Mentor®
A Siemens Business

ModelSim® Tutorial

Software Version 2020.1

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

The ModelSim Tutorial provides lessons for gaining a basic understanding of how to simulate your design. It includes step-by-step instruction on the basics of simulation - from creating a working library, compiling your design, and loading the simulator to running the simulation and debugging your results.

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 the ModelSim GUI, the interface is consistent for all platforms.)

Example Designs

ModelSim comes with Verilog and VHDL versions of the designs used in most of 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 (for example, line numbers or syntax), you will find language-specific instructions. Follow the instructions that are appropriate for the language you use.

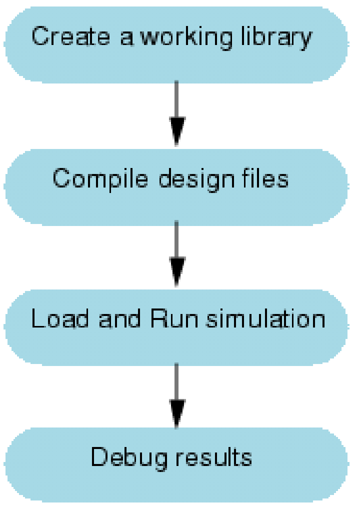
Chapter 2
Conceptual Overview

ModelSim is a verification and simulation tool for VHDL, Verilog, SystemVerilog.

Basic Simulation Flow

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

Figure 2-1. Basic Simulation Flow - Overview Lab



- Creating the Working Library

  In ModelSim, all designs are compiled into a library. You typically start a new simulation in ModelSim by creating a working library called “work,” which is the default 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.

- Loading the Simulator with Your Design and Running the Simulation

  With the design compiled, you load the simulator with your design by invoking 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

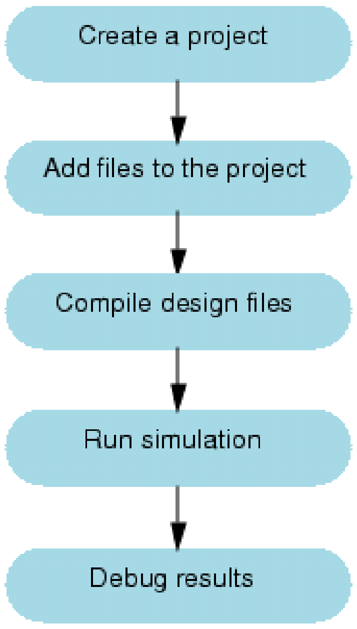
  If you do not get the results you expect, you can use the ModelSim debugging environment to track down the cause of the problem.

Project Flow

A project is a collection mechanism for an HDL design under specification or test. Even though you do not 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.

Figure 2-2. Project Flow



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.

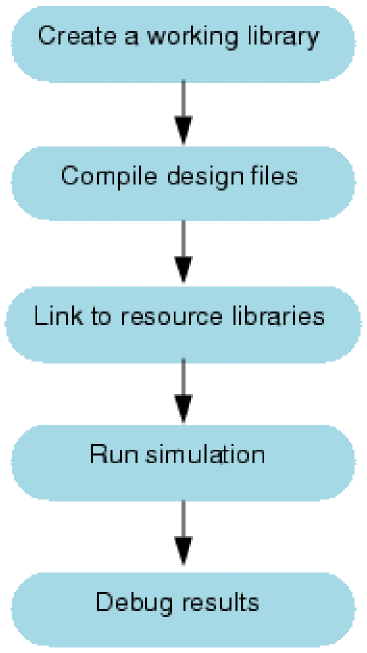
Multiple Library Flow

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 (for example, 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 test bench 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.

Figure 2-3. Multiple Library Flow



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 test bench 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:

- Using projects
- Working with multiple libraries
- Setting breakpoints and stepping through the source code
- Viewing waveforms and measuring time
- Viewing and initializing memories
- Creating stimulus with the Waveform Editor
- Automating simulation

Chapter 3
Basic Simulation

In this lesson you will guide you 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 test bench.

The pathnames are as follows:

Verilog – *<install_dir>/examples/tutorials/verilog/basicSimulation/counter.v* and t*counter.v*

VHDL – *<install_dir>/examples/tutorials/vhdl/basicSimulation/counter.vhd* and *tcounter.vhd*

This lesson uses the Verilog files *counter.v* and *tcounter.v*. 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 test bench with the VHDL counter or vice versa.

For further information, refer to:

Design Libraries, Verilog and SystemVerilog Simulation, and VHDL Simulation in the User's Manual.

The vlib, vmap, vlog, vcom, view, and run commands.

Create the Working Design Library

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

Procedure

1. Create a new directory and copy the design files for this lesson 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/tutorials/verilog/basicSimulation* to the new directory.

   VHDL: Copy *counter.vhd* and *tcounter.vhd* files from */<install_dir>/examples/tutorials/vhdl/basicSimulation* 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 box. Click **Close**.

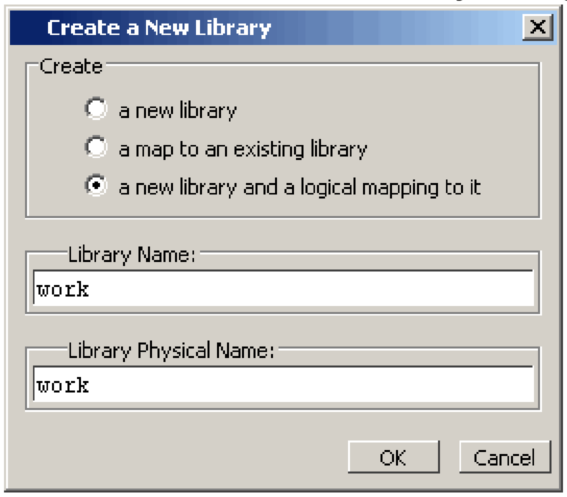
   b. Select **File > Change Directory** and change to the directory you created in step 1.

3. Create the working library.

   a. Select **File > New > Library**.

      This opens a dialog box where you specify physical and logical names for the library (Figure 3-1). You can create a new library or map to an existing library. We will be doing the former.

Figure 3-1. The Create a New Library Dialog Box


Create a New Library
Create
a new library
a map to an existing library
a new library and a logical mapping to it
OK
Cancel

Create: ( ) a new library ( ) a map to an existing library [✓ a new library and a logical mapping to it]

Library Name: work

Library Physical Name: work

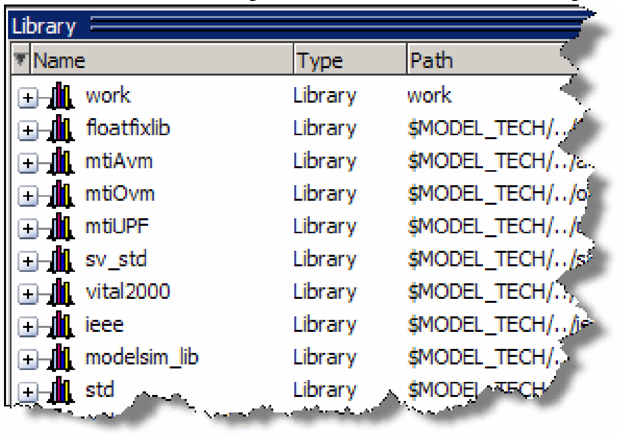
   b. Type **work** in the Library Name field (if it is not already entered automatically).

   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 Library window (Figure 3-2) and records the library mapping for future reference in the ModelSim initialization file (*modelsim.ini*).

Figure 3-2. work Library Added to the Library Window



Library

Name	Type	Path
work	Library	work
floatfixlib	Library	$MODEL_TECH/..
mtiAvm	Library	$MODEL_TECH/../a
mtiOvm	Library	$MODEL_TECH/../o
mtiUPF	Library	$MODEL_TECH/../
sv_std	Library	$MODEL_TECH/../s
vital2000	Library	$MODEL_TECH/..
ieee	Library	$MODEL_TECH/../ie
modelsim_lib	Library	$MODEL_TECH/..
std	Library	$MODEL_TECH

4. When you pressed OK in step 3c above, the following was printed to the Transcript window:

   ```
   vlib work
   vmap work work
   ```

   These two lines are the command-line equivalents of the menu selections you made. Many command-line equivalents will echo their menu-driven functions in this fashion.

Compile the Design Units

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

You can compile your source files using the menus and dialog boxes of the graphic interface, as in the Verilog example below, or by entering a command at the ModelSim> prompt.

Procedure

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

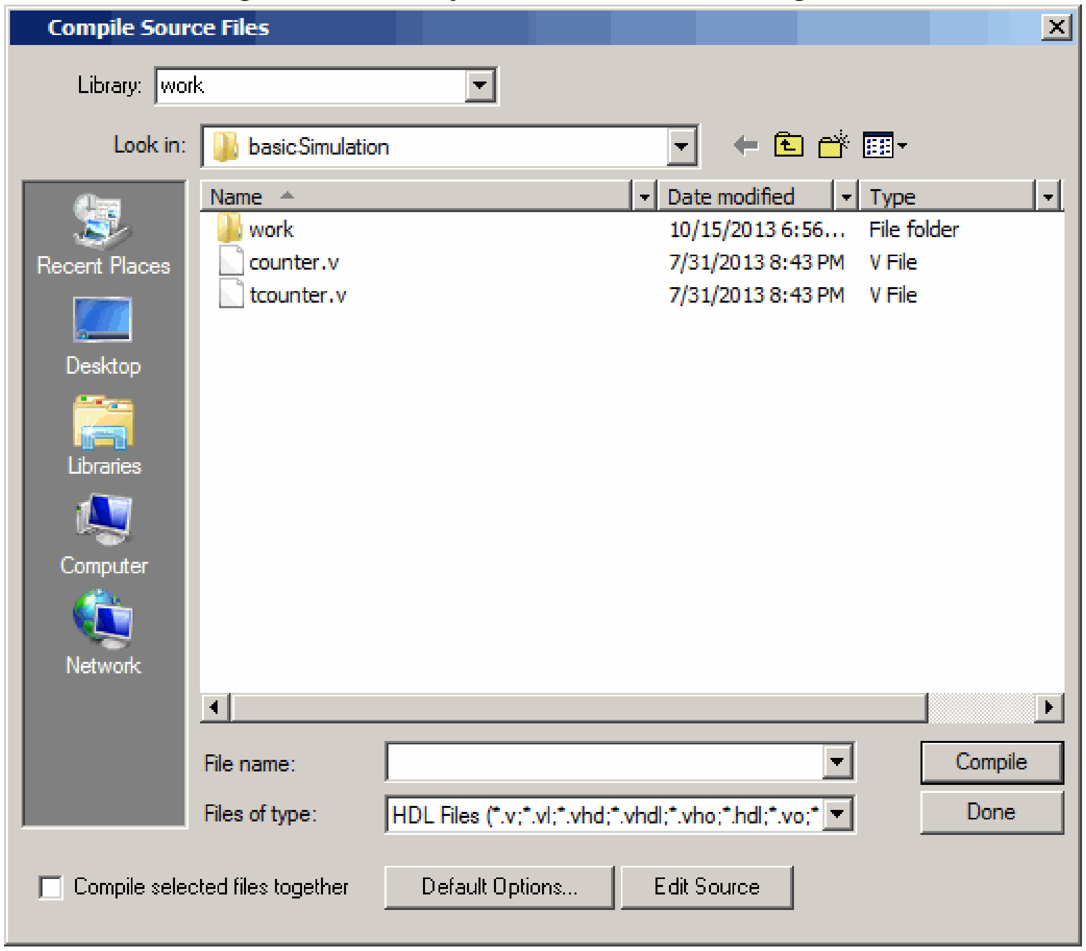
   a. Select **Compile > Compile**. This opens the Compile Source Files dialog box (Figure 3-3).

      If the Compile menu option is not available, you probably have a project open. If so, close the project by making the Library window active and selecting File > Close from the menus.

b. Select both *counter.v* and *tcounter.v* modules from the Compile Source Files dialog box and click **Compile**. The files are compiled into the *work* library.

c. When compile is finished, click **Done**.

Figure 3-3. Compile Source Files Dialog Box



Compile Source Files

Library: work

Look in: basicSimulation

Name	Date modified	Type
work	10/15/2013 6:56...	File folder
counter.v	7/31/2013 8:43 PM	V File
tcounter.v	7/31/2013 8:43 PM	V File

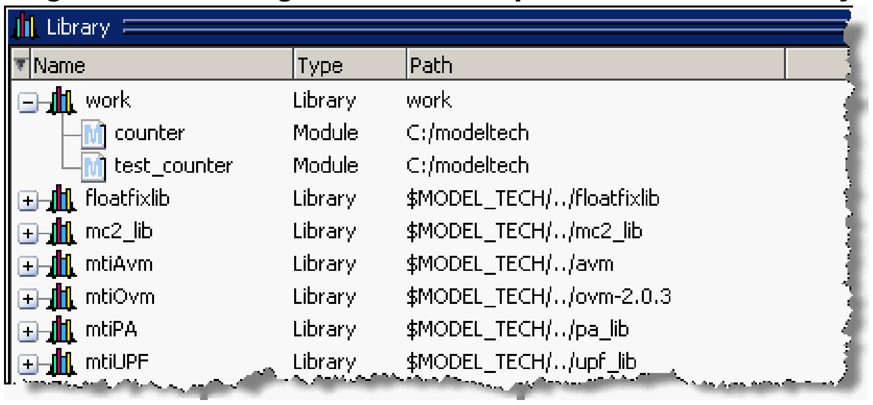
File name:

Files of type: HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*

2. View the compiled design units.

a. In the Library window, click the '+' icon next to the *work* library and you will see two design units (Figure 3-4). You can also see their types (Modules, Entities, and so on) and the path to the underlying source files.

Figure 3-4. Verilog Modules Compiled into work Library



Name	Type	Path
work	Library	work
counter	Module	C:/modeltech
test_counter	Module	C:/modeltech
floatfixlib	Library	$MODEL_TECH/../floatfixlib
mc2_lib	Library	$MODEL_TECH/../mc2_lib
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.0.3
mtiPA	Library	$MODEL_TECH/../pa_lib
mtiUPF	Library	$MODEL_TECH/../upf_lib

Load the Design

Now you are ready to load the design into the simulator.

Procedure

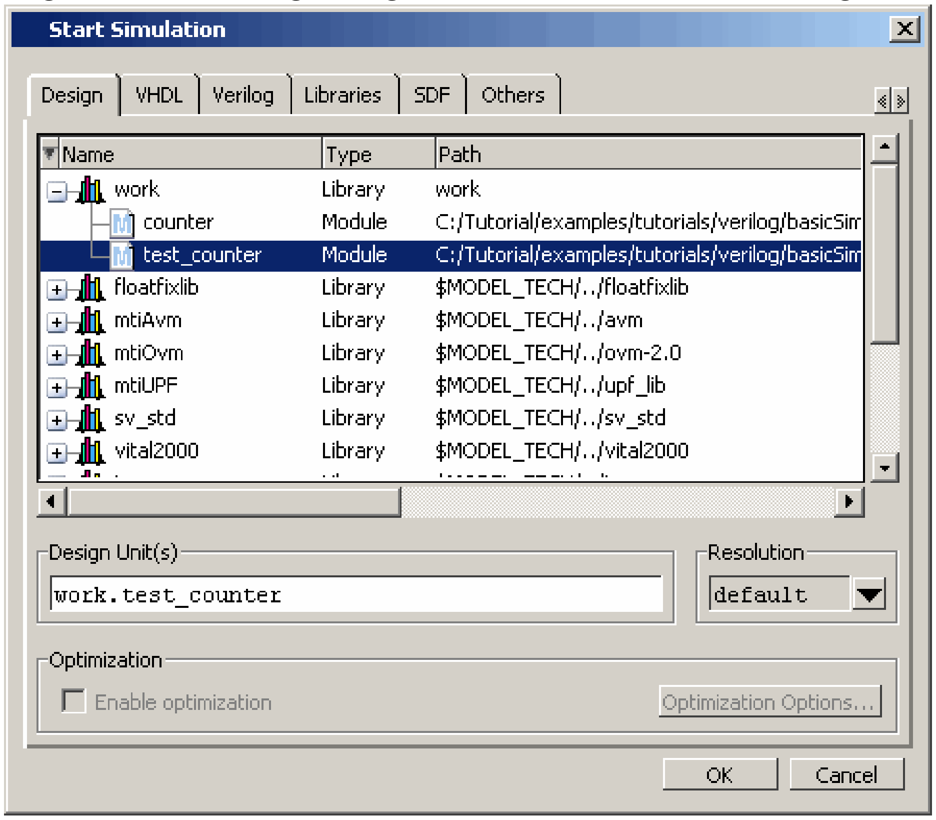
1. Load the *test_counter* module into the simulator.

   a. In the Library window, click the '+' sign next to the **work** library to show the files contained there.

   b. Double-click *test_counter* 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 box. 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 3-5).

Figure 3-5. Loading Design with Start Simulation Dialog Box

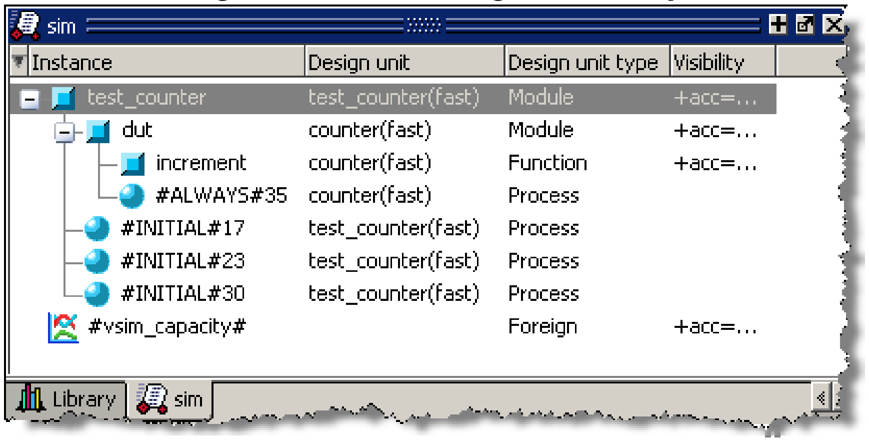


Name	Type	Path
work	Library	work
counter	Module	C:/Tutorial/examples/tutorials/verilog/basicSim
test_counter	Module	C:/Tutorial/examples/tutorials/verilog/basicSim
floatfixlib	Library	$MODEL_TECH/../floatfixlib
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.0
mtiUPF	Library	$MODEL_TECH/../upf_lib
sv_std	Library	$MODEL_TECH/../sv_std
vital2000	Library	$MODEL_TECH/../vital2000

Design Unit(s): work.test_counter
Resolution: default

When the design is loaded, a Structure window opens (labeled **sim**). This window displays the hierarchical structure of the design as shown in Figure 3-6. You can navigate within the design hierarchy in the Structure (**sim**) window by clicking on any line with a ’+’ (expand) or ’-’ (contract) icon.

Figure 3-6. The Design Hierarchy



Instance	Design unit	Design unit type	Visibility
test_counter	test_counter(fast)	Module	+acc=...
dut	counter(fast)	Module	+acc=...
increment	counter(fast)	Function	+acc=...
#ALWAYS#35	counter(fast)	Process	
#INITIAL#17	test_counter(fast)	Process	
#INITIAL#23	test_counter(fast)	Process	
#INITIAL#30	test_counter(fast)	Process	
#vsim_capacity#		Foreign	+acc=...

2. Open the Objects and Processes windows.

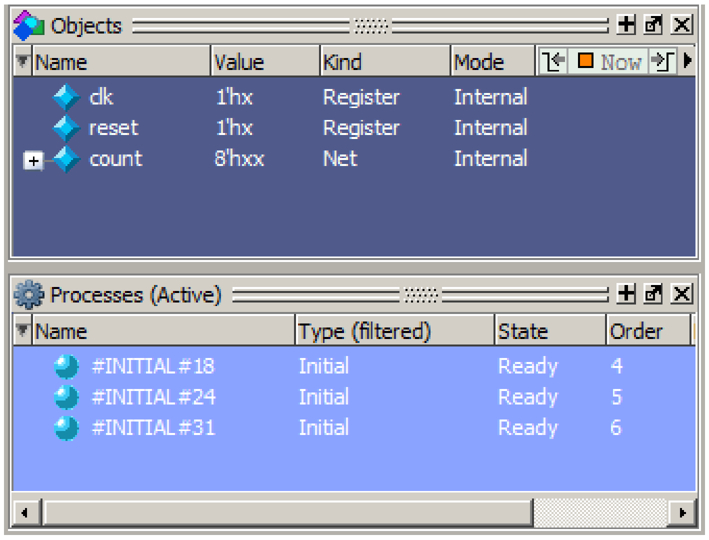
   a. Select **View** > **Objects** from the menu bar.

b. Select **View** > **Process**.

The Objects window shows the names and current values of data objects in the current region selected in the Structure (sim) window (Figure 3-7). Data objects include signals, nets, registers, constants and variables not declared in a process, generics, parameters.

The Processes window displays a list of processes in one of four viewing modes: Active, In Region, Design, and Hierarchical. The Design view mode is intended for primary navigation of ESL (Electronic System Level) designs where processes are a foremost consideration. By default, this window displays the active processes in your simulation (Active view mode).

Figure 3-7. The Object Window and Processes Window



Name	Value	Kind	Mode
clk	1'hx	Register	Internal
reset	1'hx	Register	Internal
count	8'hxx	Net	Internal

Name	Type (filtered)	State	Order
#INITIAL#18	Initial	Ready	4
#INITIAL#24	Initial	Ready	5
#INITIAL#31	Initial	Ready	6

Run the Simulation

We are ready to run the simulation. But before we do, we will open the Wave window and add signals to it.

Procedure

1. Open the Wave window.

   a. Enter view wave at the command line.

   The Wave window opens in the right side of the Main window. Resize it, if necessary, so it is visible.

   You can also use the **View** > **Wave** menu selection to open a Wave window. The Wave window is just one of several debugging windows available on the View menu.

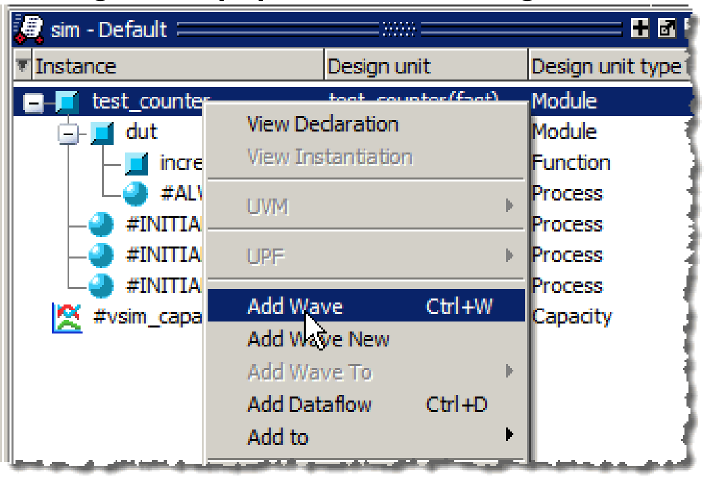
2. Add signals to the Wave window.

a. In the Structure (sim) window, right-click *test_counter* to open a popup context menu.

b. Select **Add Wave** (Figure 3-8).

All signals in the design are added to the Wave window.

Figure 3-8. Using the Popup Menu to Add Signals to Wave Window



3. Run the simulation.

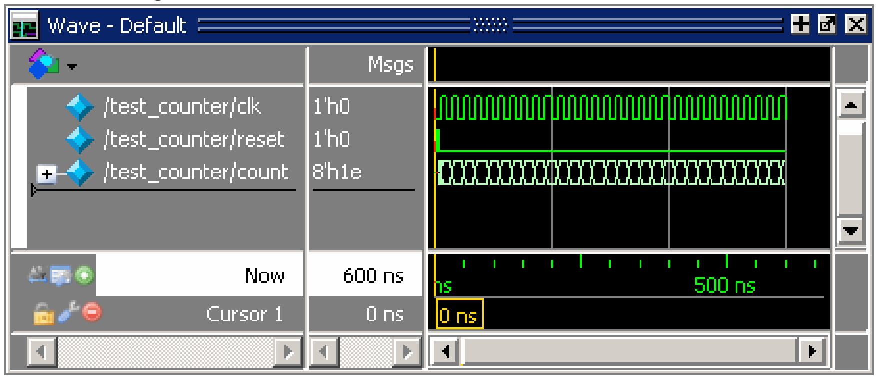
a. Click the Run icon.

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

b. Enter **run 500** at the VSIM> prompt in the Transcript window.

The simulation advances another 500 ns for a total of 600 ns (Figure 3-9).

Figure 3-9. Waves Drawn in Wave Window



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 (ie., a Verilog $stop statement) that halts the simulation.

d. Click the Break icon to stop the simulation.

Set Breakpoints and Step through the Source

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 executable lines, which are indicated with red line numbers.

Procedure

1. Open counter.v in the Source window.

   a. Select **View** > **Files** to open the Files window.

   b. Click the + sign next to the *sim* filename to see the contents of *vsim.wlf* dataset.

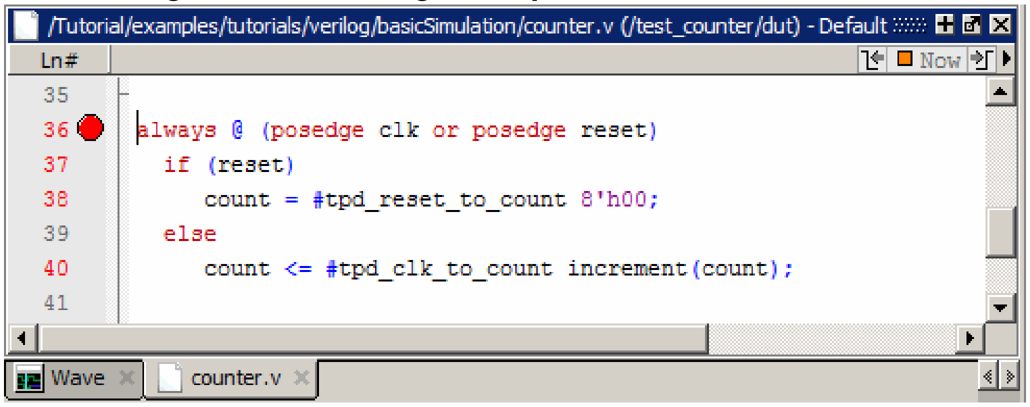
   c. Double-click counter.v (or *counter.vhd* if you are simulating the VHDL files) to open the file in the Source window.

2. Set a breakpoint on line 36 of *counter.v* (or, line 39 of counter.vhd for VHDL).

   a. Scroll to line 36 and click in the Ln# (line number) column next to the line number.

   A red dot appears in the line number column at line number 36 (Figure 3-10), indicating that a breakpoint has been set.

Figure 3-10. Setting Breakpoint in Source Window


/Tutorial/examples/tutorials/verilog/basicSimulation/counter.v (/test_counter/dut) - Default
Ln#
Now
Wave
counter.v

```
35
36 ●  always @ (posedge clk or posedge reset)
37     if (reset)
38        count = #tpd_reset_to_count 8'h00;
39     else
40        count <= #tpd_clk_to_count increment(count);
41
```

3. Disable, enable, and delete the breakpoint.

   a. Click the red dot to disable the breakpoint. It will become a gray dot.

   b. Click the gray dot again to re-enable the breakpoint. It will become a red dot.

   c. Click the red dot with your right mouse button and select **Remove Breakpoint 36**.

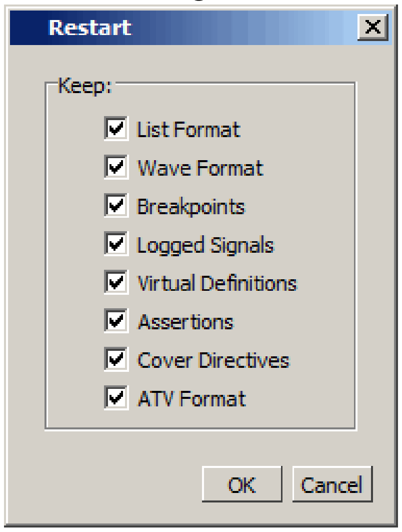
d. Click in the line number column next to line number 36 again to re-create the breakpoint.

4. Restart the simulation.

a. Click the Restart icon to reload the design elements and reset the simulation time to zero.

The Restart dialog box that appears gives you options on what to retain during the restart (Figure 3-11).

Figure 3-11. Setting Restart Functions


Restart

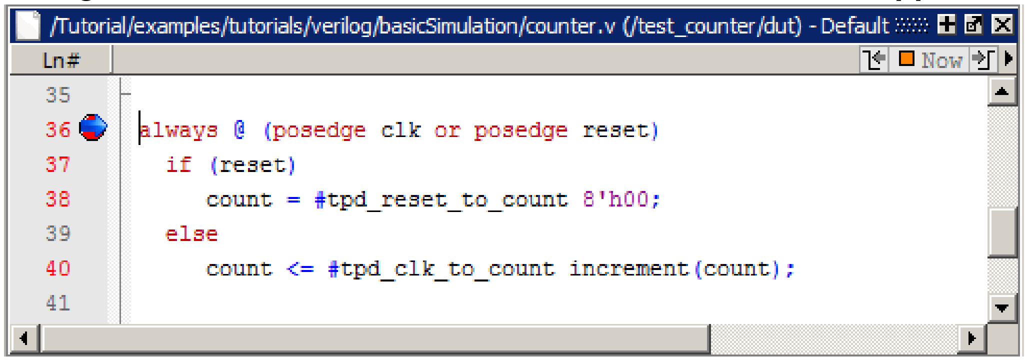
Keep:
- ✓ List Format
- ✓ Wave Format
- ✓ Breakpoints
- ✓ Logged Signals
- ✓ Virtual Definitions
- ✓ Assertions
- ✓ Cover Directives
- ✓ ATV Format

b. Click the **OK** button in the Restart dialog box.

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 3-12), and issues a Break message in the Transcript window.

Figure 3-12. Blue Arrow Indicates Where Simulation Stopped.

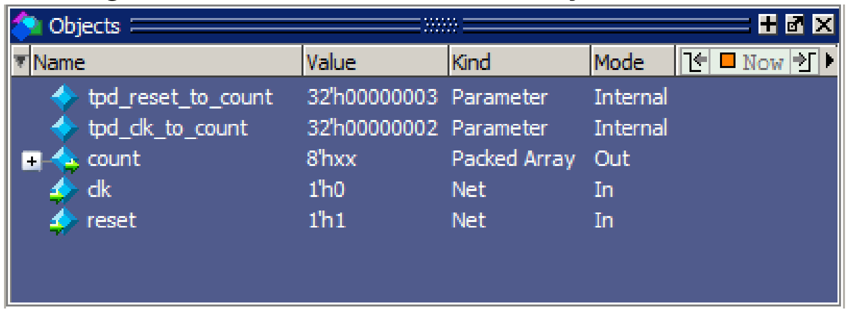

/Tutorial/examples/tutorials/verilog/basicSimulation/counter.v (/test_counter/dut) - Default

```
35
36  always @ (posedge clk or posedge reset)
37    if (reset)
38       count = #tpd_reset_to_count 8'h00;
39    else
40       count <= #tpd_clk_to_count increment(count);
41
```

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 3-13).

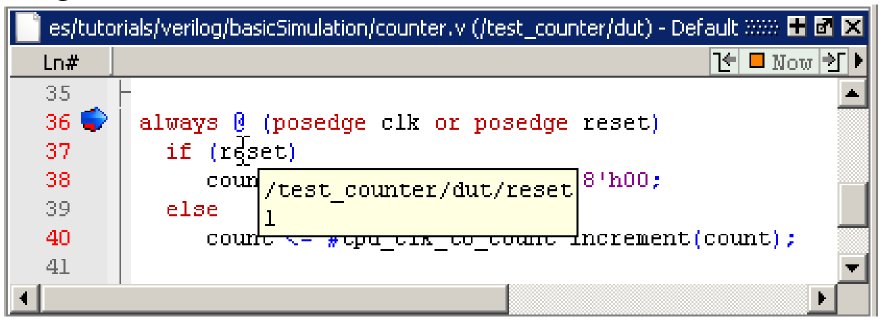
Figure 3-13. Values Shown in Objects Window



Name	Value	Kind	Mode
tpd_reset_to_count	32'h00000003	Parameter	Internal
tpd_clk_to_count	32'h00000002	Parameter	Internal
count	8'hxx	Packed Array	Out
clk	1'h0	Net	In
reset	1'h1	Net	In

- Set your mouse pointer over a variable in the Source window and a yellow box will appear with the variable name and the value of that variable at the time of the selected cursor in the Wave window (Figure 3-14).

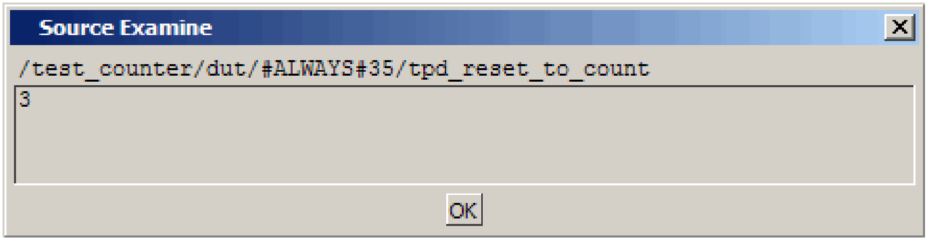
Figure 3-14. Hover Mouse Over Variable to Show Value


es/tutorials/verilog/basicSimulation/counter.v (/test_counter/dut) - Default
Ln#
Now

```
always @ (posedge clk or posedge reset)
  if (reset)
     coun /test_counter/dut/reset 8'h00;
  else 1
     count <= #tpd_clk_to_count increment(count);
```

- Highlight a signal, parameter, or variable in the Source window, right-click it, and select **Examine** from the pop-up menu to display the variable and its current value in a Source Examine window (Figure 3-15).

Figure 3-15. Parameter Name and Value in Source Examine Window



```
/test_counter/dut/#ALWAYS#35/tpd_reset_to_count
3
```

- use the **examine** command at the VSIM> prompt to output a variable value to the Transcript window (that is, examine count)

5. Try out the step commands.

   a. Click the Step Into icon on the Step 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.

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.

Chapter 4 Projects

In this lesson you will create a project.

Projects contain a work library and a session state that is stored in an *.mpf* file. Projects 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

Design Files for this Lesson

The sample design for this lesson is a simple 8-bit, binary up-counter with an associated test bench.

The pathnames are as follows:

Verilog – *<install_dir>/examples/tutorials/verilog/projects/counter.v* and t*counter.v*

VHDL – *<install_dir>/examples/tutorials/vhdl/projects/counter.vhd* and *tcounter.vhd*

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

For further information, refer to Projects in the User's Manual.

Project Work Flow

Common tasks for creating and building a new project.

Create a New Project

We will start the process of creating a new project by defining the project settings.

Procedure

1. Create a new directory and copy the design files for this lesson 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/tutorials/verilog/projects* to the new directory.

   VHDL: Copy *counter.vhd* and *tcounter.vhd* files from */<install_dir>/examples/tutorials/vhdl/projects* to the new directory.

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

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

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

3. Create a new project.

   a. Select **File > New > Project** (Main window) from the menu bar.

      This opens the Create Project dialog box where you can enter a Project Name, Project Location (that is, directory), and Default Library Name (Figure 4-1). You can also reference library settings from a selected .ini file or copy them directly into the project. The default library is where compiled design units will reside.

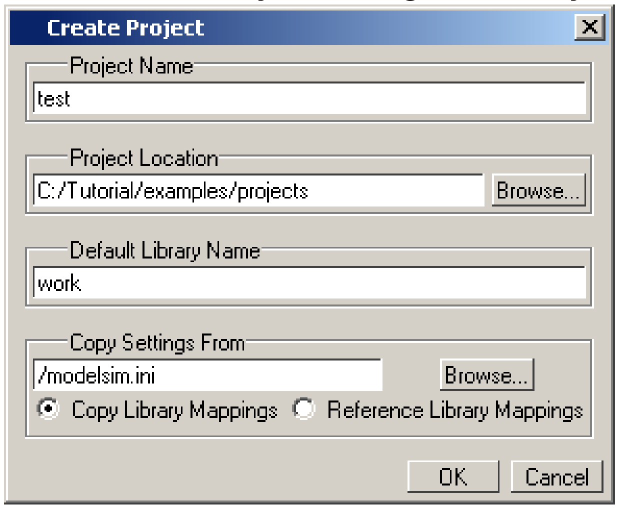
   b. Type **test** in the Project Name field.

   c. Click the **Browse** button for the Project Location field to select a directory where the project file will be stored.

   d. Leave the Default Library Name set to *work*.

e. Click **OK**.

Figure 4-1. Create Project Dialog Box - Project Lab



Project Name: test

Project Location: C:/Tutorial/examples/projects

Default Library Name: work

Copy Settings From: /modelsim.ini

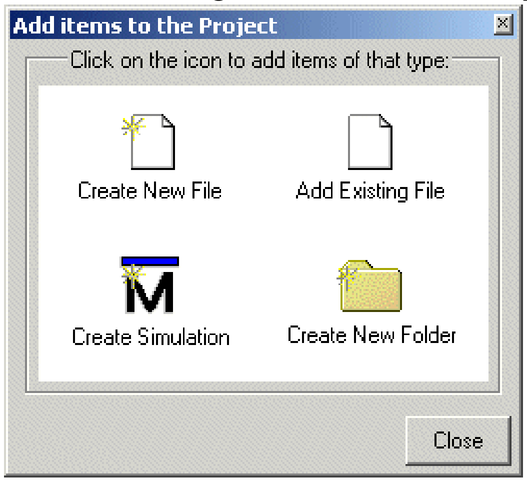
[✓ Copy Library Mappings] Reference Library Mappings

Add Objects to the Project

Once you click OK to accept the new project settings, a blank Project window and the "Add items to the Project" dialog box will appear.

From the dialog box (Figure 4-2) you can create a new design file, add an existing file, add a folder for organization purposes, or create a simulation configuration (discussed below).

Figure 4-2. Adding New Items to a Project



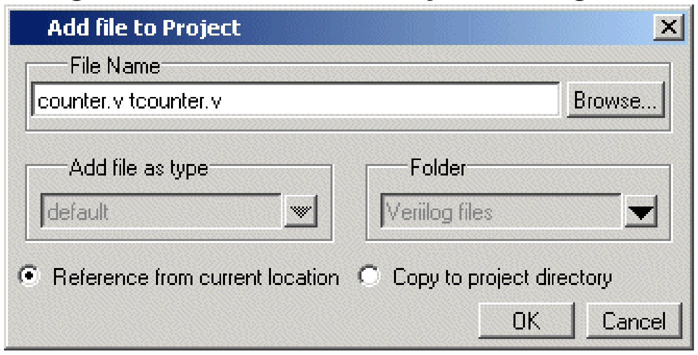
Procedure

Add two existing files.

a. Click **Add Existing File**.

This opens the Add file to Project dialog box (Figure 4-3). This dialog box lets you browse to find files, specify the file type, specify a folder to which the file will be added, and identify whether to leave the file in its current location or to copy it to the project directory.

Figure 4-3. Add file to Project Dialog Box



File Name: counter.v tcounter.v
Add file as type: default
Folder: Verilog files
[✓ Reference from current location] Copy to project directory

b. Click the **Browse** button for the File Name field. This opens the “Select files to add to project” dialog box and displays the contents of the current directory.

c. Verilog: Select *counter.v* and *tcounter.v* and click **Open**.
VHDL: Select *counter.vhd* and *tcounter.vhd* and click **Open**.

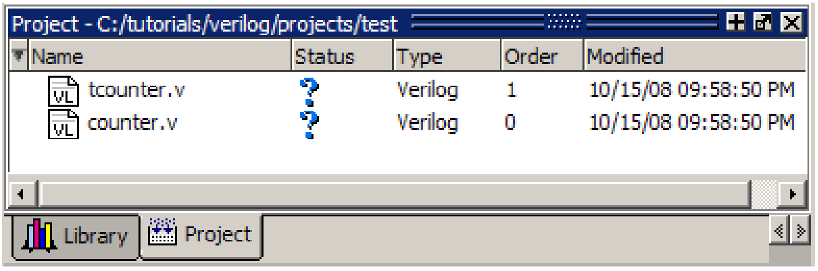
This closes the “Select files to add to project” dialog box and displays the selected files in the “Add file to Project” dialog box (Figure 4-3).

d. Click **OK** to add the files to the project.

e. Click **Close** to dismiss the Add items to the Project dialog box.

You should now see two files listed in the Project window (Figure 4-4). Question-mark icons in the Status column indicate that the file has not been compiled or that the source file has changed since the last successful compile. The other columns identify file type (for example, Verilog or VHDL), compilation order, and modified date.

Figure 4-4. Newly Added Project Files Display a '?' for Status



Name	Status	Type	Order	Modified
tcounter.v	?	Verilog	1	10/15/08 09:58:50 PM
counter.v	?	Verilog	0	10/15/08 09:58:50 PM

Changing Compile Order (VHDL)

By default ModelSim performs default binding of VHDL designs when you load the design with the **vsim** command. However, you can elect to perform default binding at compile time. If you elect to do default binding at compile, then the compile order is important. Follow these steps to change compilation order within a project.

Procedure

Change the compile order.

a. Select **Compile > Compile Order**.

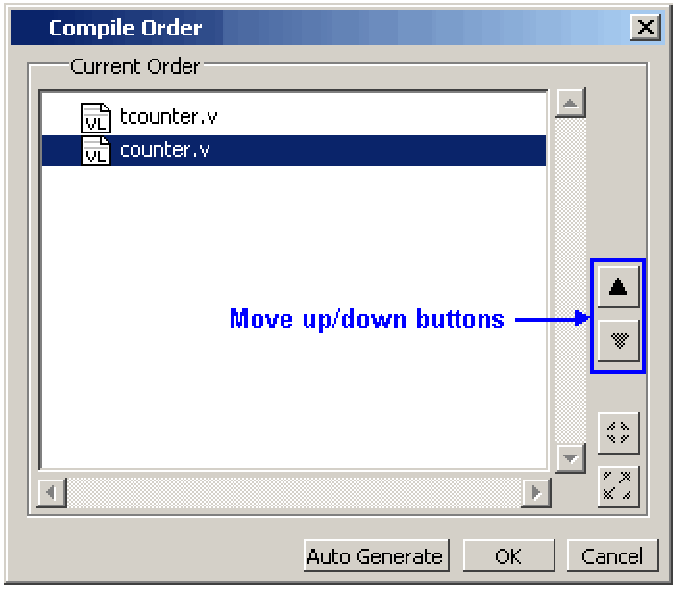
   This opens the Compile Order dialog box.

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) cannot 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 (Figure 4-5).

Figure 4-5. Compile Order Dialog Box



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

Related Topics

Default Binding [ModelSim User's Manual]

Compile the Design

With the Project settings defined and objects added to the project, you are ready to compile the design.

Procedure

1. Compile the files.

   a. Right-click either *counter.v* or *tcounter.v* in the Project window and select **Compile** > **Compile All** from the pop-up menu.

      ModelSim compiles both files and changes the symbol in the Status column to a green check mark. A check mark means the compile succeeded. If compile fails, the symbol will be a red ’X’, and you will see an error message in the Transcript window.

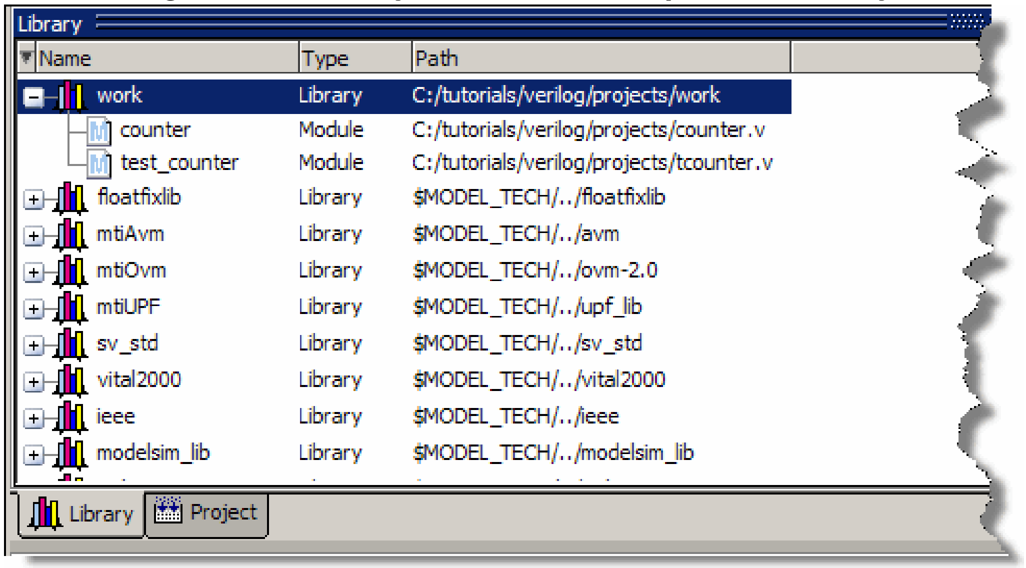
2. View the design units.

   a. Click the **Library** tab (Figure 4-6).

   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 4-6. Library Window with Expanded Library



Name	Type	Path
work	Library	C:/tutorials/verilog/projects/work
counter	Module	C:/tutorials/verilog/projects/counter.v
test_counter	Module	C:/tutorials/verilog/projects/tcounter.v
floatfixlib	Library	$MODEL_TECH/../floatfixlib
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.0
mtiUPF	Library	$MODEL_TECH/../upf_lib
sv_std	Library	$MODEL_TECH/../sv_std
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib

Load the Design

Now we are ready to load the design into the simulator.

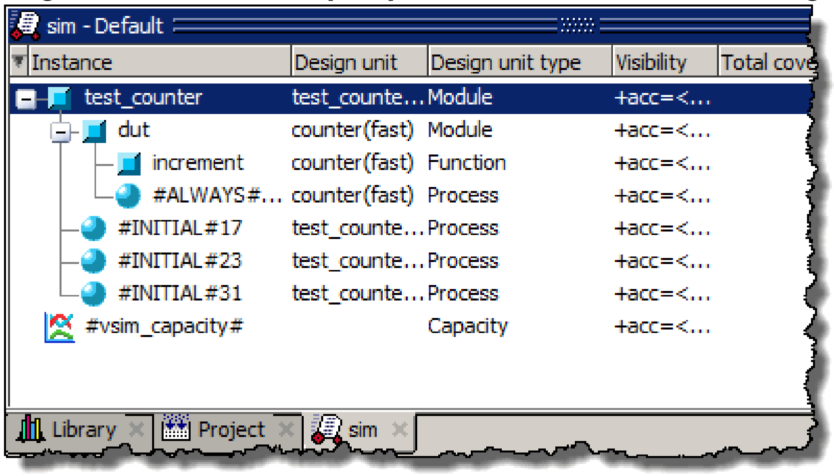
Procedure

1. Load the *test_counter* design unit.

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

   The Structure (sim) window appears as part of the tab group with the Library and Project windows (Figure 4-7).

   Figure 4-7. Structure(sim) window for a Loaded Design

Instance	Design unit	Design unit type	Visibility	Total cove
test_counter	test_counte…	Module	+acc=<…	
dut	counter(fast)	Module	+acc=<…	
increment	counter(fast)	Function	+acc=<…	
#ALWAYS#…	counter(fast)	Process	+acc=<…	
#INITIAL#17	test_counte…	Process	+acc=<…	
#INITIAL#23	test_counte…	Process	+acc=<…	
#INITIAL#31	test_counte…	Process	+acc=<…	
#vsim_capacity#		Capacity	+acc=<…	

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

2. End the simulation.

   a. Select **Simulate > End Simulation**.

   b. Click **Yes**.

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 4-3). If you create a folder after adding files, you edit the file properties to move it to that folder.

Adding Folders

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

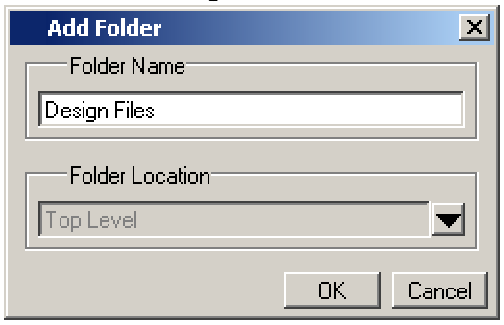
Procedure

1. Add a new folder.

   a. Right-click in the Projects window and select **Add to Project** > **Folder**.

   b. Type **Design Files** in the **Folder Name** field (Figure 4-8).

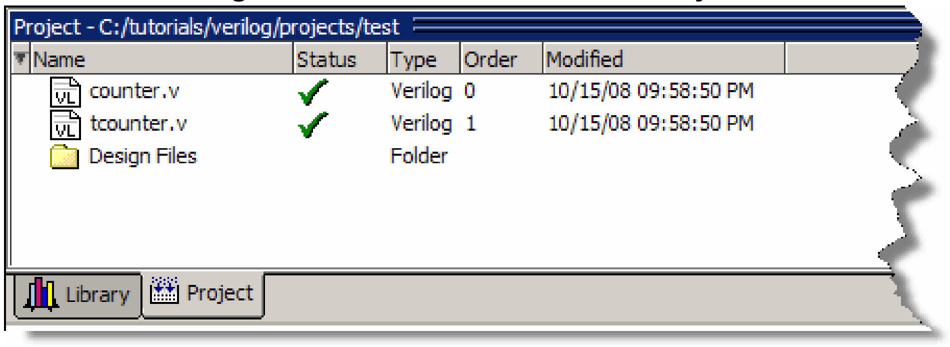
Figure 4-8. Adding New Folder to Project


Add Folder
Folder Name: Design Files
Folder Location: Top Level

   c. Click **OK**.

   The new Design Files folder is displayed in the Project window (Figure 4-9).

Figure 4-9. A Folder Within a Project


Project - C:/tutorials/verilog/projects/test

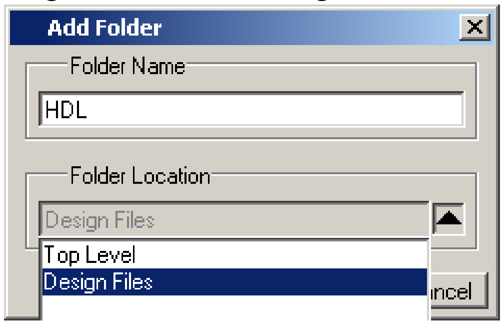
Name	Status	Type	Order	Modified
counter.v	✓	Verilog	0	10/15/08 09:58:50 PM
tcounter.v	✓	Verilog	1	10/15/08 09:58:50 PM
Design Files		Folder		

2. Add a sub-folder.

   a. Right-click anywhere in the Project window and select **Add to Project > Folder**.

   b. Type **HDL** in the **Folder Name** field (Figure 4-10).

Figure 4-10. Creating Subfolder


Add Folder

Folder Name: HDL

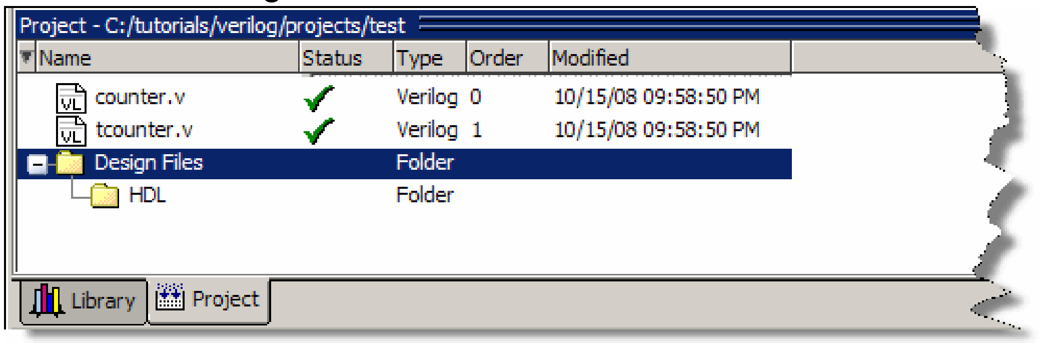
Folder Location: Design Files

   c. Click the **Folder Location** drop-down arrow and select *Design Files*.

   d. Click **OK**.

   A ’+’ icon appears next to the *Design Files* folder in the Project window (Figure 4-11).

Figure 4-11. A folder with a Sub-folder


Project - C:/tutorials/verilog/projects/test

Name	Status	Type	Order	Modified
counter.v	✓	Verilog	0	10/15/08 09:58:50 PM
tcounter.v	✓	Verilog	1	10/15/08 09:58:50 PM
Design Files		Folder		
HDL		Folder		

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

Moving Files to Folders

If you do not place files into a folder when you first add the files to the project, you can move them into a folder using the Project Compiler Settings dialog box.

Procedure

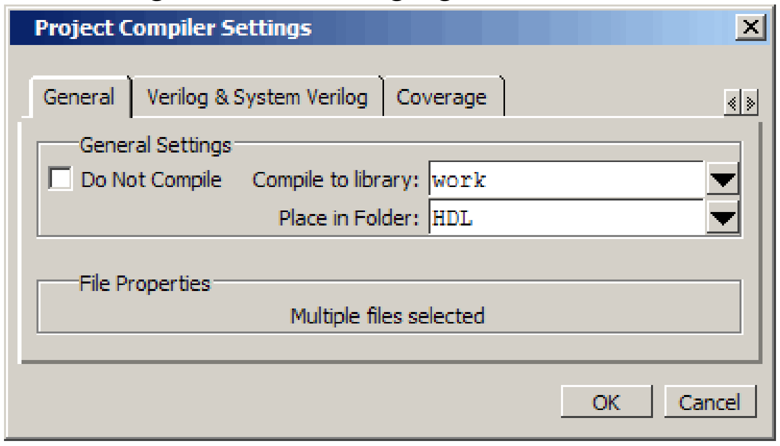
Move *tcounter.v* and *counter.v* to the *HDL* folder.

a. Select both *counter.v* and *tcounter.v* in the Project window.

b. Right-click either file and select **Properties**.

This opens the Project Compiler Settings dialog box (Figure 4-12), which allows you to set a variety of options on your design files.

Figure 4-12. Changing File Location



Project Compiler Settings

General | Verilog & System Verilog | Coverage

General Settings

Do Not Compile: ☐

Compile to library: work

Place in Folder: HDL

File Properties

Multiple files selected

OK | Cancel

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

d. Click **OK**.

The selected files are moved into the HDL folder. Click the '+' icon next to the HDL folder to see the files.

The files are now marked with a '?' in the Status column because you moved the files. The project no longer knows if the previous compilation is still valid.

Using Simulation Configurations

A Simulation Configuration associates a design unit(s) and its simulation options. For example, let us say that 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 window and you can double-click it to load *tcounter.v* along with its options.

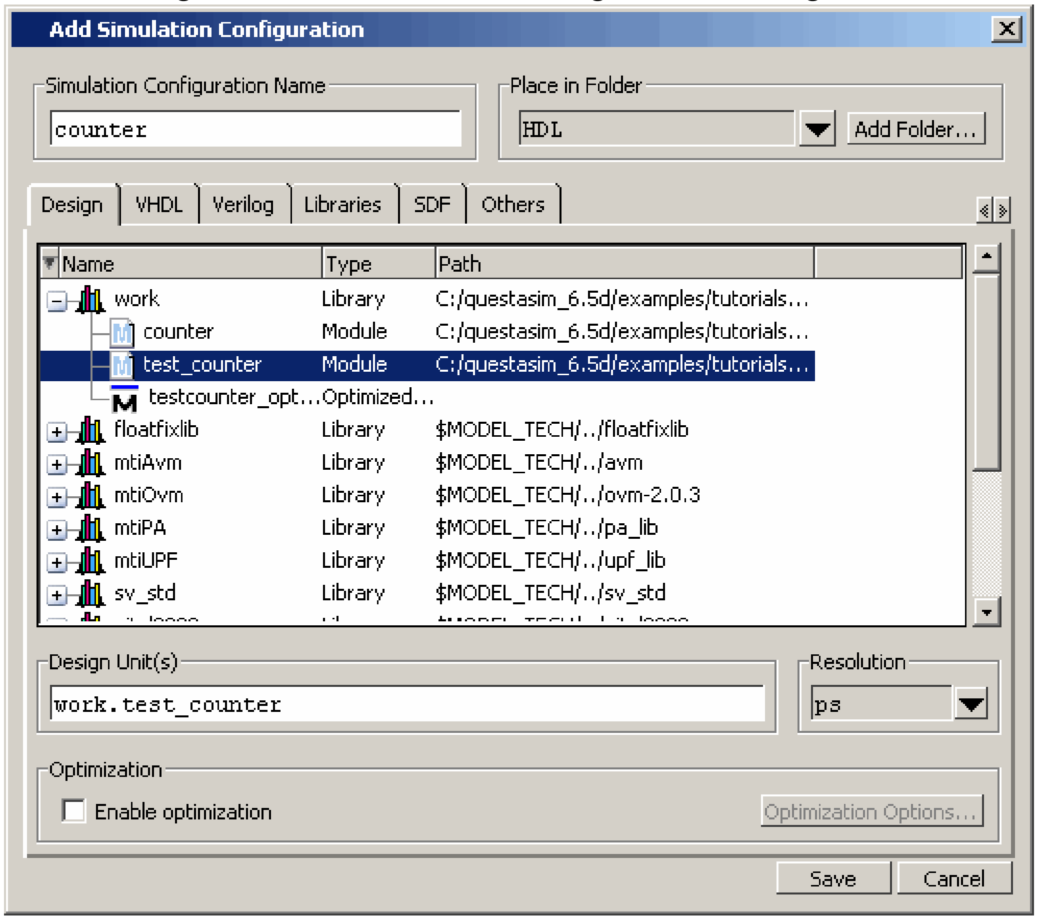
Procedure

1. Create a new Simulation Configuration.

   a. Right-click in the Project window and select **Add to Project > Simulation Configuration** from the popup menu.

   This opens the Add Simulation Configuration dialog box (Figure 4-13). The tabs in this dialog box present several simulation options. You may want to explore the tabs to see what is available. You can consult the ModelSim User's Manual to get a description of each option.

Figure 4-13. Simulation Configuration Dialog Box



Simulation Configuration Name: counter
Place in Folder: HDL

Name	Type	Path
work	Library	C:/questasim_6.5d/examples/tutorials...
counter	Module	C:/questasim_6.5d/examples/tutorials...
test_counter	Module	C:/questasim_6.5d/examples/tutorials...
testcounter_opt...	Optimized...	
floatfixlib	Library	$MODEL_TECH/../floatfixlib
mtiAvm	Library	$MODEL_TECH/../avm
mtiOvm	Library	$MODEL_TECH/../ovm-2.0.3
mtiPA	Library	$MODEL_TECH/../pa_lib
mtiUPF	Library	$MODEL_TECH/../upf_lib
sv_std	Library	$MODEL_TECH/../sv_std

Design Unit(s): work.test_counter
Resolution: ps
Optimization: Enable optimization

   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 (-hazards)**.

g. Click **Save**.

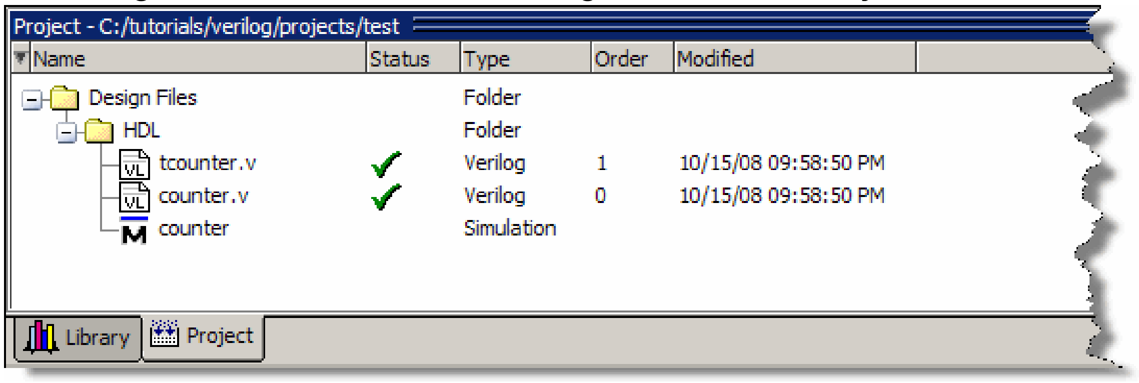
The files *tcounter.v* and *counter.v* show question mark icons in the status column because they have changed location since they were last compiled and need to be recompiled.

h. Select one of the files, *tcounter.v* or *counter.v*.

i. Select **Compile > Compile All**.

The Project window now shows a Simulation Configuration named *counter* in the HDL folder (Figure 4-14).

Figure 4-14. A Simulation Configuration in the Project window



Project - C:/tutorials/verilog/projects/test

Name	Status	Type	Order	Modified
Design Files		Folder		
HDL		Folder		
tcounter.v	✓	Verilog	1	10/15/08 09:58:50 PM
counter.v	✓	Verilog	0	10/15/08 09:58:50 PM
counter		Simulation		

2. Load the Simulation Configuration.

a. Double-click the *counter* Simulation Configuration in the Project window.

In the Transcript window of the Main window, the **vsim** (the ModelSim simulator) invocation shows the **-hazards** and **-t ps** switches. These are the command-line equivalents of the options you specified in the Simulate dialog box.

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. In the Project window, right-click and select **Close Project**.

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

Chapter 5
Working With Multiple Libraries

In this lesson you will practice working with multiple libraries. 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 test bench 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 test bench.

The pathnames are as follows:

- Verilog — *<install_dir>/examples/tutorials/verilog/libraries/counter.v* and *tcounter.v*
- VHDL — *<install_dir>/examples/tutorials/vhdl/libraries/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.

For further information, refer to Design Libraries in the User's Manual.

Creating the Resource Library

Before creating the resource library, make sure the *modelsim.ini* in your install directory is "Read Only." This will prevent permanent mapping of resource libraries to the master *modelsim.ini* file.

For additional information, see Permanently Mapping VHDL Resource Libraries.

Procedure

1. Create a directory for the resource library.

   Create a new directory called resource_library. Copy *counter.v* from *<install_dir>/ examples/tutorials/verilog/libraries* to the new directory.

2. Create a directory for the test bench.

   Create a new directory called testbench that will hold the test bench and project files. Copy *tcounter.v* from *<install_dir>/examples/tutorials/verilog/libraries* 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 resource_library 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 box appears, click **Close**.

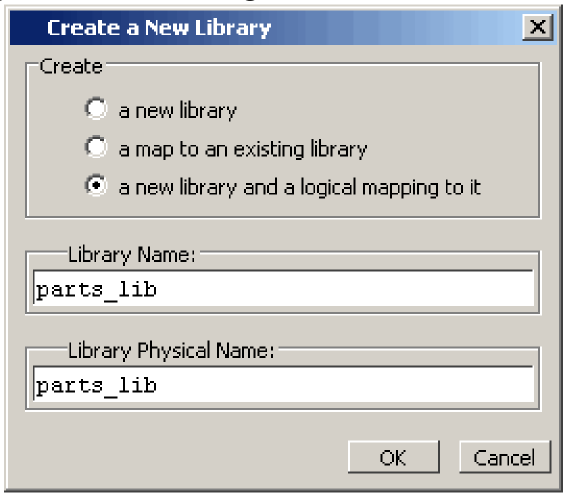
   b. Select **File > Change Directory** and change to the resource_library directory you created in step 1.

4. Create the resource library.

   a. Select **File > New > Library**.

   b. Type **parts_lib** in the Library Name field (Figure 5-1).

Figure 5-1. Creating New Resource Library



( ) a new library
( ) a map to an existing library
(●) a new library and a logical mapping to it

Library Name: parts_lib

Library Physical Name: parts_lib

The Library Physical Name field is filled out automatically.

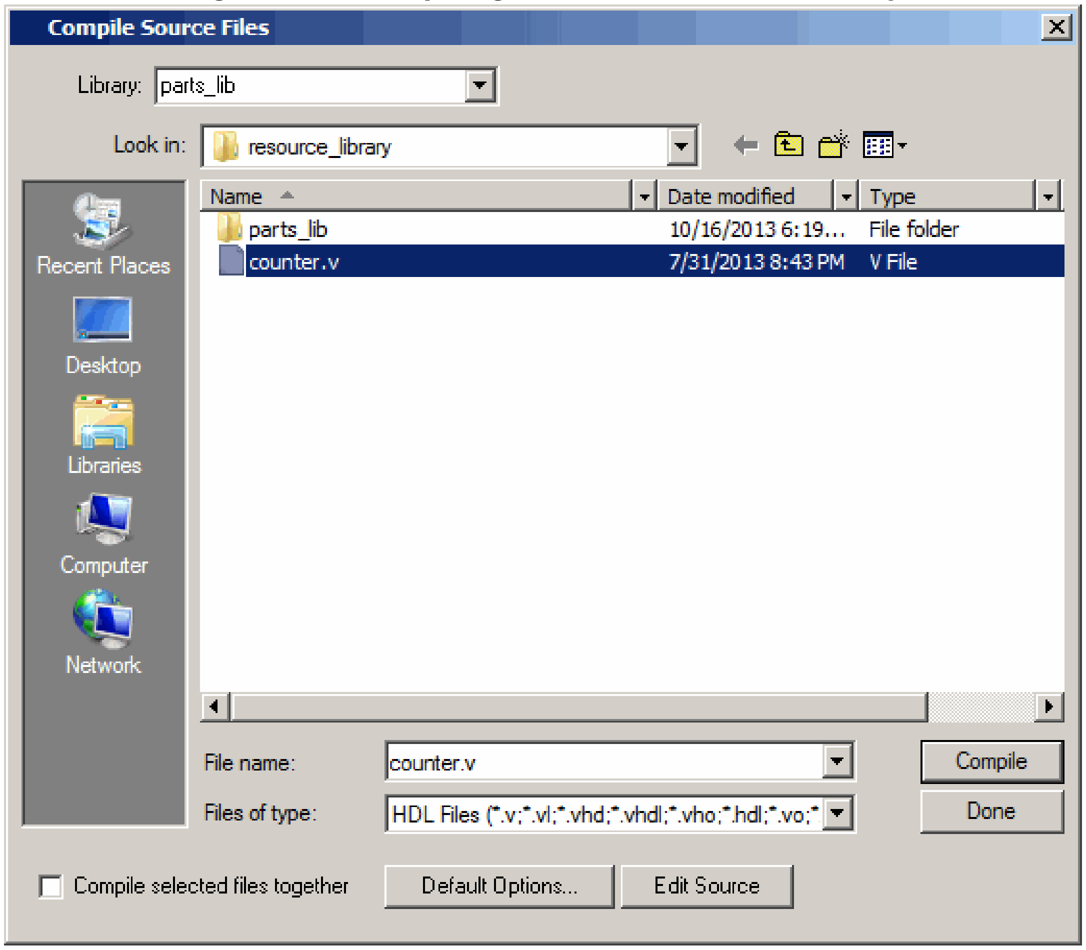
Once you click OK, ModelSim creates a directory for the library, lists it in the Library window, and modifies the *modelsim.ini* file to record this new library for the future.

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 5-2).

Figure 5-2. Compiling into the Resource Library



Library: parts_lib

Look in: resource_library

Name	Date modified	Type
parts_lib	10/16/2013 6:19...	File folder
counter.v	7/31/2013 8:43 PM	V File

File name: counter.v

Files of type: HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*

☐ Compile selected files together

   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.

Creating the Project

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

Procedure

1. Create the project.

    a. Select **File** > **New** > **Project**.

    b. Type **counter** in the Project Name field.

    c. Do not change the Project Location field or the Default Library Name field. (The default library name is *work*.)

    d. Make sure "Copy Library Mappings" is selected. The default *modelsim.ini* file will be used.

    e. Click **OK**.

2. Add the test bench to the project.

    a. Click **Add Existing File** in the Add items to the Project dialog box.

    b. Click the **Browse** button and select *tcounter.v* in the "Select files to add to project" dialog box.

    c. Click **Open**.

    d. Click **OK**.

    e. Click **Close** to dismiss the "Add items to the Project" dialog box.

    The *tcounter.v* file is listed in the Project window.

3. Compile the test bench.

    a. Right-click *tcounter.v* and select **Compile** > **Compile Selected**.

Loading Without Linking Libraries

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

ModelSim responds differently for Verilog and VHDL in this situation.

Verilog

The following procedure is for those working with Verilog designs.

Load the Verilog Test Bench

Now we will load the Verilog test bench into the simulator.

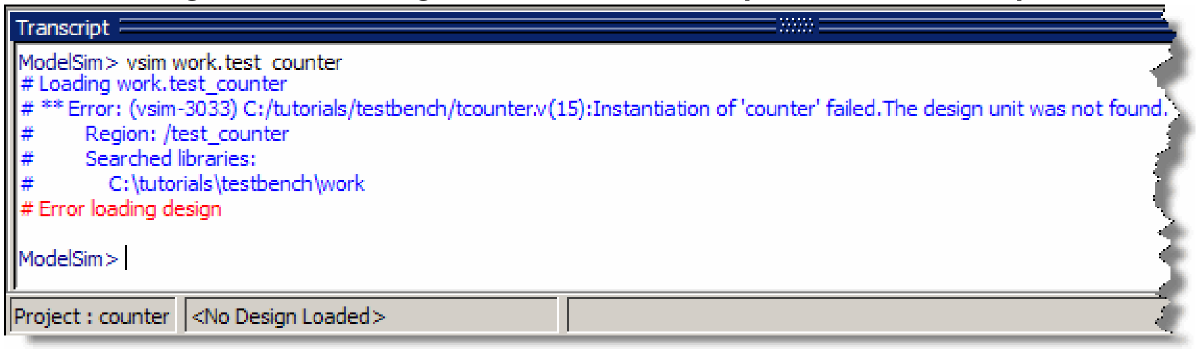
Procedure

Load a Verilog design with a missing resource library.

a. In the Library window, click the ’+’ icon next to the *work* library and double-click *test_counter*.

   The Transcript reports an error (Figure 5-3). When you see a message that contains text like "Error: (vsim-3033)", you can view more detail by using the **verror** command.

Figure 5-3. Verilog Simulation Error Reported in Transcript



```
ModelSim> vsim work.test_counter
# Loading work.test_counter
# ** Error: (vsim-3033) C:/tutorials/testbench/tcounter.v(15):Instantiation of 'counter' failed.The design unit was not found.
#       Region: /test_counter
#       Searched libraries:
#           C:\tutorials\testbench\work
# Error loading design

ModelSim>
```

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*.

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

VHDL

The following procedure is for those working with VHDL designs.

Load the VHDL Test Bench

Now we will load the VHDL test bench into the simulator.

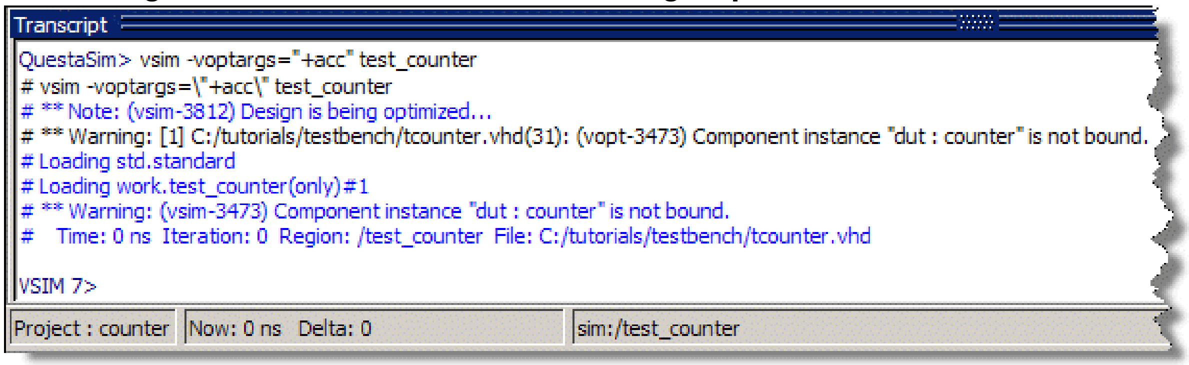
Procedure

1. Load the VHDL test bench with a missing resource library.

   a. In the Library window, click the '+' icon next to the *work* library and double-click *test_counter*.

      The Main window Transcript reports a warning (Figure 5-4). When you see a message that contains text like "Warning: (vsim-3473)", you can view more detail by using the **verror** command.

Figure 5-4. VHDL Simulation Warning Reported in Main Window



```
Transcript
QuestaSim> vsim -voptargs="+acc" test_counter
# vsim -voptargs=\"+acc\" test_counter
# ** Note: (vsim-3812) Design is being optimized...
# ** Warning: [1] C:/tutorials/testbench/tcounter.vhd(31): (vopt-3473) Component instance "dut : counter" is not bound.
# Loading std.standard
# Loading work.test_counter(only)#1
# ** Warning: (vsim-3473) Component instance "dut : counter" is not bound.
#    Time: 0 ns  Iteration: 0  Region: /test_counter  File: C:/tutorials/testbench/tcounter.vhd

VSIM 7>
Project : counter   Now: 0 ns   Delta: 0   sim:/test_counter
```

   b. Type **verror 3473** at the VSIM> 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.

2. The process for linking to a resource library differs between Verilog and VHDL. If you are using Verilog, follow the steps in Linking to the Resource Library. If you are using VHDL, follow the steps in Permanently Mapping VHDL Resource Libraries one page later.

Linking to the Resource Library

Linking to a resource library requires that you specify a "search library" when you invoke the simulator.

Procedure

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 resource_library directory you created earlier in the lesson.

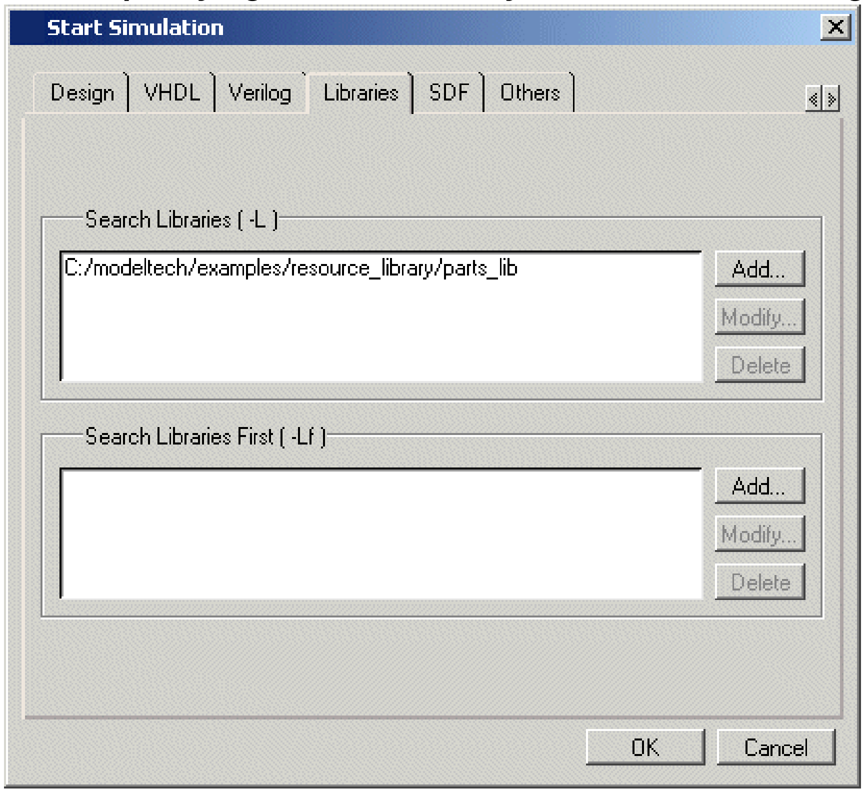
e. Click OK.

The dialog box should have *parts_lib* listed in the Search Libraries field (Figure 5-5).

f. Click OK.

The design loads without errors.

Figure 5-5. Specifying a Search Library in the Simulate Dialog Box



Search Libraries (-L)

C:/modeltech/examples/resource_library/parts_lib

Search Libraries First (-Lf)

Permanently Mapping VHDL Resource Libraries

If you reference particular VHDL resource libraries in every VHDL 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 will not actually practice it in this tutorial, here are the steps for editing the file:

Procedure

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.

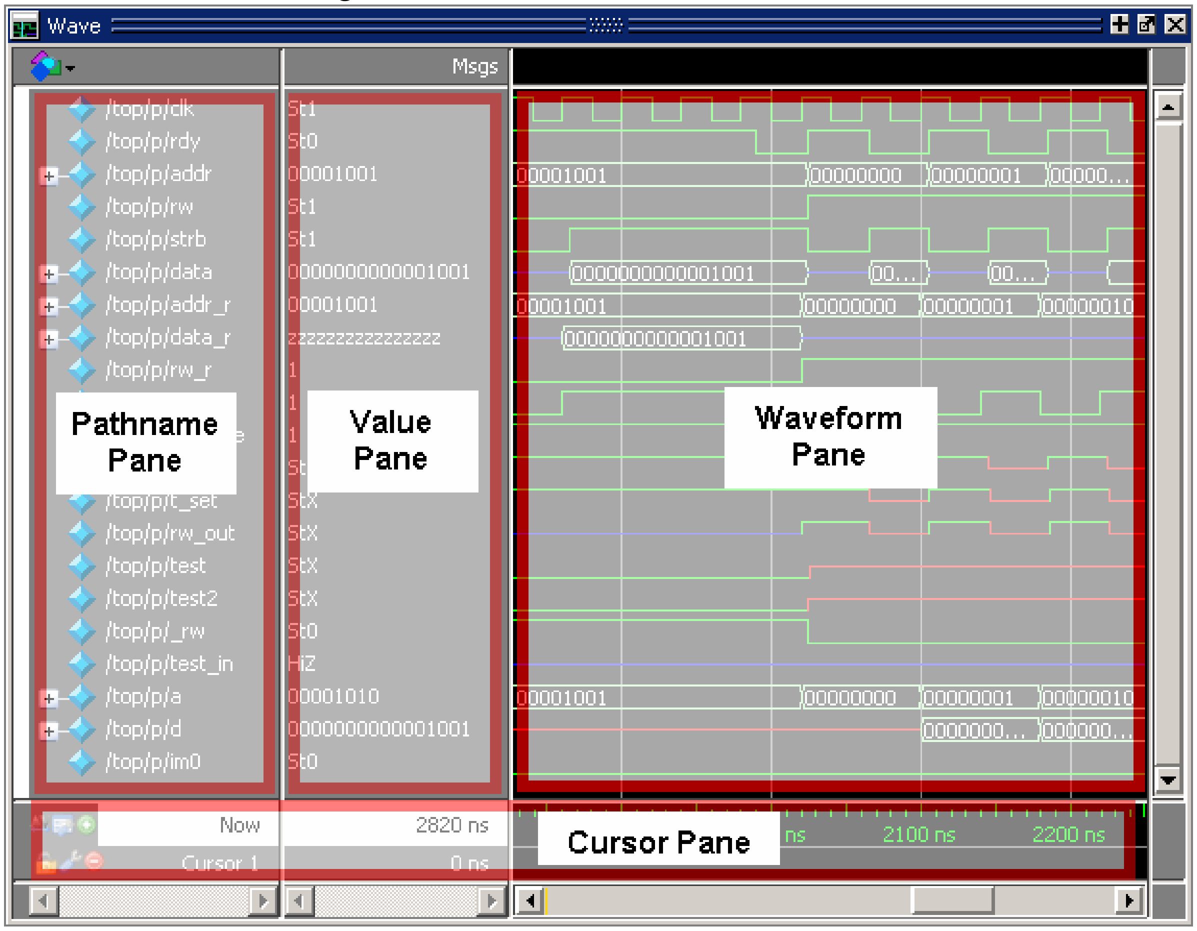
1. Select **Simulate > End Simulation**.
2. Click **Yes**.
3. Select the Project window to make it active.
4. Select **File > Close Project**. Click **OK**.

Chapter 6
Analyzing Waveforms

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 panes (Figure 6-1). You can resize the pathnames pane, the values pane, and the waveform pane by clicking and dragging the bar between any two panes.

Figure 6-1. Panes of the Wave Window



Loading a Design

For the examples in this exercise, we will use the design simulated in the Basic Simulation lesson.

Procedure

1. If you just finished a different 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 box appears, click **Close**.

2. Load the design.

   a. Select **File** > **Change Directory** and open the directory you created in the “Basic Simulation” lesson.

      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 opens a Structure (sim) window.

Add Objects to the Wave Window

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

Procedure

1. Add objects from the Objects window.

   a. Open an Objects window by selecting **View** > **Objects**.

   b. Select an item in the Objects window, right-click, and then select **Add to** > **Wave** > **Signals in Region**. ModelSim opens a Wave window and displays signals in the region.

      - Or, place the cursor over an object and click the right mouse button to open a context menu. Then click **Add Wave** to place the object in the Wave window.

      - Or, select a group of objects then click the right mouse button to open the context menu and click **Add Wave**.

2. Undock the Wave window.

   By default ModelSim opens the Wave window in the right side of the Main window. You can change the default via the Preferences dialog box (**Tools** > **Edit Preferences**). Refer to Setting GUI Preferences in the *GUI Reference Manual* for more information.

   a. Click the **undock** icon on the Wave window.

      The Wave window becomes a standalone, undocked window. Resize the window as needed.

3. Add objects using drag-and-drop.

   You can drag an object to the Wave window from many other windows (for example, Structure, Objects, and Locals).

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

   b. Drag an instance from the Structure (sim) window to the Wave window.

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

   c. Drag a signal from the Objects window to the Wave window.

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

4. Add objects using the add wave command.

   a. Type the following at the VSIM> prompt.

      ```
      add wave *
      ```

      ModelSim adds all objects from the current selected region.

   b. Run the simulation for 500 ns so you can see waveforms.

Zooming the Waveform Display

There are numerous methods for zooming the Waveform display.

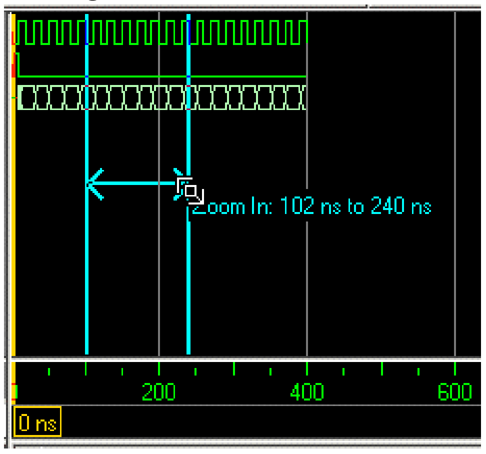
Procedure

1. Click the **Zoom Mode** icon on the Wave window toolbar.

   a. In the waveform display, click and drag down and to the right.

      You should see blue vertical lines and numbers defining an area to zoom in (Figure 6-2).

Figure 6-2. Zooming in with the Zoom Mode Mouse Pointer



2. Select **View** > **Zoom** > **Zoom Last**.

   a. The waveform display restores the previous display range.

3. Click the **Zoom In** icon a few times. 🔍

4. In the waveform display, click and drag up and to the right.

   You should see a blue line and numbers defining an area to zoom out.

5. Select **View** > **Zoom** > **Zoom Full**.

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 in the cursor timeline brings the 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.

First, dock the Wave window in the Main window by clicking the dock icon.

Working with a Single Cursor

Let's look at the information provided when using a single cursor.

Procedure

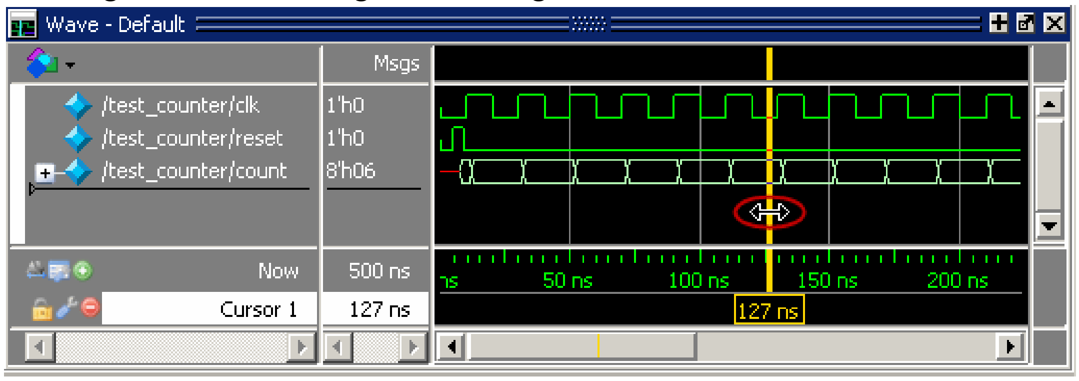
1. Position the cursor by clicking in the cursor timeline then dragging.

   a. Click the **Select Mode** icon on the Wave window toolbar.

   b. Click anywhere in the cursor timeline.

   The cursor snaps to the time where you clicked (Figure 6-3).

Figure 6-3. Working with a Single Cursor in the Wave Window



   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, position the mouse pointer over the cursor line. When the pointer changes to a two-headed arrow (Figure 6-3), click and hold the left mouse button to select the cursor. Drag the cursor to the right of a transition.

The cursor *snaps* to the nearest transition to the left when you release the mouse button. Cursors *snap* to a waveform edge when you drag a cursor to within ten pixels of an edge. You can set the snap distance in the Window Preferences dialog box (select **Tools > Window Preferences**).

e. In the cursor timeline pane, select the yellow timeline indicator box then drag the cursor to the right of a transition (Figure 6-3).

The cursor does not snap to a transition when you drag in the timeline pane.

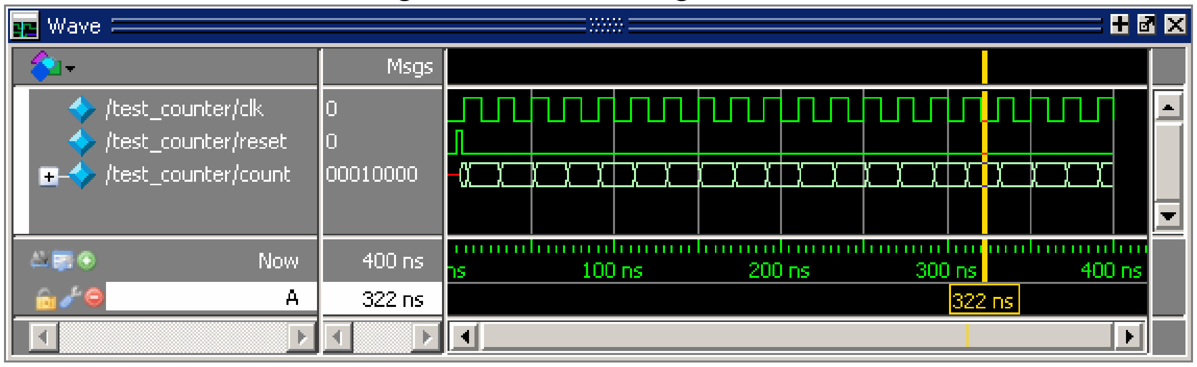
2. Rename the cursor.

a. Right-click **Cursor 1** in the cursor pane, then select and delete the text.

b. Type **A** and press Enter.

The cursor name changes to **A** (Figure 6-4).

Figure 6-4. Renaming a Cursor



3. Jump the cursor to the next or previous transition.

a. Click signal **count** in the pathname pane.

b. Click the **Find Next Transition** icon on the Wave window toolbar.

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

c. Click the **Find Previous Transition** icon on the Wave window toolbar.

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

Working with Multiple Cursors

Even more information is available when working with multiple cursors.

Procedure

1. Add a second cursor.

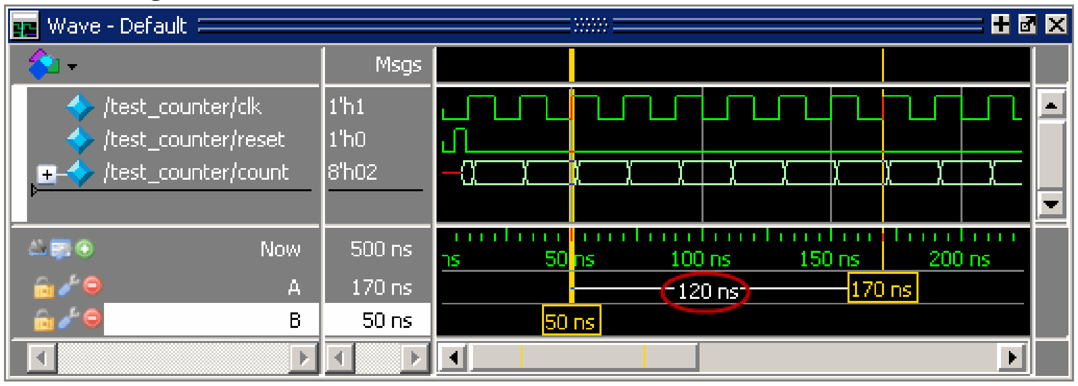
   a. Click the **Insert Cursor** icon on the Wave window toolbar.

   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 6-5).

Figure 6-5. Interval Measurement Between Two Cursors



Wave - Default

	Msgs
/test_counter/clk	1'h1
/test_counter/reset	1'h0
/test_counter/count	8'h02
Now	500 ns
A	170 ns
B	50 ns

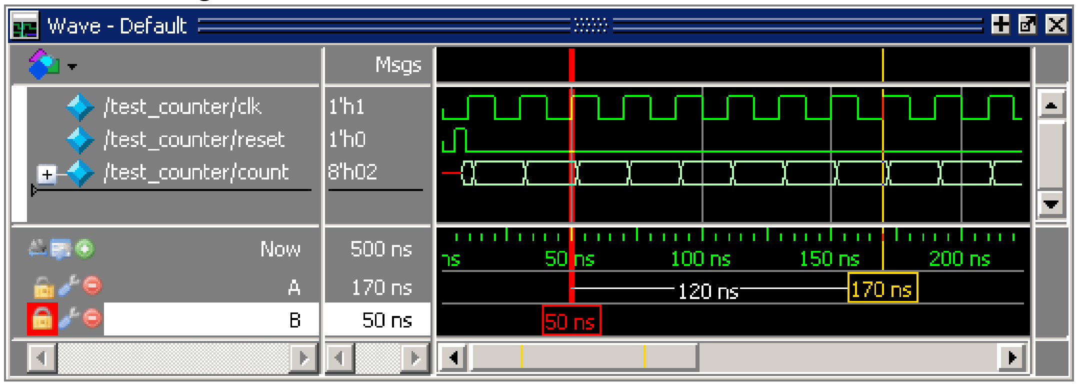
2. Lock cursor *B*.

   a. Right-click the yellow time indicator box associated with cursor *B* (at 56 ns).

   b. Select **Lock B** from the popup menu.

   The cursor color changes to red and you can no longer drag the cursor (Figure 6-6).

Figure 6-6. A Locked Cursor in the Wave Window



Wave - Default

	Msgs
/test_counter/clk	1'h1
/test_counter/reset	1'h0
/test_counter/count	8'h02
Now	500 ns
A	170 ns
B	50 ns

3. Delete cursor *B*.

a. Right-click cursor *B* (the red box at 56 ns) and select **Delete B**.

Lesson Wrap-Up

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

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

Related Topics

Wave Window [ModelSim GUI Reference Manual]

Recording Simulation Results With Datasets [ModelSim User's Manual]

Chapter 7
Viewing And Initializing Memories

In this lesson you will learn how to view and initialize memories.

ModelSim defines and lists any of the following as memories:

- 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 installation comes with Verilog and VHDL versions of the example design.

Example files are located in the following directories:

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

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

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

For further information, refer to Memory List Window in the GUI Reference Manual, and the mem display, mem load, mem save, and radix commands.

Compile and Load the Design

Before viewing and initializing memories we need to compile and load a design.

Procedure

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/tutorials/verilog/memory* to the new directory.

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

2. Start ModelSim and change to the *exercise* directory.

   If you just finished a different 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 box 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 *.v** at the ModelSim> prompt to compile all verilog files in the design.

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

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.

View a Memory and its Contents

The Memory List window lists all memory instances in the design, showing for each instance the range, depth, and width. Double-clicking an instance opens a window displaying the memory data.

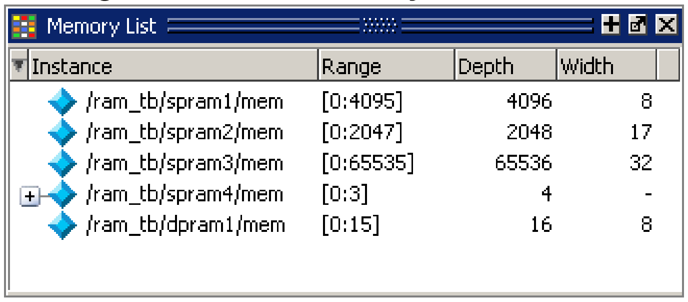
Procedure

1. Open the Memory List window and view the data of a memory instance.

   a. If the Memory List window is not already open, select **View > Memory List**.

A Memory List window is shown in Figure 7-1.

Figure 7-1. The Memory List Window



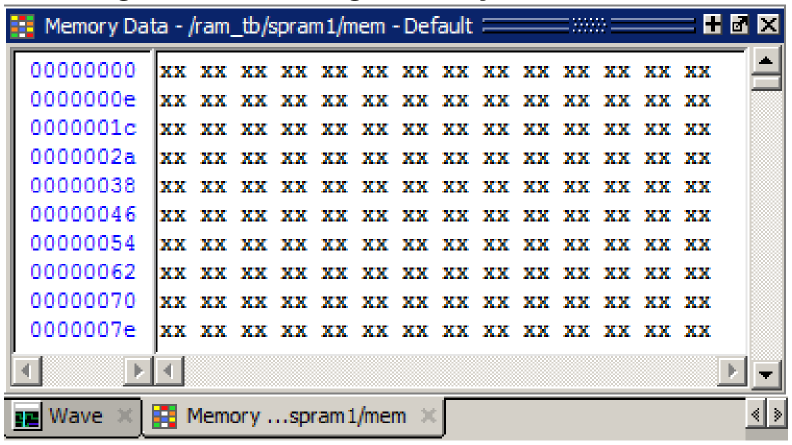
Instance	Range	Depth	Width
/ram_tb/spram1/mem	[0:4095]	4096	8
/ram_tb/spram2/mem	[0:2047]	2048	17
/ram_tb/spram3/mem	[0:65535]	65536	32
/ram_tb/spram4/mem	[0:3]	4	-
/ram_tb/dpram1/mem	[0:15]	16	8

b. Double-click the */ram_tb/spram1/mem* instance in the memory list to view its contents.

A Memory Data window opens displaying the contents of spram1. The first column (blue hex characters) lists the addresses, and the remaining columns show the data values.

If you are using the Verilog example design, the data is all X (Figure 7-2) because you have not yet simulated the design.

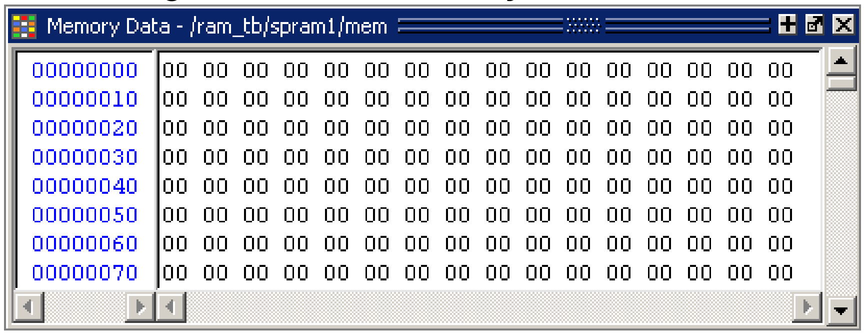
Figure 7-2. Verilog Memory Data Window



```
00000000 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
0000000e xx xx xx xx xx xx xx xx xx xx xx xx xx xx
0000001c xx xx xx xx xx xx xx xx xx xx xx xx xx xx
0000002a xx xx xx xx xx xx xx xx xx xx xx xx xx xx
00000038 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
00000046 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
00000054 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
00000062 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
00000070 xx xx xx xx xx xx xx xx xx xx xx xx xx xx
0000007e xx xx xx xx xx xx xx xx xx xx xx xx xx xx
```

If you are using the VHDL example design, the data is all zeros (Figure 7-3).

Figure 7-3. VHDL Memory Data Window



```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

c. Double-click the instance */ram_tb/spram2/mem* in the Memory List window. This opens a second Memory Data window that contains the addresses and data for the spram2 instance. For each memory instance that you click in the Memory List window, a new Memory Data window opens.

2. Simulate the design.

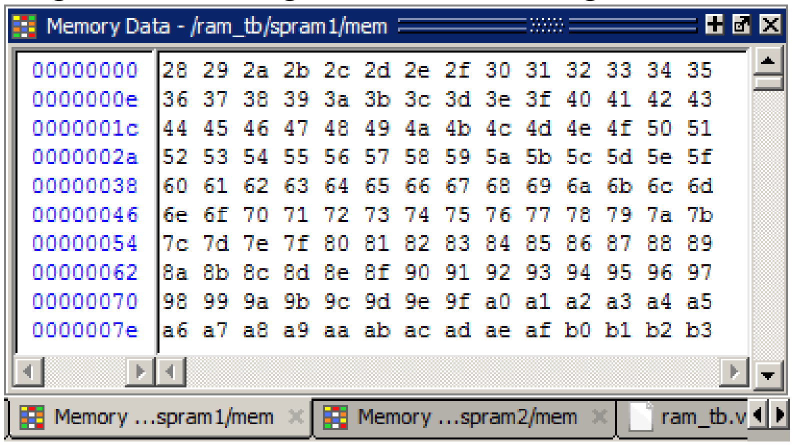
a. Click the **Run -All** icon in the Main window.

A Source window opens showing the source code for the *ram_tb* file at the point where the simulation stopped.

VHDL: In the Transcript window, you will see NUMERIC_STD warnings that can be ignored and an assertion failure that is functioning to stop the simulation. The simulation itself has not failed.

a. Click the **Memory ...spram1/mem** tab to bring that Memory data window to the foreground. The Verilog data fields are shown in Figure 7-4.

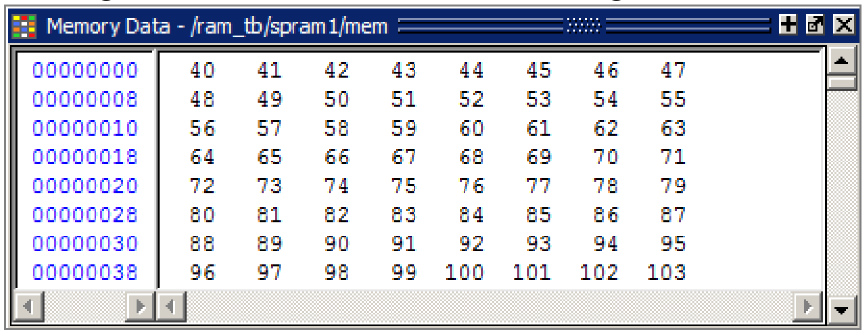
Figure 7-4. Verilog Data After Running Simulation



```
Memory Data - /ram_tb/spram1/mem

00000000  28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35
0000000e  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43
0000001c  44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 51
0000002a  52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
00000038  60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d
00000046  6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b
00000054  7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89
00000062  8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97
00000070  98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5
0000007e  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3
```

The VHDL data fields are shown in Figure 7-5.

Figure 7-5. VHDL Data After Running Simulation



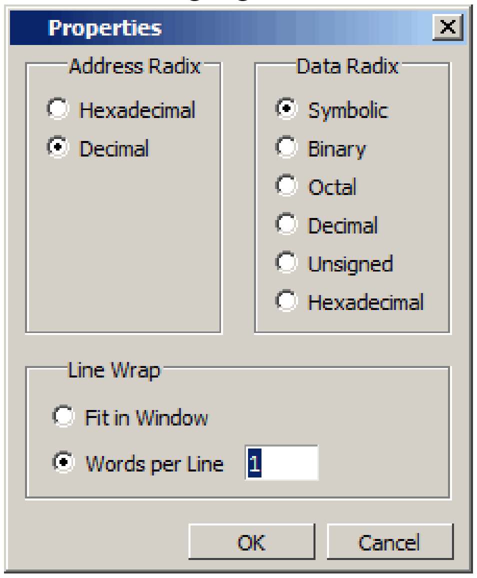
```
Memory Data - /ram_tb/spram1/mem

00000000   40   41   42   43   44   45   46   47
00000008   48   49   50   51   52   53   54   55
00000010   56   57   58   59   60   61   62   63
00000018   64   65   66   67   68   69   70   71
00000020   72   73   74   75   76   77   78   79
00000028   80   81   82   83   84   85   86   87
00000030   88   89   90   91   92   93   94   95
00000038   96   97   98   99  100  101  102  103
```

3. Change the address radix and the number of words per line for instance */ram_tb/spram1/ mem*.

a. Right-click anywhere in the spram1 Memory Data window and select **Properties**.

b. The Properties dialog box opens (Figure 7-6).

Figure 7-6. Changing the Address Radix



Address Radix: Hexadecimal [✓ Decimal]
Data Radix: [✓ Symbolic] Binary Octal Decimal Unsigned Hexadecimal
Line Wrap: Fit in Window [✓ Words per Line] 1

c. For the Address Radix, select **Decimal**. This changes the radix for the addresses only.

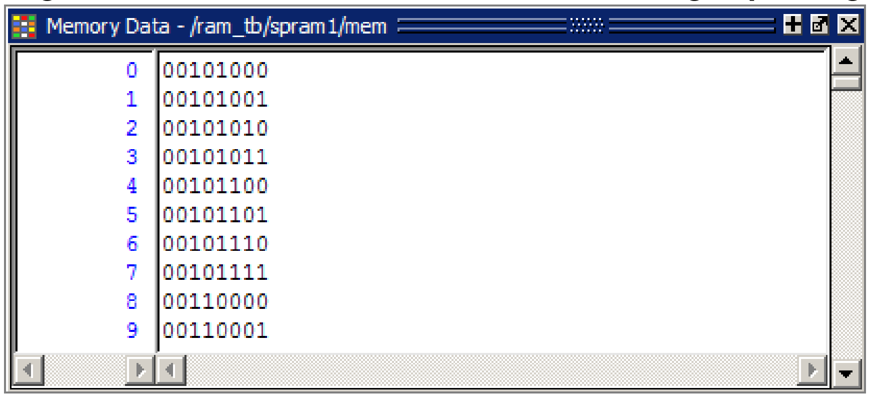
d. Change the Data Radix to **Symbolic**.

e. Select Words per line and type **1** in the field.

f. Click **OK**.

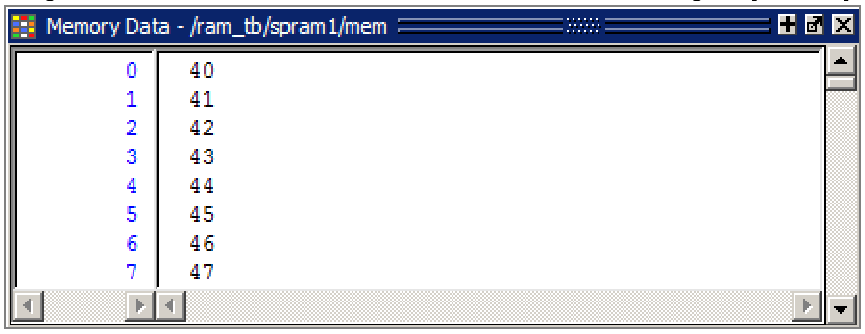
You can see the Verilog results of the settings in Figure 7-7 and the VHDL results in Figure 7-8. If the figure does not match what you have in your ModelSim session, check to make sure you set the Address Radix rather than the Data Radix. Data Radix should still be set to Symbolic, the default.

Figure 7-7. New Address Radix and Line Length (Verilog



```
0 00101000
1 00101001
2 00101010
3 00101011
4 00101100
5 00101101
6 00101110
7 00101111
8 00110000
9 00110001
```

Figure 7-8. New Address Radix and Line Length (VHDL)



```
0   40
1   41
2   42
3   43
4   44
5   45
6   46
7   47
```

Navigate 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.

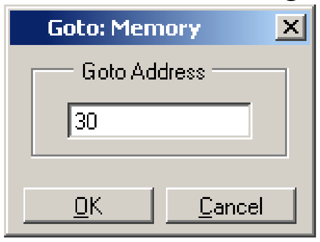
Procedure

1. Use Goto to find a specific address.

   a. Right-click anywhere in the address column and select Goto .

   The Goto dialog (Figure 7-9) box opens in the data pane.

   Figure 7-9. Goto Dialog Box

   Goto Address: 30

   b. Type **30** in the Goto Address field.

   c. Click **OK**.

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

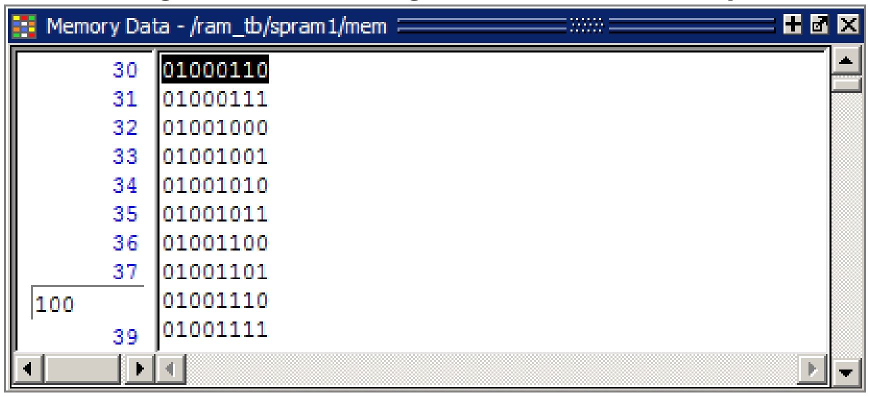
2. Edit the address location directly.

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

      i. Double click address 38 in the address column.

      ii. Enter address 100 (Figure 7-10).

Figure 7-10. Editing the Address Directly



Memory Data - /ram_tb/spram1/mem

```
   30 01000110
   31 01000111
   32 01001000
   33 01001001
   34 01001010
   35 01001011
   36 01001100
   37 01001101
100   01001110
   39 01001111
```

iii. Press the Enter or Return key on your keyboard.

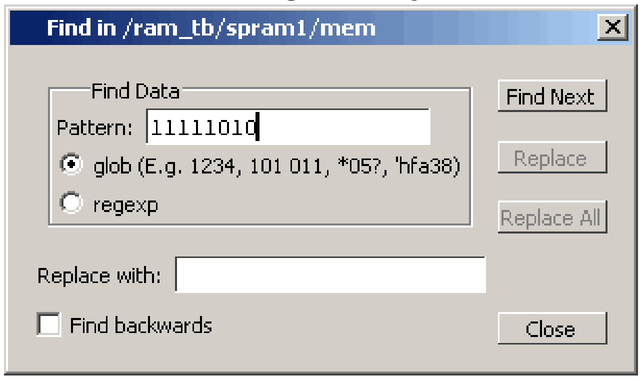
The pane jumps to address 100.

3. Now, find a particular data entry.

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

      The Find in dialog box opens (Figure 7-11).

Figure 7-11. Searching for a Specific Data Value



Find in /ram_tb/spram1/mem

Find Data
Pattern: 11111010
[✓ glob (E.g. 1234, 101 011, *05?, 'hfa38)] regexp
Replace with:
Find backwards

   b. **Verilog:** Type **11111010** in the Find Data field and click **Find Next**.

      VHDL: Type **250** 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.

Export Memory Data to a File

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

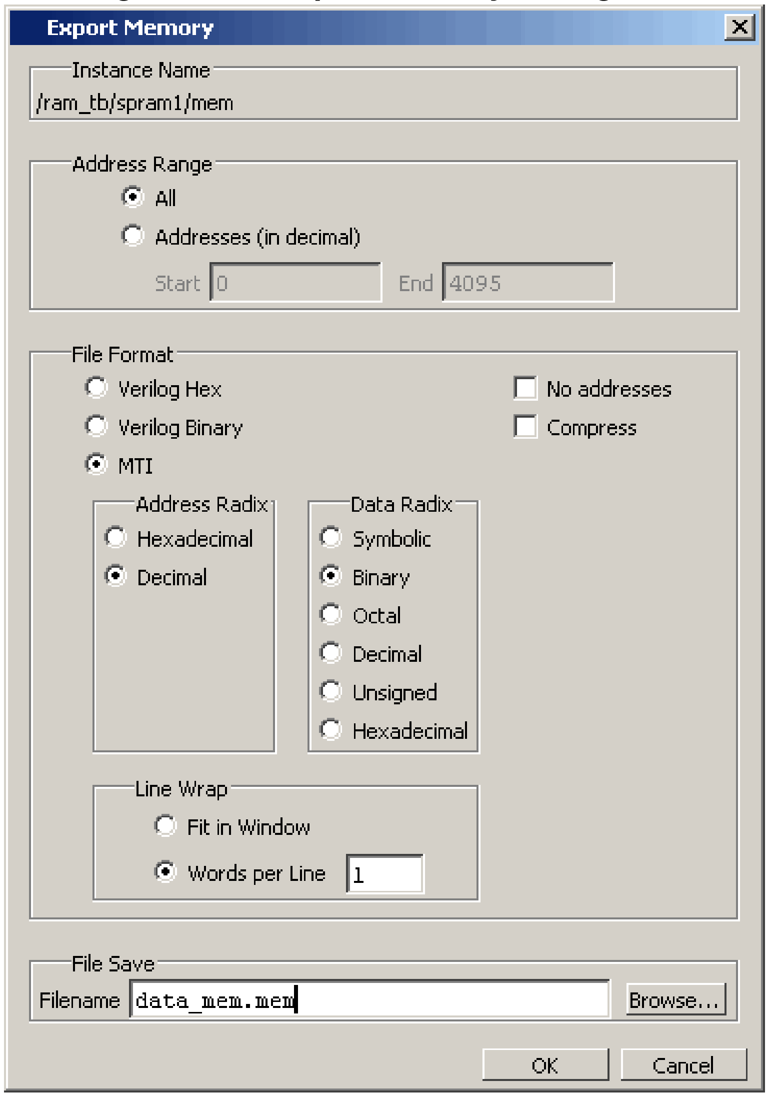
Procedure

1. Export a memory pattern from the */ram_tb/spram1/mem* instance to a file.

   a. Make sure */ram_tb/spram1/mem* is open and selected.

   b. Select **File** > **Export** > **Memory Data** to bring up the Export Memory dialog box (Figure 7-12).

Figure 7-12. Export Memory Dialog Box



Instance Name: /ram_tb/spram1/mem

Address Range: [✓ All] Addresses (in decimal) — Start 0, End 4095

File Format: Verilog Hex, Verilog Binary, [✓ MTI]; No addresses ☐, Compress ☐

Address Radix: Hexadecimal, [✓ Decimal]

Data Radix: Symbolic, [✓ Binary], Octal, Decimal, Unsigned, Hexadecimal

Line Wrap: Fit in Window, [✓ Words per Line] 1

File Save — Filename: data_mem.mem

   c. For the Address Radix, select **Decimal**.

   d. For the Data Radix, select **Binary**.

   e. For the Words per Line, set to 1.

   f. Type **data_mem.mem** into the Filename field.

   g. Click **OK**.

You can view the exported file in any editor.

Memory pattern files can be exported 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. Export a relocatable memory pattern file from the */ram_tb/spram2/mem* instance.
    a. Select the Memory Data window for the */ram_tb/spram2/mem* instance.
    b. Right-click on the memory contents to open a popup menu and select Properties.
    c. In the Properties dialog box, set the Address Radix to Decimal; the Data Radix to Binary; and the Line Wrap to 1 Words per Line. Click **OK** to accept the changes and close the dialog box.
    d. Select **File > Export > Memory Data**to bring up the Export Memory dialog box.
    e. For the Address Range, specify a Start address of **0** and End address of **250**.
    f. For the File Format, select MTI and **No addresses** to create a memory pattern that you can use to relocate somewhere else in the memory, or in another memory.
    g. For Address Radix select Decimal, and for Data Radix select Binary.
    h. For the Words per Line, set to 1.
    i. Enter the filename as *reloc.mem*, then click **OK** to save the memory contents and close the dialog box. You will use this file for initialization in the next section.

Initialize a Memory

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

First, you will initialize a memory from a file only. You will use the one you exported previously, *data_mem.mem*.

Procedure

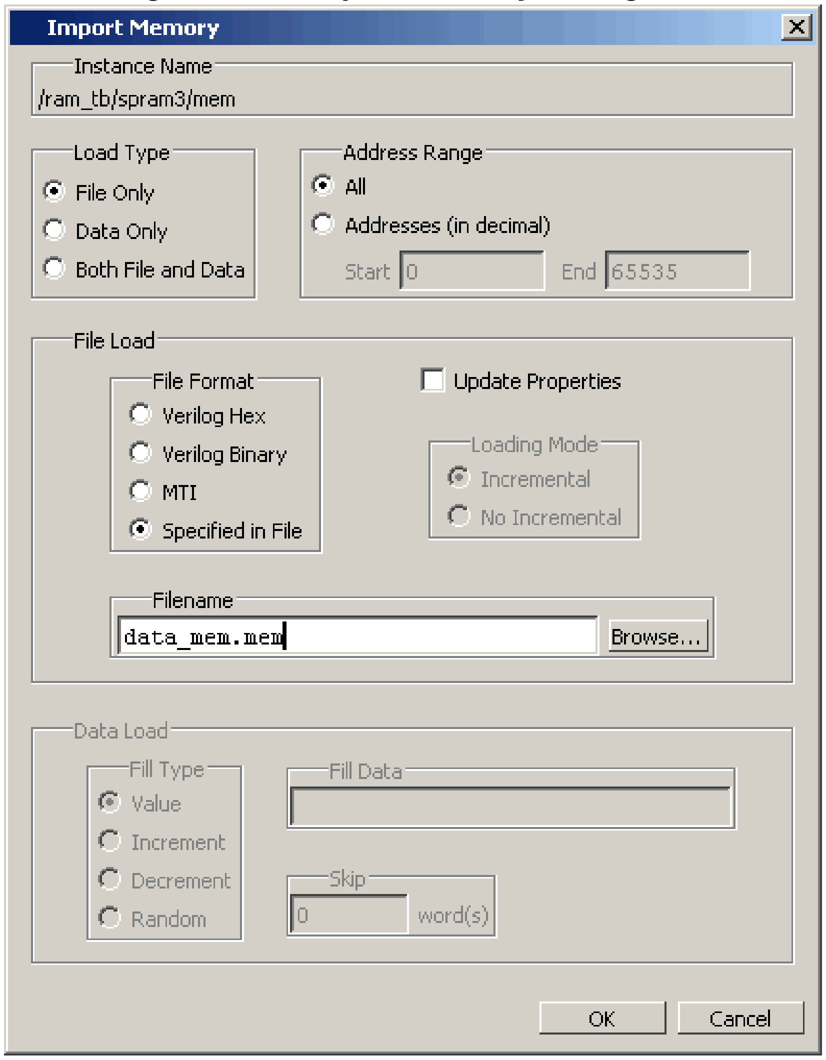
1. View instance */ram_tb/spram3/mem*.
    a. Double-click the */ram_tb/spram3/mem* instance in the Memory List window.

       This will open a new Memory Data window to display the contents of */ram_tb/spram3/mem*. Familiarize yourself with the contents so you can identify changes once the initialization is complete.
    b. Right-click and select **Properties** to bring up the Properties dialog box.
    c. Change the Address Radix to **Decimal**, Data Radix to **Binary**, Words per Line to **1**, and click **OK**.

2. Initialize *spram3* from a file.

   a. Right-click anywhere in the data column and select Import Data Patterns to bring up the Import Memory dialog box (Figure 7-13).

Figure 7-13. Import Memory Dialog Box



Import Memory

Instance Name: /ram_tb/spram3/mem

Load Type: [✓ File Only] Data Only Both File and Data

Address Range: [✓ All] Addresses (in decimal) Start 0 End 65535

File Load

File Format: Verilog Hex Verilog Binary MTI [✓ Specified in File]

Update Properties ☐

Loading Mode: [✓ Incremental] No Incremental

Filename: data_mem.mem

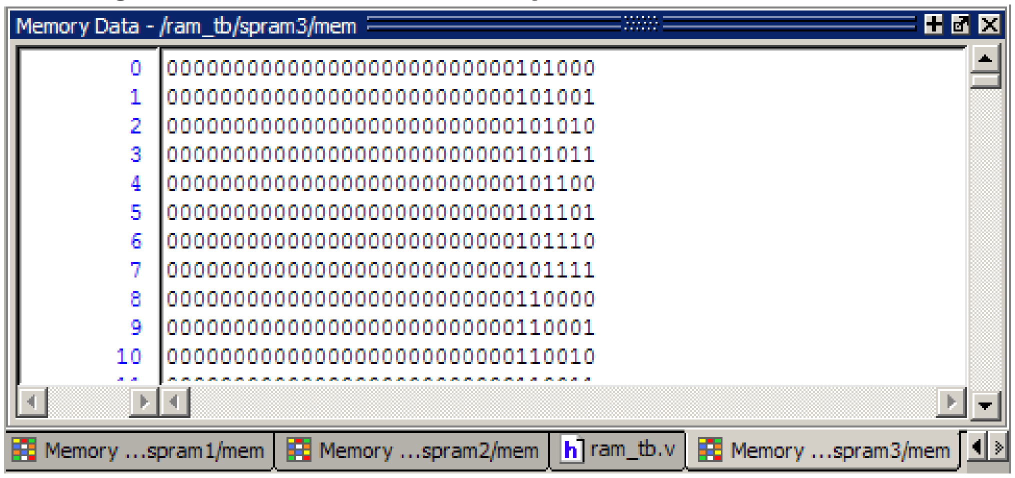
The default Load Type is File Only.

   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 7-14).

Figure 7-14. Initialized Memory from File and Fill Pattern



Memory Data - /ram_tb/spram3/mem

```
 0  00000000000000000000000000101000
 1  00000000000000000000000000101001
 2  00000000000000000000000000101010
 3  00000000000000000000000000101011
 4  00000000000000000000000000101100
 5  00000000000000000000000000101101
 6  00000000000000000000000000101110
 7  00000000000000000000000000101111
 8  00000000000000000000000000110000
 9  00000000000000000000000000110001
10  00000000000000000000000000110010
```

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

3. Import 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 *spram3* and select **Import Data Patterns** to bring up the Import Memory dialog box.

   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, select the **MTI File Format** and enter *reloc.mem* in the Filename field.

   e. For Data Load, select a Fill Type of **Increment**.

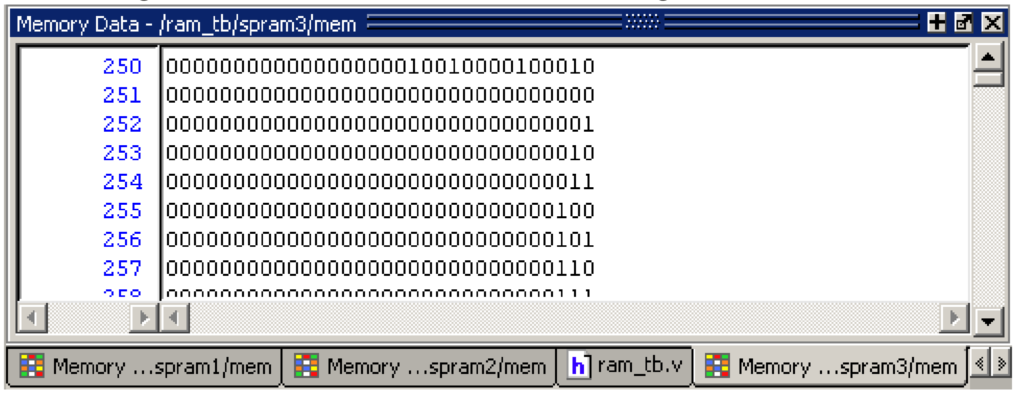
   f. In the Fill Data field, set the seed value of **0** for the incrementing data.

   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 7-15).

Figure 7-15. Data Increments Starting at Address 251



```
250  00000000000000000010010000100010
251  00000000000000000000000000000000
252  00000000000000000000000000000001
253  00000000000000000000000000000010
254  00000000000000000000000000000011
255  00000000000000000000000000000100
256  00000000000000000000000000000101
257  00000000000000000000000000000110
```

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

4. Right-click in one of the Memory Data windows and select **Close All**.

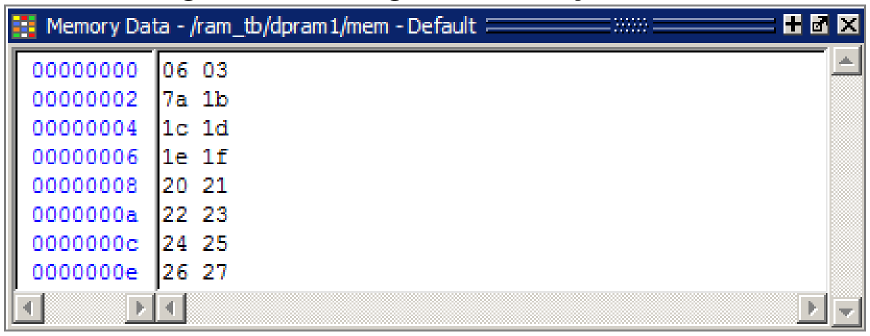
Interactive Debugging Commands

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

Procedure

1. Open a memory instance and change its display characteristics.
   a. Double-click instance */ram_tb/dpram1/mem* in the Memory List window.
   b. Right-click in the *dpram1* Memory Data window and select **Properties**.
   c. Change the Address and Data Radix to **Hexadecimal**.
   d. Select **Words per line** and enter **2**.
   e. Click **OK**. The result should be as shown in Figure 7-16.

Figure 7-16. Original Memory Content

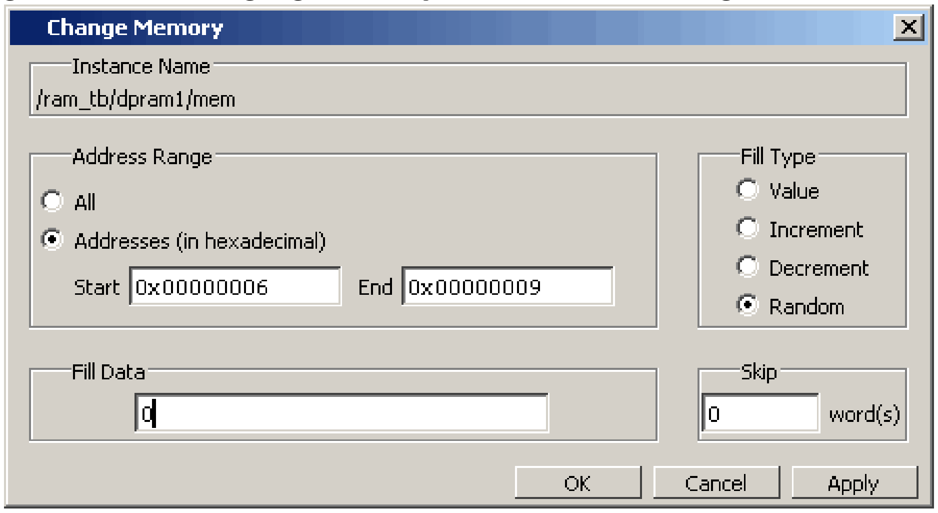


```
00000000  06 03
00000002  7a 1b
00000004  1c 1d
00000006  1e 1f
00000008  20 21
0000000a  22 23
0000000c  24 25
0000000e  26 27
```

2. Initialize a range of memory addresses from a fill pattern.

a. Right-click in the data column of */ram_tb/dpram1/mem* and select **Change** to open the Change Memory dialog box (Figure 7-17).

Figure 7-17. Changing Memory Content for a Range of Addresses


Change Memory

Instance Name: /ram_tb/dpram1/mem

Address Range: All [✓ Addresses (in hexadecimal)]

Start: 0x00000006 End: 0x00000009

Fill Type: Value / Increment / Decrement / [✓ Random]

Fill Data: 0

Skip: 0 word(s)

b. Select **Addresses** and enter the start address as **0x00000006** and the end address as **0x00000009**. The "0x" hex notation is optional.

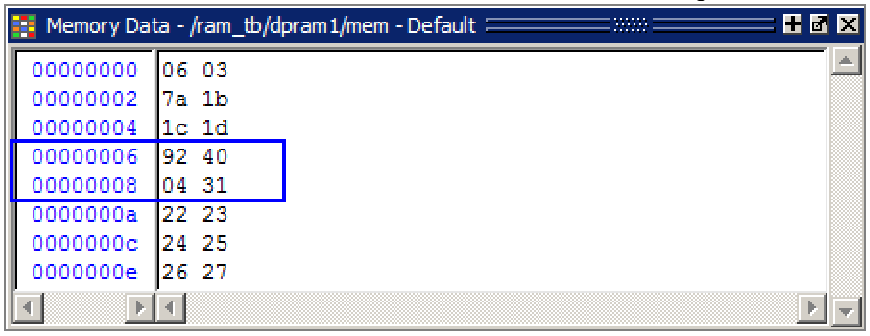
c. Select **Random** as the Fill Type.

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 7-18).

Figure 7-18. Random Content Generated for a Range of Addresses


Memory Data - /ram_tb/dpram1/mem - Default

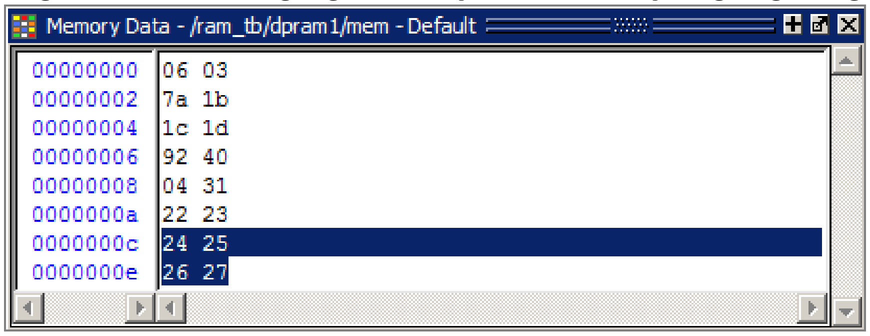
```
00000000 06 03
00000002 7a 1b
00000004 1c 1d
00000006 92 40
00000008 04 31
0000000a 22 23
0000000c 24 25
0000000e 26 27
```

3. Change contents by highlighting.

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

   a. Highlight the data for the addresses **0x0000000c:0x0000000e**, as shown in Figure 7-19.

Figure 7-19. Changing Memory Contents by Highlighting



```
00000000 06 03
00000002 7a 1b
00000004 1c 1d
00000006 92 40
00000008 04 31
0000000a 22 23
0000000c 24 25
0000000e 26 27
```

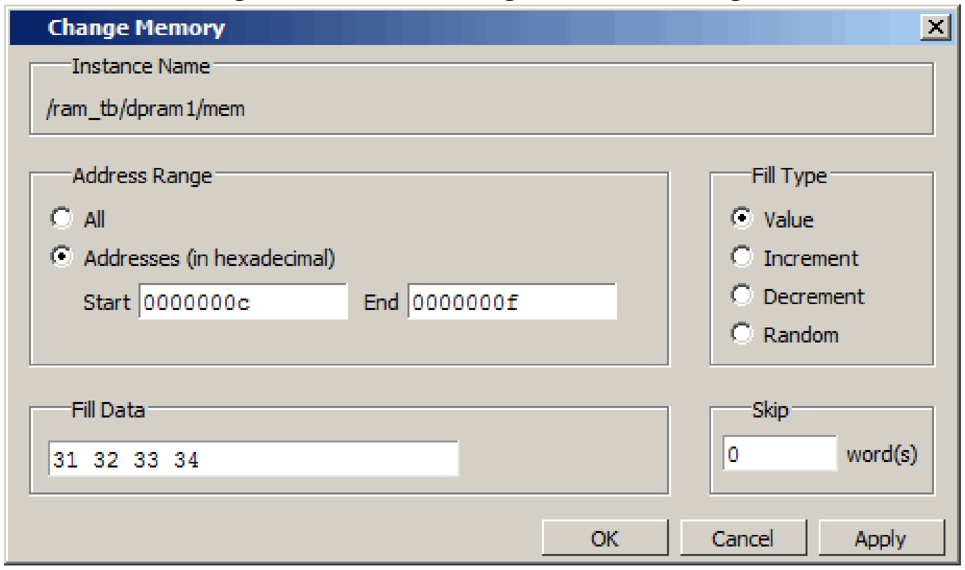
b. Right-click the highlighted data and select **Change**.

This brings up the Change memory dialog box. Note that the Addresses field is already populated with the range you highlighted.

c. Select **Value** as the Fill Type. (Refer to Figure 7-20)

d. Enter the data values into the Fill Data field as follows: **31 32 33 34.**

Figure 7-20. Entering Data to Change



Instance Name: /ram_tb/dpram1/mem

Address Range: [ ] All [✓ Addresses (in hexadecimal)]
Start 0000000c End 0000000f

Fill Type: [✓ Value] [ ] Increment [ ] Decrement [ ] Random

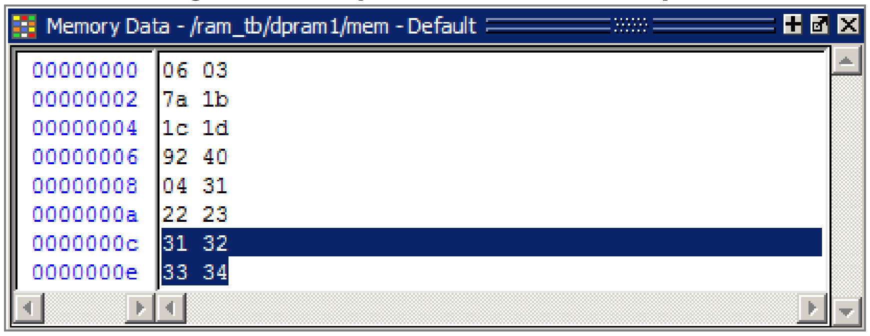
Fill Data: 31 32 33 34

Skip: 0 word(s)

e. Click **OK**.

The data in the address locations change to the values you entered (Figure 7-21).

Figure 7-21. Changed Memory Contents for the Specified Addresses



```
00000000  06 03
00000002  7a 1b
00000004  1c 1d
00000006  92 40
00000008  04 31
0000000a  22 23
0000000c  31 32
0000000e  33 34
```

4. Edit data in place.

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

   a. Double-click any value in the Data column.

   b. Enter the desired value and press the Enter or Return key on your keyboard.

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

Lesson Wrap-Up

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

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

Chapter 8
Automating Simulation

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, through 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.

Note

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

Creating a Simple DO File

Creating a DO file is as simple as typing a set of commands in a text file. In this exercise, you will create a DO file that loads a design, adds signals to the Wave window, provides stimulus to those signals, and then advances the simulation. You can also create a DO file from a saved transcript file.

Refer to “Saving a Transcript File as a DO file” in the GUI Reference Manual.

Procedure

1. Change to the directory you created in the “Basic Simulation” lesson.
2. Create a DO file that will add signals to the Wave window, force signals, and run the simulation.
   a. Select **File > New > Source > Do** to create a new DO file.

b. Enter the following commands into the Source window:

```
vsim test_counter
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**.

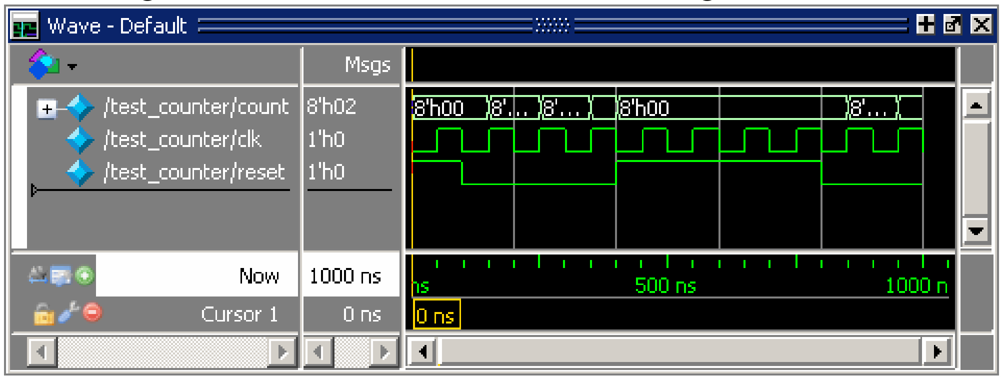
   b. Type *sim.do* in the File name: field and save it to the current directory.

4. Execute the DO file.

   a. Enter *do sim.do* at the VSIM> prompt.

   ModelSim loads the design, executes the saved commands and draws the waves in the Wave window. (Figure 8-1)

Figure 8-1. Wave Window After Running the DO File



5. When you are done with this exercise, select **File** > **Quit** to quit ModelSim.

Running 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 (for example, vsim, vlib, vlog, and so on) 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.

Procedure

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 the following files into it:

   - */<install_dir>/examples/tutorials/verilog/automation/counter.v*
   - */<install_dir>/examples/tutorials/verilog/automation/stim.do*

   This lesson uses the Verilog file *counter.v*. If you have a VHDL license, use *the counter.vhd* and *stim.do* files in the */<install_dir>/examples/tutorials/vhdl/automation* directory 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*.

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 *

#change radix to symbolic
radix -symbolic

# 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 command line mode simulation.

   a. Enter the following command at the DOS/UNIX prompt:

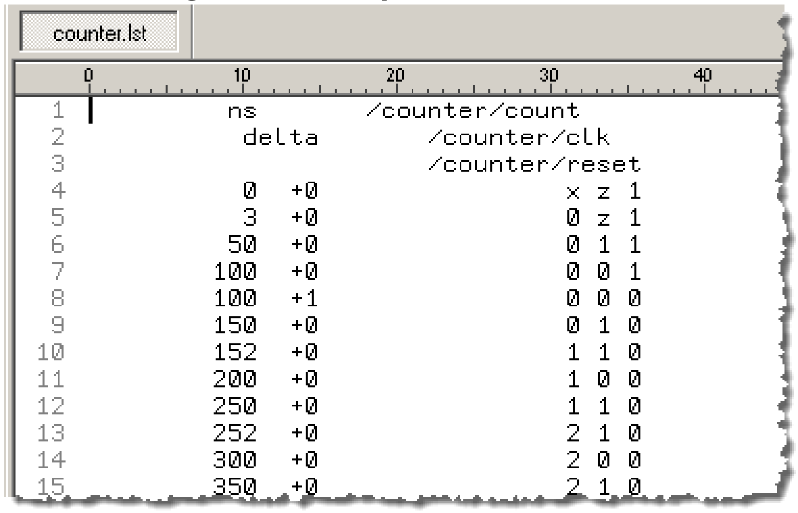
      vsim -c -do sim.do counter -wlf counter.wlf

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. Output produced by the Verilog version of the design should look like Figure 8-2:

Figure 8-2. Output of the Counter



```
 1             ns        /counter/count
 2               delta          /counter/clk
 3                              /counter/reset
 4              0   +0                    x  z  1
 5              3   +0                    0  z  1
 6             50   +0                    0  1  1
 7            100   +0                    0  0  1
 8            100   +1                    0  0  0
 9            150   +0                    0  1  0
10            152   +0                    1  1  0
11            200   +0                    1  0  0
12            250   +0                    1  1  0
13            252   +0                    2  1  0
14            300   +0                    2  0  0
15            350   +0                    2  1  0
```

The output may appear slightly different if you used the VHDL version.

6. View the results in the GUI.

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

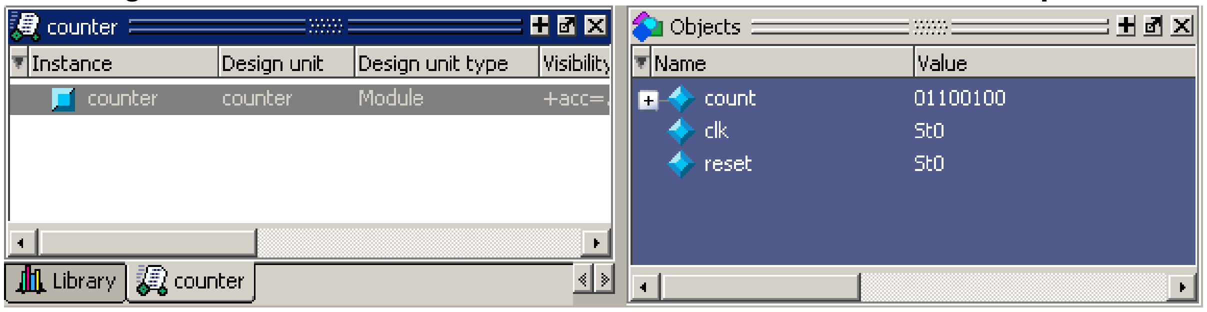
Note

Make sure your PATH environment variable is set with the current version of ModelSim at the front of the string.

   a. Type vsim -view counter.wlf at the prompt.

      The GUI opens and a dataset tab named "counter" is displayed (Figure 8-3).

Figure 8-3. The counter.wlf Dataset in the Main Window Workspace



Instance	Design unit	Design unit type	Visibility
counter	counter	Module	+acc=

Name	Value
count	01100100
clk	St0
reset	St0

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

The waveforms display in the Wave window.

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

Using Tcl with the Simulator

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.

Procedure

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 - 100] [expr $stime + 50]" 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 100 time units to the current simulation time plus 50 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 {[examine count]== "8'h27"} {
      add_wave_zoom $now 1
   } elseif {[examine count]== "8'h47"} {
      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” into the directory you created in the Basic Simulation lesson.

2. Load the test_counter design unit and make sure the radix is set to binary.

   a. Start ModelSim.

   b. Select **File > Change Directory** and change to the directory you saved the DO file to above (the directory you created in the Basic Simulation lesson).

   c. Type radix -binary at the ModelSim > prompt

   d. Enter the following command at the ModelSim > prompt:

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.

   The simulation runs and the DO file creates two bookmarks.

   c. If the Wave window is docked in the Main window make it the active window (click anywhere in the Wave window), then select**Bookmarks > bk1**. If the window is undocked, select **Bookmarks > bk1** in the Wave window.

   Watch the Wave window zoom in and scroll to the time when count is 8'h27. Try the bk2 bookmark as well.

Lesson Wrap-Up

This concludes this lesson.

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

Related Topics

Tcl and DO Files in the User's Manual.

Index

Mentor Graphics
A Siemens Business

End-User License Agreement with EDA Software Supplemental Terms