range, depth, and width of each memory.
 - b VHDL: The radix for enumerated types is Symbolic. To change the radix to binary for the purposes of this lesson, type the following command at the vsim prompt:
VSIM> radix bin
 - c Double-click the */ram_tb/spram1/mem* instance in the memories list to view its contents. A **mem** tab is created in the MDI frame to display the memory contents. The data are all **X (0 in VHDL)** since you have not yet simulated the design. The first column (blue hex characters) lists the addresses (Figure 44), and the remaining columns show the data values.
 - d In the Memories tab of the Workspace, double-click instance */ram_tb/spram2/mem*.

This creates a new tab in the MDI frame called **mem(1)** that contains the addresses and data for the *spram2* instance. Each time you double-click a new memory instance in the Workspace, a new tab is created for that instance in the MDI frame.

Figure 43: Viewing the memories tab in the Main window workspace

Instance	Range	Depth	Width
/ram_tb/spram1/mem	[0:4095]	4096	8
/ram_tb/spram2/mem	[0:2047]	2048	17
/ram_tb/spram3/mem	[0:65535]	65536	32
/ram_tb/spram4/mem	[0:3]	4	16
/ram_tb/dpram1/mem	[0:15]	16	8

Library sim Files Memories

Figure 44: The mem tab in the MDI pane shows instance /ram_tb/spram1/mem

[illegible]

- 2 Simulate the design.
 - a Click the **run -all** icon in the Main window.
 - b Click the **mem** tab of the MDI frame to bring the `/ram_tb/spram1/mem` instance to the foreground (Figure 45).



VHDL:

In the Transcript pane, you will see an assertion failure that is functioning to stop the simulation. The simulation itself has not failed.

- 3 Let's change the address radix and the number of words per line for instance `/ram_tb/spram1/mem`.
 - a Right-click anywhere in the **mem** tab and select **Properties**.
The Properties dialog box opens (Figure 46).
 - b For the **Address Radix**, select **Decimal**.
 - c Select **Words per line** and type **1** in the field.
 - d Click OK.

You can see the results of the settings in Figure 47.

Figure 45: Memory display updates with simulation

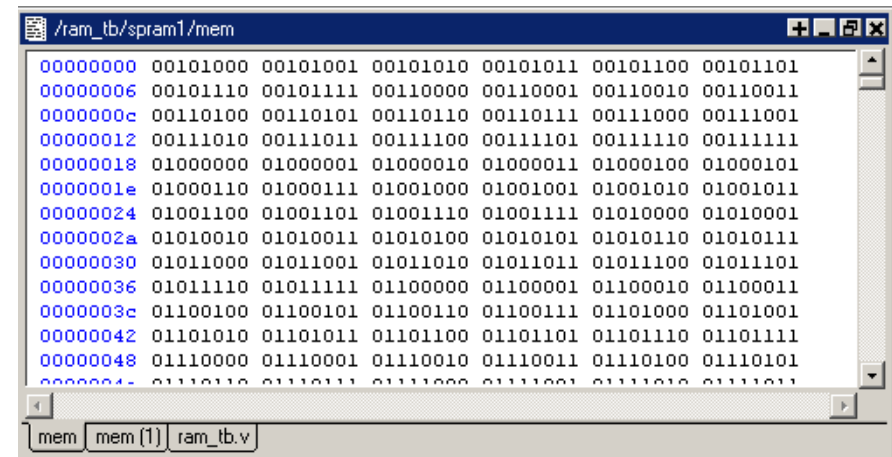
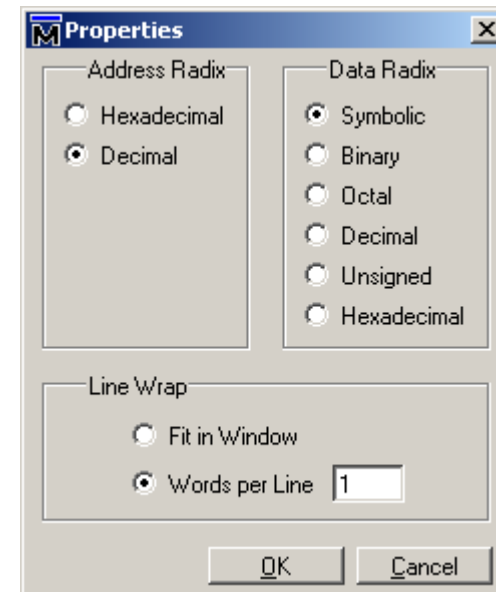


Figure 46: Changing the address radix



Navigating within the memory

You can navigate to specific memory address locations, or to locations containing particular data patterns. First, you will go to a specific address.

- 1 Use Goto to find a specific address.
 - a Right-click anywhere in address column and select **Goto**.
The Goto dialog box opens in the data pane.
 - b Type **30** in the dialog box (Figure 48).
 - c Click OK.

The requested address appears in the top line of the window.

Figure 47: Memory window: new address radix and line length

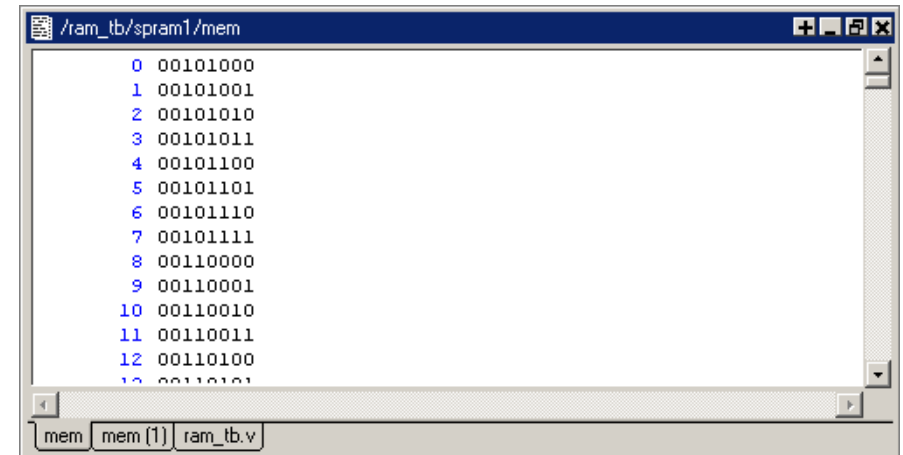
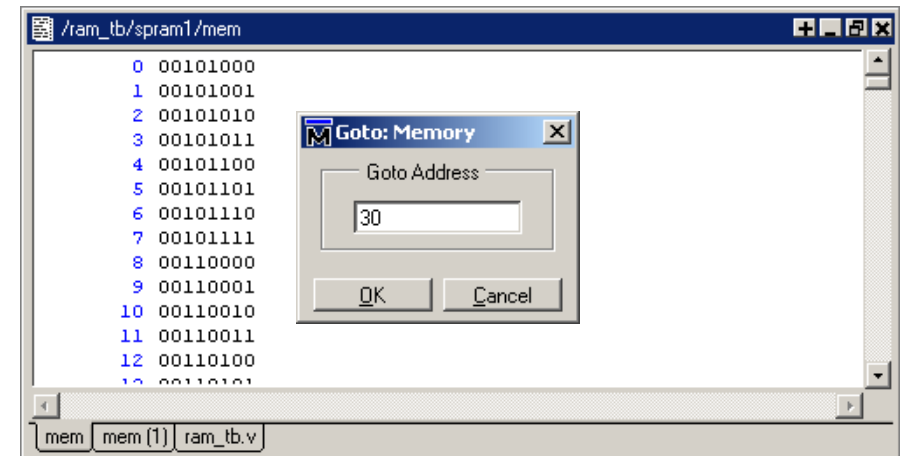


Figure 48: The Goto dialog box



- 2 Edit the address location directly.

To quickly move to a particular address, do the following:

- a Double click any address in the address column.
- b Enter any desired address. (Figure 49)
- c Press <Enter> on your keyboard.

The pane scrolls to that address.

- 3 Now, let's find a particular data entry.

- a Right-click anywhere in the data column and select **Find**.

The Find in dialog box opens (Figure 50).

- b Type **11111010** in the **Find data:** field and click **Find Next**.

The data scrolls to the first occurrence of that address. Click **Find Next** a few more times to search through the list.

- c Click Close to close the dialog box.

Figure 49: Edit the address directly

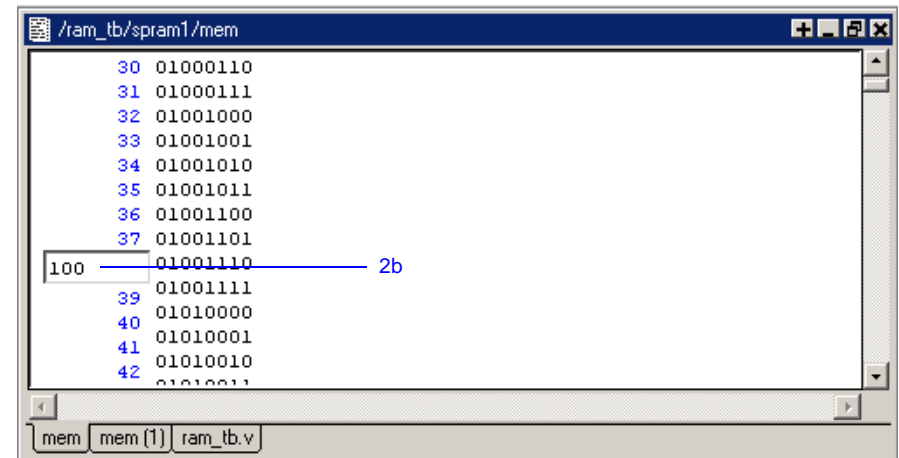
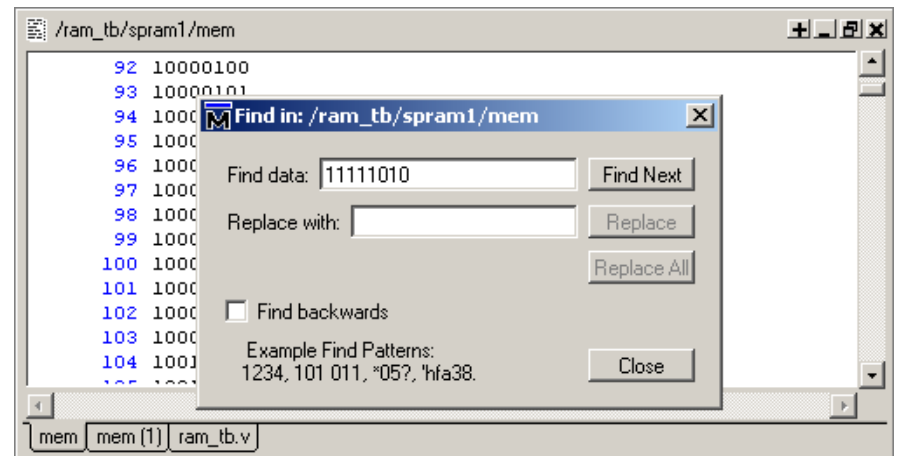


Figure 50: Find in: searching for data value



Saving memory contents to a file

You can save memory contents to a file that can be loaded at some later point in simulation.

- 1 Save a memory pattern from the `/ram_tb/spram1/mem` instance to a file.
 - a Make sure `/ram_tb/spram1/mem` is open and selected in the MDI frame.
 - b Select **File > Save** to bring up the Save Memory dialog box (Figure 51).
 - c For the Address Radix, select **Decimal**.
 - d For the Data Radix, select **Binary**.
 - e Type `data_mem.mem` into the Filename field.
 - f Click OK.

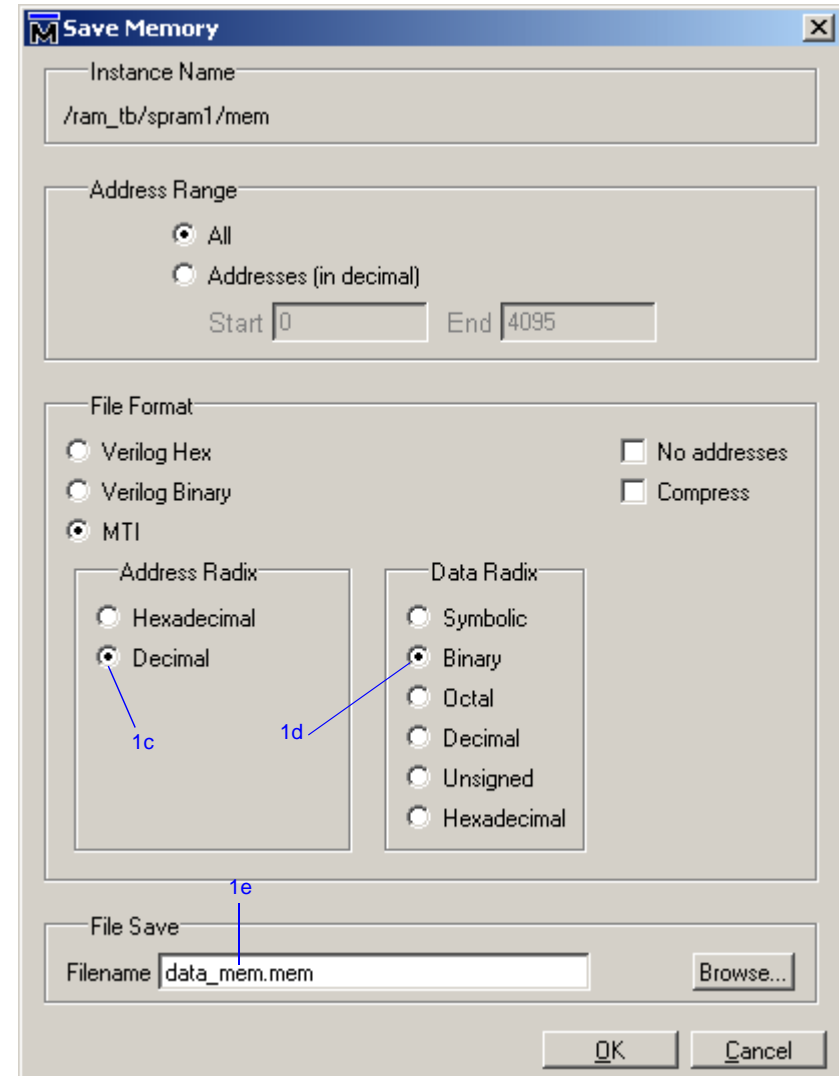
You can view the saved file in any editor.

Memory pattern files can be saved as relocatable files, simply by leaving out the address information. Relocatable memory files can be loaded anywhere in a memory because no addresses are specified.

- 2 Save a relocatable memory pattern file.
 - a Select the **mem(1)** tab in the MDI pane to see the data for the `/ram_tb/spram2/mem` instance.
 - b Right-click in the **mem(1)** tab to open a popup menu and select **Properties**.
 - c In the Properties dialog, set the Address Radix to Decimal and the Data Radix to Binary. Click OK to accept the changes and close the dialog.
 - d Select **File > Save** to bring up the Save Memory dialog box.
 - e Specify a Start address of **0** and End address of **250**.
 - f For Address Radix select Decimal, and for Data Radix select Binary.
 - g Click **No addresses** to create a memory pattern that you can use to relocate somewhere else in the memory, or in another memory.
 - h Enter the file name as `reloc.mem`, then click OK to save the memory contents and close the dialog.

You will use this file for initialization in the next section.

Figure 51: Save Memory dialog box



Initializing a memory

In ModelSim, it is possible to initialize a memory using one of three methods: from a saved memory file, from a fill pattern, or from both.

First, let's initialize a memory from a file only. You will use one you saved previously, *data_mem.mem*.

- 1 View instance */ram_tb/spram3/mem*.
 - a Double-click the */ram_tb/spram3/mem* instance in the Memories tab. This will open a new tab – **mem(2)** – in the MDI frame to display the contents of */ram_tb/spram3/mem*. Scan these contents so you can identify changes once the initialization is complete.
 - b Right-click and select **Properties** to bring up the Properties dialog.
 - c Change the Address Radix to Decimal and click OK.
- 2 Initialize *spram3* from a file.
 - a Right-click anywhere in the data column and select **Load** to bring up the Load Memory dialog box (Figure 52).
The default Load Type is File Only.
 - b Type *data_mem.mem* in the Filename field.
 - c Click OK.

The addresses in instance */ram_tb/spram3/mem* are updated with the data from *data_mem.mem* (Figure 53).

Figure 52: Load Memory dialog box

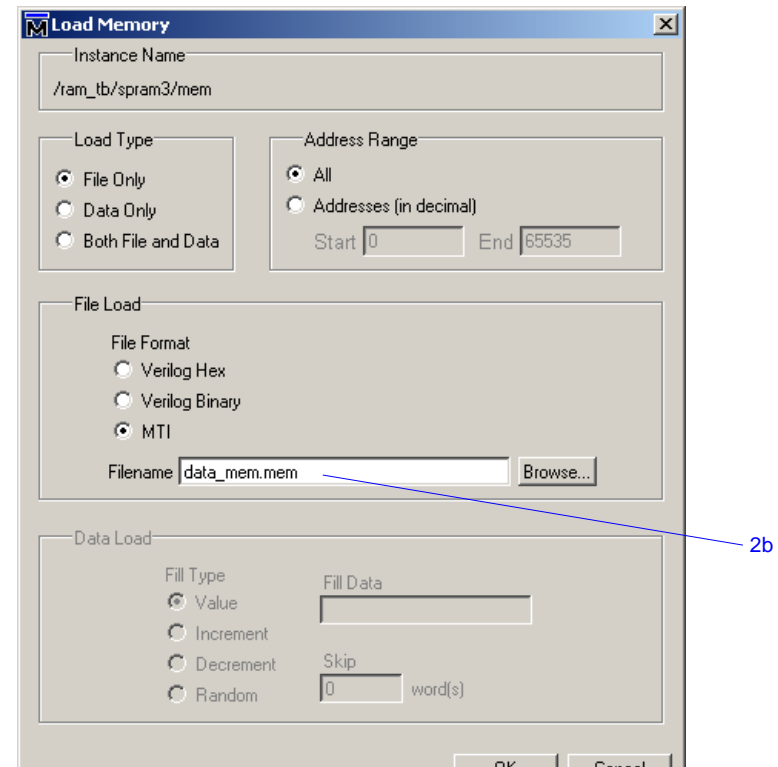
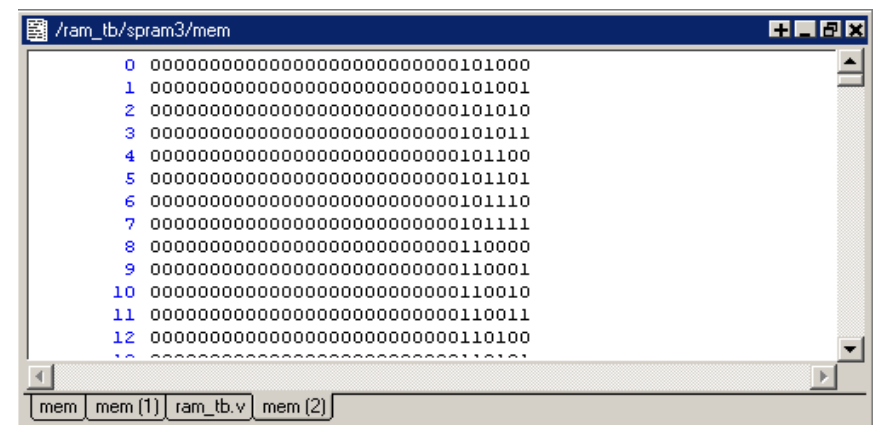


Figure 53: Initialized memory from file and fill pattern



In this next step, you will experiment with loading from both a file and a fill pattern. You will initialize *spram3* with the 250 addresses of data you saved previously into the relocatable file *reloc.mem*. You will also initialize 50 additional address entries with a fill pattern.

- 3 Load the `/ram_tb/spram3/mem` instance with a relocatable memory pattern (`reloc.mem`) and a fill pattern.
 - a Right-click in the data column of the **mem(2)** tab and select **Load** to bring up the Load Memory dialog box (Figure 54).
 - b For Load Type, select **Both File and Data**.
 - c For Address Range, select **Addresses** and enter **0** as the Start address and **300** as the End address.

This means that you will be loading the file from 0 to 300. However, the `reloc.mem` file contains only 251 addresses of data. Addresses 251 to 300 will be loaded with the fill data you specify next.
 - d For File Load, enter **reloc.mem** in the Filename field.
 - e For Data Load, select a Fill Type of **Increment**.
 - f In the Fill Data field, set the seed value of **0** for the incrementing data.
 - g Click OK.
 - h View the data near address 250 by double-clicking on any address in the Address column and entering **250**.

You can see the specified range of addresses overwritten with the new data. Also, you can see the incrementing data beginning at address 251 ([Figure 55](#)).

Now, before you leave this section, go ahead and clear the instances already being viewed.

- 4 Right-click in the **mem(2)** tab and select **Close All**.

Figure 54: Loading a relocatable memory file

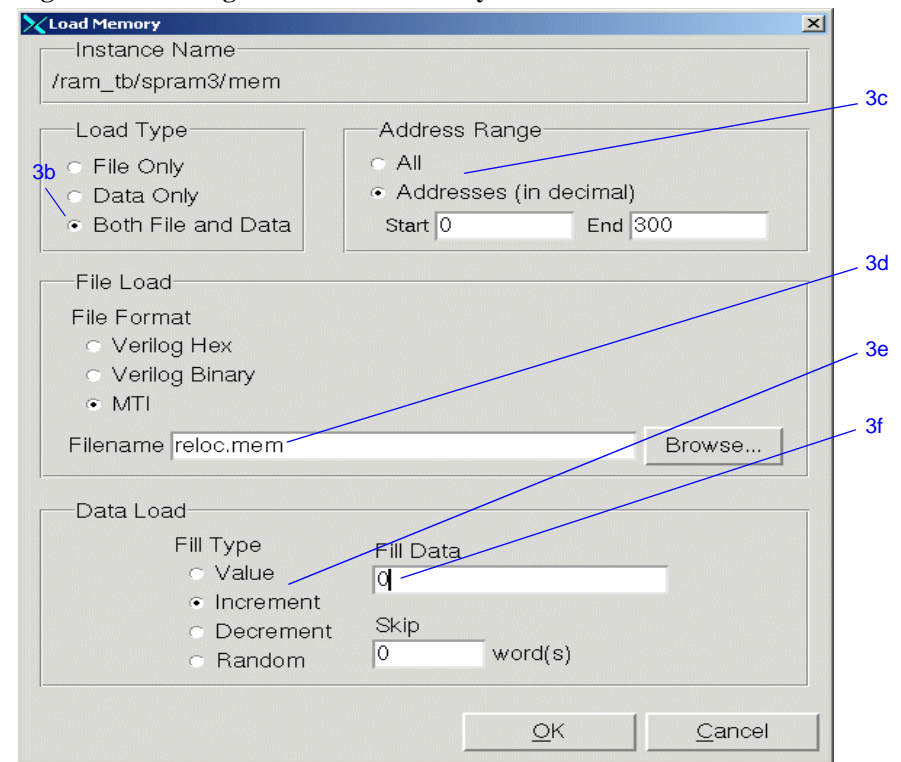
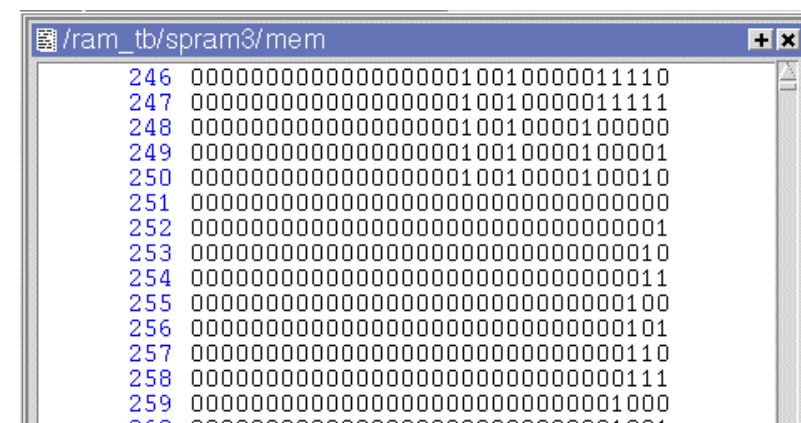


Figure 55: Overwritten values in memory instance



Interactive debugging commands

The memory panes can also be used interactively for a variety of debugging purposes. The features described in this section are useful for this purpose.

- 1 Open a memory instance and change its display characteristics.
 - a Double-click instance `/ram_tb/dpram1/mem` in the Memories tab.
 - b Right-click in the **mem** tab and select **Properties**.
 - c Change the Data Radix to **Hexadecimal**.
 - d Select **Words per line** and enter **2**.
 - e Click OK.
- 2 Initialize a range of memory addresses from a fill pattern.
 - a Right-click in the data column of the **mem** tab and select **Change** to open the Change Memory dialog (Figure 57).
 - b Click the **Addresses** radio button and enter the start address as **0x00000006** and the end address as **0x00000009**. The "0x" hex notation is optional.
 - c Select **Random** as the **Fill Type**.
 - d Enter **0** as the **Fill Data**, setting the seed for the Random pattern.
 - e Click OK.

The data in the specified range are replaced with a generated random fill pattern (Figure 58).

Figure 56: Original memory contents

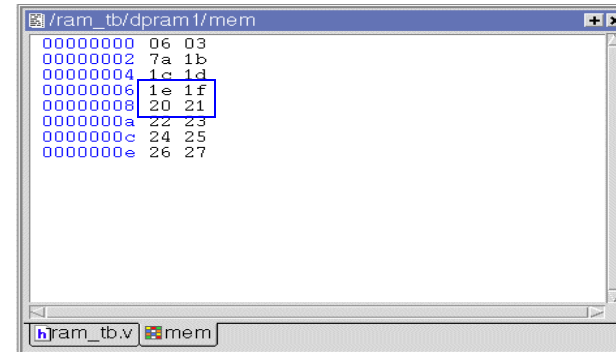


Figure 57: Changing memory contents for a range of addresses

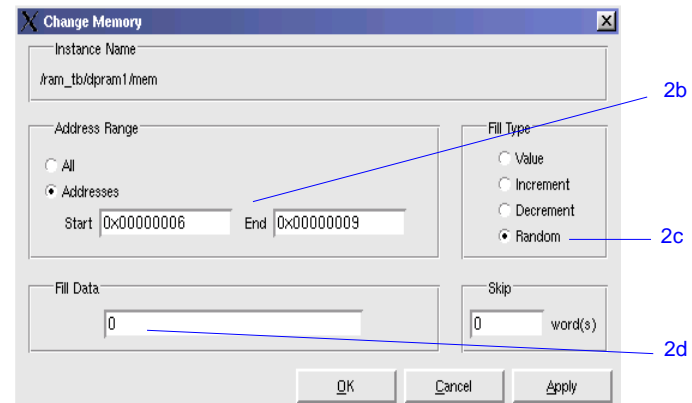
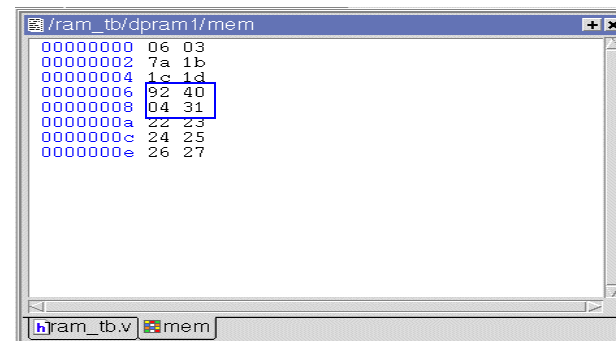


Figure 58: Random contents of a range of addresses



3 Change contents by highlighting.

You can also change data by highlighting them in the Address Data pane.

- Highlight the data for the addresses **0x0000000c:0x0000000e**, as shown in [Figure 59](#).
- Right-click the highlighted data and select **Change**.
This brings up the Change dialog box ([Figure 60](#)). Note that the Addresses field is already populated with the range you highlighted.
- Select **Value** as the Fill Type.
- Enter the data values into the Fill Data field as follow: **34 35 36**
- Click OK.

The data in the address locations change to the values you entered ([Figure 61](#)).

Figure 59: Changing contents by highlighting

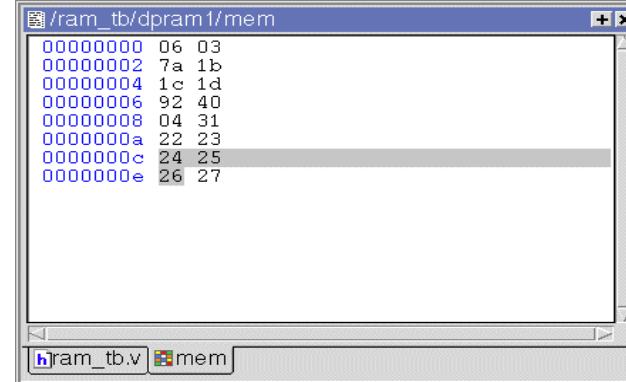


Figure 60: Entering data to change

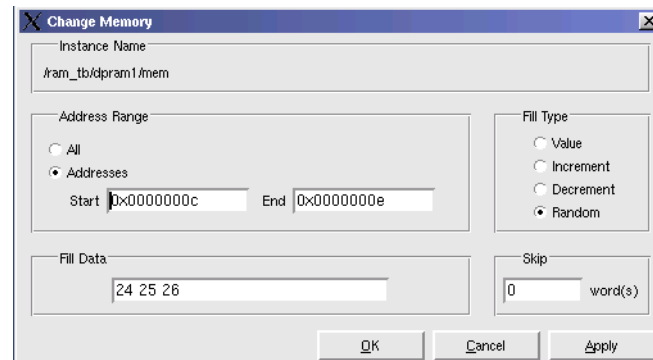
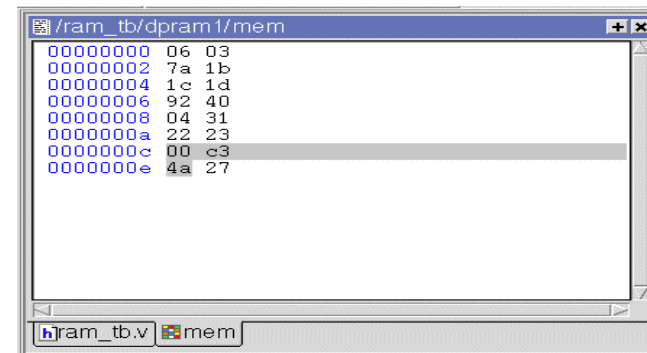


Figure 61: Changed contents for specified addresses



4 Edit data in place.

To edit only one value at a time, do the following:

- Double click any value in the Data column.
- Enter the desired value and press <Enter>.
- When you are finished editing all values, press the <Enter> key on your keyboard to exit the editing mode.

If you needed to cancel the edit function, press the <Esc> key on your keyboard.

Lesson Wrap-up

This concludes this lesson. Before continuing we need to end the current simulation.

- 1 Select **Simulate > End Simulation**. Click Yes.

Lesson 7 - Automating ModelSim

Topics


The following topics are covered in this lesson:

Introduction	T-74
Related reading	T-74
Creating a simple DO file	T-75
Running ModelSim in command-line mode	T-77
Using Tcl with ModelSim	T-80
Lesson Wrap-up	T-82

Introduction

Aside from executing a couple of pre-existing DO files, the previous lessons focused on using ModelSim in interactive mode: executing single commands, one after another, via the GUI menus or Main window command line. In situations where you have repetitive tasks to complete, you can increase your productivity with DO files.

DO files are scripts that allow you to execute many commands at once. The scripts can be as simple as a series of ModelSim commands with associated arguments, or they can be full-blown Tcl programs with variables, conditional execution, and so forth. You can execute DO files from within the GUI or you can run them from the system command prompt without ever invoking the GUI.

 **Important:** This lesson assumes that you have added the `<install_dir>/modeltech/<platform>` directory to your PATH. If you did not, you will need to specify full paths to the tools (i.e., vlib, vmap, vlog, vcom, and vsim) that are used in the lesson.

Related reading

ModelSim User's Manual – [12 - Tcl and macros \(DO files\)](#) (UM-123)

Practical Programming in Tcl and Tk, Brent B. Welch, Copyright 1997

Creating a simple DO file

Creating DO files is as simple as typing the commands in a text file. Alternatively, you can save the Main window transcript as a DO file. In this exercise, you will use the transcript to create a DO file that adds signals to the Wave window, provides stimulus to those signals, and then advances the simulation.

- 1 Load the *test_counter* design unit.
 - a If necessary, start ModelSim.
 - b Change to the directory you created in [Lesson 2 - Basic simulation](#).
 - c In the Library tab of the Workspace pane, double-click the *test_counter* design unit to load it.
- 2 Enter commands to add signals to the Wave window, force signals, and run the simulation.
 - a Select **File > New > Source > Do** to create a new DO file.
 - a Enter the following commands into the source window:

```
add wave count
add wave clk
add wave reset
force -freeze clk 0 0, 1 {50 ns} -r 100
force reset 1
run 100
force reset 0
run 300
force reset 1
run 400
force reset 0
run 200
```
- 3 Save the file.
 - a Select **File > Save As**.
 - b Type **sim.do** in the File name: field and save it to the current directory.

T-76 Lesson -

- 4 Load the simulation again and use the DO file.
 - a Type **quit -sim** at the VSIM> prompt.
 - b Type **vsim test_counter** at the ModelSim> prompt.
 - c Type **do sim.do** at the VSIM> prompt.

ModelSim executes the saved commands and draws the waves in the Wave window.
- 5 When you are done with this exercise, select **File > Quit** to quit ModelSim.

Running ModelSim in command-line mode

We use the term "command-line mode" to refer to simulations that are run from a DOS/ UNIX prompt without invoking the GUI. Several ModelSim commands (e.g., vsim, vlib, vlog, etc.) are actually stand-alone executables that can be invoked at the system command prompt. Additionally, you can create a DO file that contains other ModelSim commands and specify that file when you invoke the simulator.

- 1 Create a new directory and copy the tutorial files into it.

Start by creating a new directory for this exercise. Create the directory and copy these files into it:

- `<install_dir>\modeltech\examples\counter.v`
- `<install_dir>\modeltech\examples\stim.do`

We have used the Verilog file *counter.v* in this example. If you have a VHDL license, use *counter.vhd* instead.

- 2 Create a new design library and compile the source file.

Again, enter these commands at a DOS/ UNIX prompt in the new directory you created in step 1.

- a Type **vlib work** at the DOS/ UNIX prompt.
- b For Verilog, type **vlog counter.v** at the DOS/ UNIX prompt. For VHDL, type **vcom counter.vhd**.

T-78 Lesson -

3 Create a DO file.

- a Open a text editor.
- b Type the following lines into a new file:

```
# list all signals in decimal format
add list -decimal *

# read in stimulus
do stim.do

# output results
write list counter.lst

# quit the simulation
quit -f
```

- c Save the file with the name *sim.do* and place it in the current directory.

4 Run the batch-mode simulation.

- a Type ***vsim -c -do sim.do counter -wlf counter.wlf*** at the DOS/ UNIX prompt.

The **-c** argument instructs ModelSim not to invoke the GUI. The **-wlf** argument saves the simulation results in a WLF file. This allows you to view the simulation results in the GUI for debugging purposes.

5 View the list output.

- a Open *counter.lst* and view the simulation results.

```
ns          /counter/count
delta       /counter/clk
           /counter/reset
0 +0                x z *
1 +0                0 z *
50 +0               0 * *
100 +0              0 0 *
100 +1              0 0 0
150 +0              0 * 0
151 +0              1 * 0
200 +0              1 0 0
250 +0              1 * 0
.
.
.
```

This is the output produced by the Verilog version of the design. It may appear slightly different if you used the VHDL version.

6 View the results in the GUI.

Since you saved the simulation results in *counter.wlf*, you can view them in the GUI by invoking VSIM with the **-view** argument.

- a Type **vsim -view counter.wlf** at the DOS/ UNIX prompt.

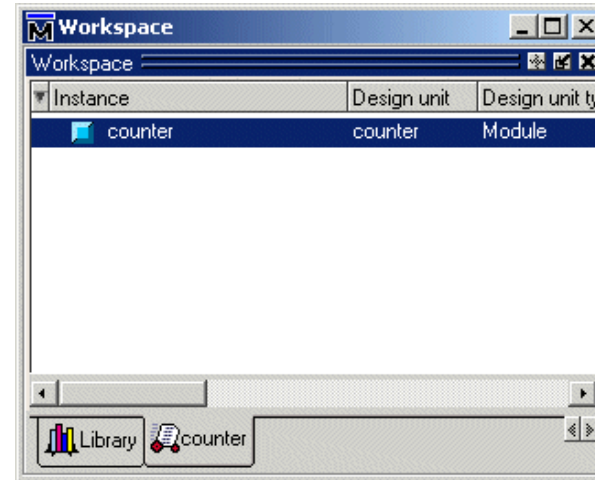
The GUI opens and a dataset tab named "counter" is displayed in the Workspace (Figure 62).

- b Right-click the *counter* instance and select **Add > Add to Wave**.

The waveforms display in the Wave window.

7 When you finish viewing the results, select **File > Quit** to close ModelSim.

Figure 62: A dataset in the Main window Workspace



Using Tcl with ModelSim

The DO files used in previous exercises contained only ModelSim commands. However, DO files are really just Tcl scripts. This means you can include a whole variety of Tcl constructs such as procedures, conditional operators, math and trig functions, regular expressions, and so forth.

In this exercise you'll create a simple Tcl script that tests for certain values on a signal and then adds bookmarks that zoom the Wave window when that value exists. Bookmarks allow you to save a particular zoom range and scroll position in the Wave window.

1 Create the script.

- a In a text editor, open a new file and enter the following lines:

```
proc add_wave_zoom {stime num} {
    echo "Bookmarking wave $num"
    bookmark add wave "bk$num" "[expr $stime - 50] [expr $stime +
100]" 0
}
```

These commands do the following:

- Create a new procedure called "add_wave_zoom" that has two arguments, *stime* and *num*.
- Create a bookmark with a zoom range from the current simulation time minus 50 time units to the current simulation time plus 100 time units.

- b Now add these lines to the bottom of the script:

```
add wave -r /*
when {clk'event and clk="1"} {
    echo "Count is [exa count]"
    if {[exa count]== "00100111"} {
        add_wave_zoom $now 1
    } elseif {[exa count]== "01000111"} {
        add_wave_zoom $now 2
    }
}
```


These commands do the following:

- Add all signals to the Wave window.
- Use a when statement to identify when *clk* transitions to 1.
- Examine the value of *count* at those transitions and add a bookmark if it is a certain value.

c Save the script with the name "*add_bkmrk.do*."

Save it into the directory you created in [Lesson 2 - Basic simulation](#).

2 Load the *test_counter* design unit.

- a Start ModelSim.
- b Select **File > Change Directory** and change to the directory you saved the DO file to in step 1c above.
- c In the Library tab of the Main window, expand the *work* library and double-click the *test_counter* design unit.

3 Execute the DO file and run the design.

- a Type **do add_bkmrk.do** at the VSIM> prompt.
- b Type **run 1500 ns** at the VSIM> prompt.
- c The simulation runs and the DO file creates two bookmarks. Select **View > Bookmarks > bm1**.

Watch the Wave window zoom on and scroll to the time when *count* is 00100111. Try the **bm2** bookmark as well.

Lesson Wrap-up

This concludes this lesson.

- 1 Select **File > Quit** to close ModelSim.

Index

A

add wave command [52](#)

B

break icon [24](#)

breakpoints

 setting [25](#)

 stepping [26](#)

C

command-line mode [77](#)

compile order, changing [33](#)

compiling your design [11](#), [21](#)

cursors, Wave window [54](#)

D

design library

 working type [14](#)

DO files [73](#)

documentation [7](#)

E

error messages, more information [44](#)

external libraries, linking to [44](#)

F

folders, in projects [35](#)

format, saving for Wave window [57](#)

L

libraries

 design library types [14](#)

 linking to external libraries [44](#)

 mapping to permanently [47](#)

 resource libraries [14](#)

 working libraries [14](#)

 working, creating [19](#)

linking to external libraries [44](#)

M

macros [73](#)

manuals [7](#)

mapping libraries permanently [47](#)

memories

 changing values [69](#)

 initializing [67](#)

 viewing [59](#)

memory contents, saving to a file [66](#)

Memory window [59](#)

N

NumericStd warnings, disabling [61](#)

O

options, simulation [37](#)

P

projects [29](#)
 adding items to [32](#)
 creating [31](#)
 flow overview [13](#)
 organizing with folders [35](#)
 simulation configurations [37](#)

Q

quit command [44](#)

R

radix command [62](#)
run -all [24](#)
run command [23](#)

S

saving simulation options [37](#)
simulation
 basic flow overview [11](#)
 restarting [25](#)
 running [23](#)
simulation configurations [37](#)
stepping after a breakpoint [26](#)

T

Tcl, using in ModelSim [80](#)
time, measuring in Wave window [54](#)

V

vcom command [61](#)
verror command [44](#)
vlib command [61](#)
vlog command [61](#)
vsim command [19](#)

W

Wave window [49](#)
 adding items to [52](#)
 cursors [54](#)
 measuring time with cursors [54](#)
 saving format [57](#)
 zooming [53](#)
working library, creating [11](#), [19](#)

Z

zooming, Wave window [53](#)