ModelSim® Tutorial

Software Version 6.3c

September 2007

Table of Contents

List of Examples

List of Figures

List of Tables

Chapter 1
Introduction

Assumptions

We assume that you are familiar with the use of your operating system. You should also be familiar with the window management functions of your graphic interface: OpenWindows, OSF/Motif, CDE, KDE, GNOME, or Microsoft Windows 2000/XP.

We also assume that you have a working knowledge of the language in which your design and/or testbench is written (i.e., VHDL, Verilog, etc.). Although ModelSim™ is an excellent tool to use while learning HDL concepts and practices, this document is not written to support that goal.

Before you Begin

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

Examples show Windows path separators - use separators appropriate for your operating system when trying the examples.

Example Designs

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

Chapter 2
Conceptual Overview

Introduction

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

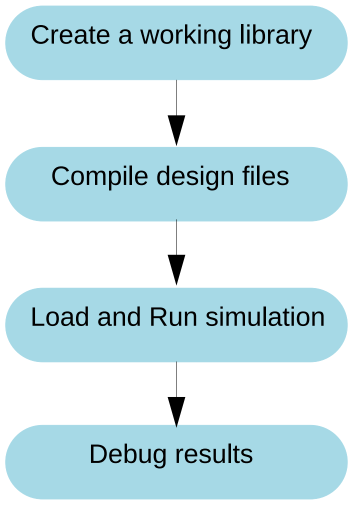
This lesson provides a brief conceptual overview of the ModelSim simulation environment. It is divided into fourtopics, which you will learn more about in subsequent lessons.

- Basic simulation flow — Refer to *Chapter 3 Basic Simulation*.
- Project flow — Refer to *Chapter 4 Projects*.
- Multiple library flow — Refer to *Chapter 5 Working With Multiple Libraries*.
- Debugging tools — Refer to remaining lessons.

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". "Work" is the library name used by the compiler as the default destination for compiled design units.

- Compiling Your Design

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

- 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

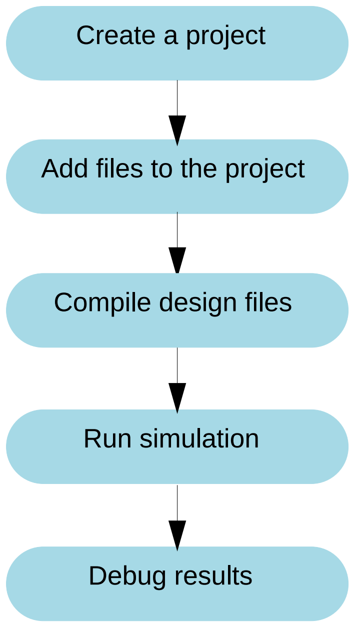
  If you don’t get the results you expect, you can use ModelSim’s robust debugging environment to track down the cause of the problem.

Project Flow

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

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

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.

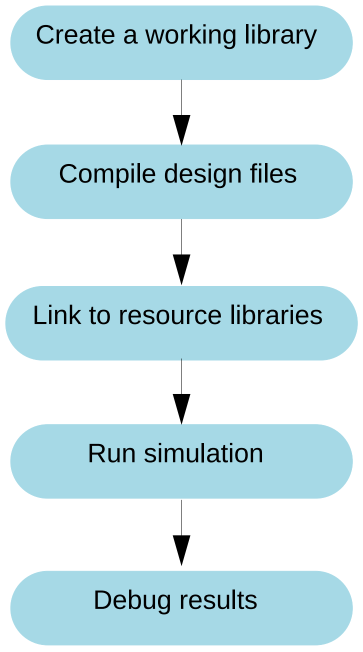
Multiple Library Flow

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 (e.g., a silicon vendor).

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

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

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 testbench to the project.

Debugging Tools

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

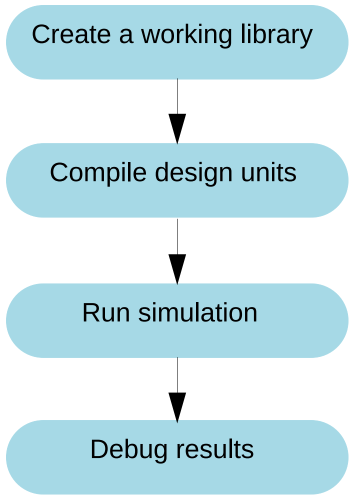
- Using projects
- Working with multiple libraries
- Setting breakpoints and stepping through the source code
- Viewing waveforms and measuring time
- Viewing and initializing memories
- Creating stimulus with the Waveform Editor
- Automating simulation

Chapter 3
Basic Simulation

Introduction

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

Figure 3-1. Basic Simulation Flow - Simulation Lab



Design Files for this Lesson

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

Verilog – *<install_dir>/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 testbench with the VHDL counter or vice versa.

Related Reading

User's Manual Chapters: Design Libraries, Verilog and SystemVerilog Simulation, and VHDL Simulation.

Reference Manual commands: vlib, vmap, vlog, vcom, view, and run.

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.

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. Click **Close**.

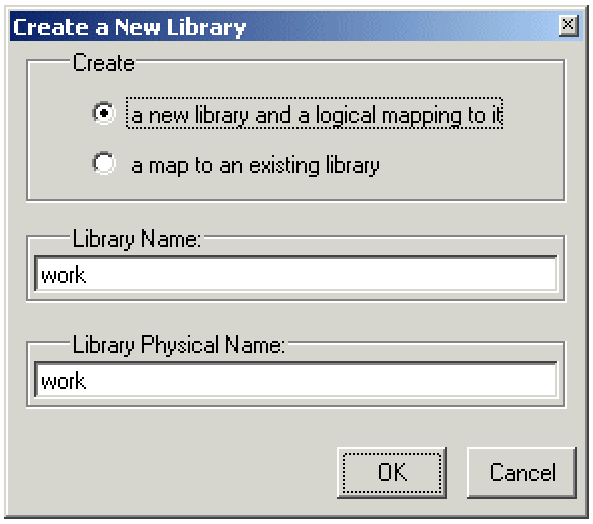
   b. Select **File > Change Directory** and change to the directory you created in step 1.

3. Create the working library.

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

      This opens a dialog where you specify physical and logical names for the library (Figure 3-2). You can create a new library or map to an existing library. We'll be doing the former.

Figure 3-2. The Create a New Library Dialog



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

Library Name: work

Library Physical Name: work

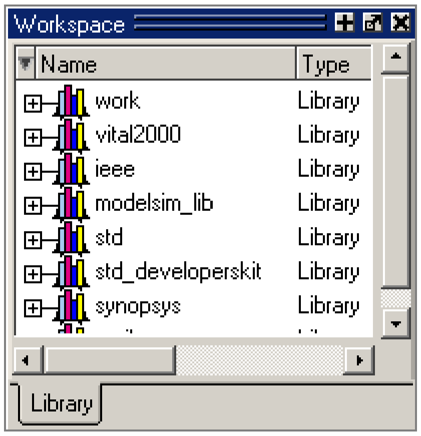
   b. Type **work** in the Library Name field (if it isn't already entered automatically).

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

Figure 3-3. work Library in the Workspace



Name	Type
work	Library
vital2000	Library
ieee	Library
modelsim_lib	Library
std	Library
std_developerskit	Library
synopsys	Library

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

```
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

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

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

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

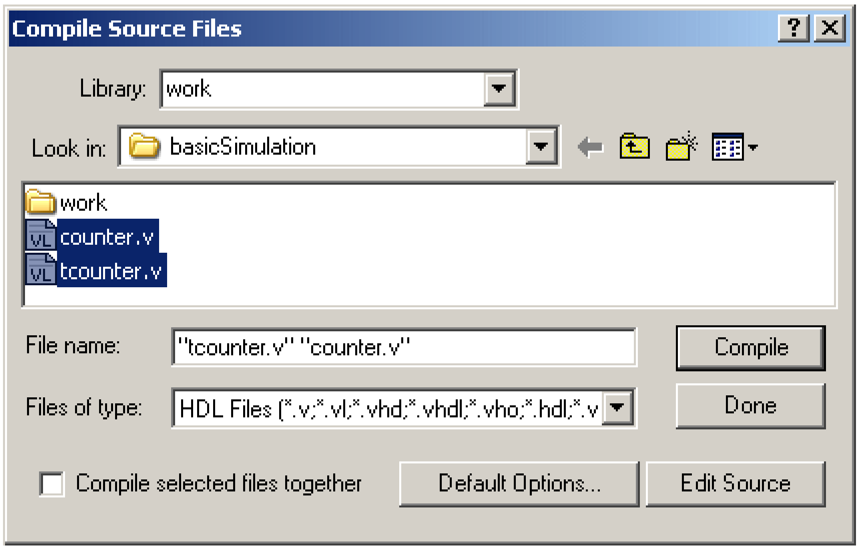
   a. Select **Compile > Compile**. This opens the Compile Source Files dialog (Figure 3-4).

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

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

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

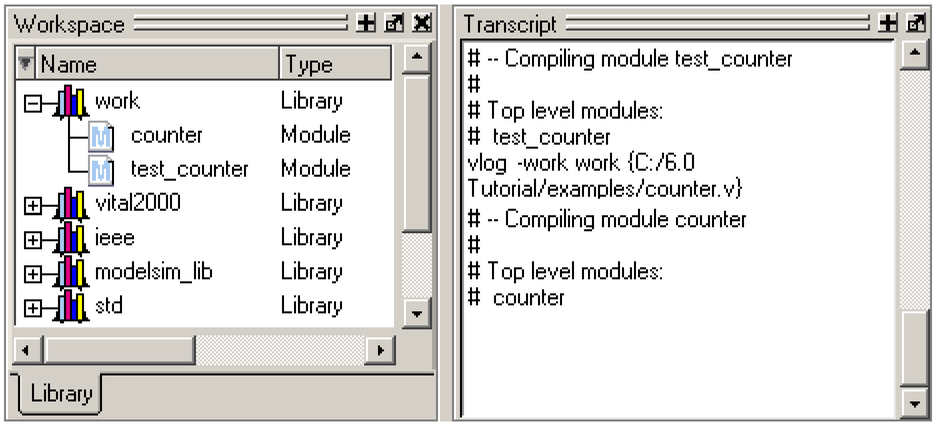
Figure 3-4. Compile Source Files Dialog


Compile Source Files
Library: work
Look in: basicSimulation
work
counter.v
tcounter.v
File name: "tcounter.v" "counter.v"
Files of type: HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.v
Compile
Done
Compile selected files together
Default Options...
Edit Source

2. View the compiled design units.

a. On the Library tab, click the '+' icon next to the *work* library and you will see two design units (Figure 3-5). You can also see their types (Modules, Entities, etc.) and the path to the underlying source files (scroll to the right if necessary).

Figure 3-5. Verilog Modules Compiled into work Library



```
# -- Compiling module test_counter
#
# Top level modules:
#  test_counter
vlog -work work {C:/6.0
Tutorial/examples/counter.v}
# -- Compiling module counter
#
# Top level modules:
#  counter
```

Load the Design

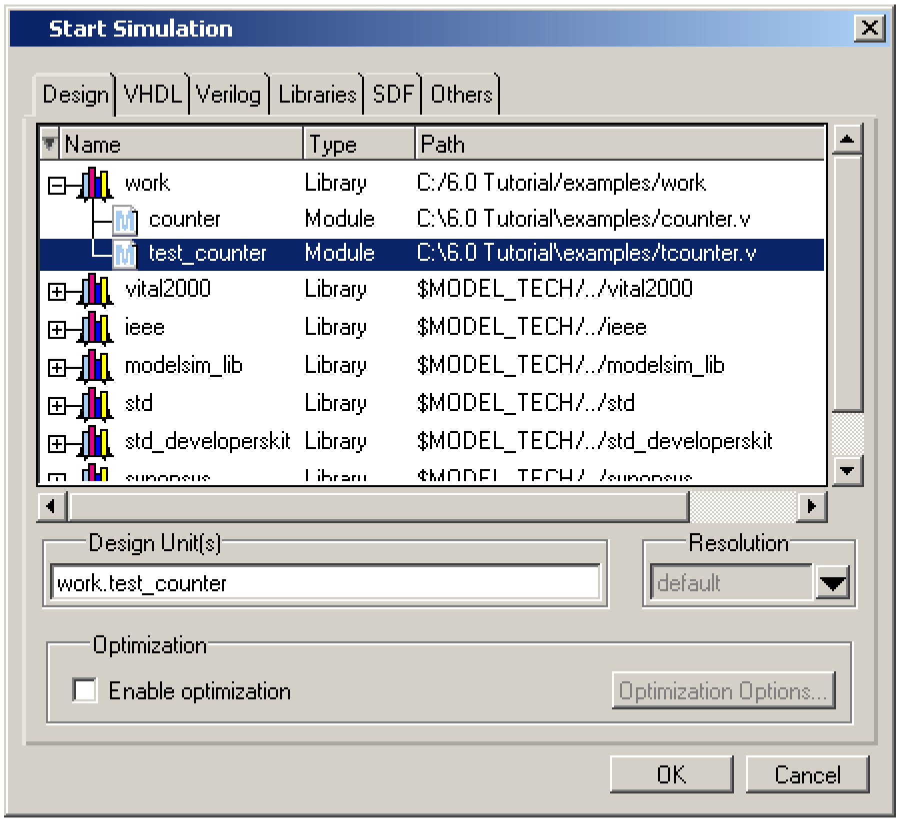
1. Load the *test_counter* module into the simulator.

a. In the Workspace, 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. 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 3-6).

Figure 3-6. Loading Design with Start Simulation Dialog



Name	Type	Path
work	Library	C:/6.0 Tutorial/examples/work
counter	Module	C:\6.0 Tutorial\examples/counter.v
test_counter	Module	C:\6.0 Tutorial\examples/tcounter.v
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std_developerskit

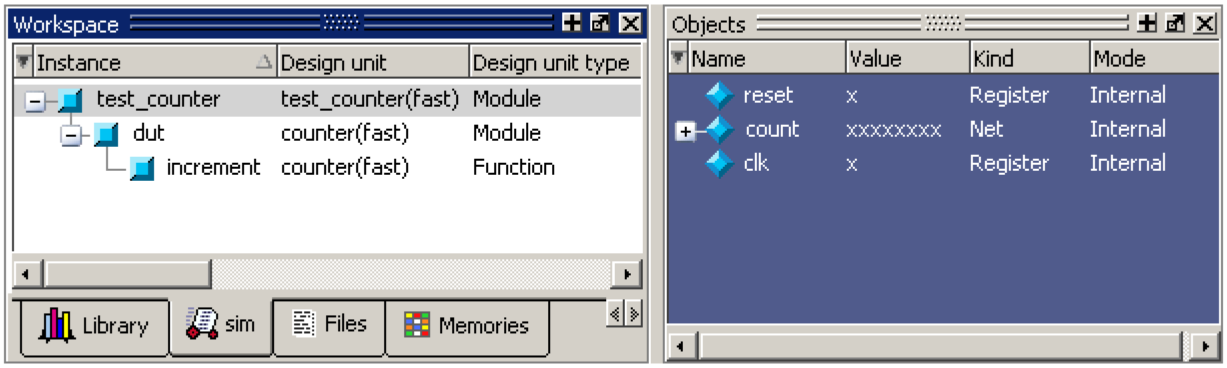
Design Unit(s): work.test_counter
Resolution: default
Optimization: Enable optimization

The command line equivalent for loading this design is:

```
vsim test_counter
```

When the design is loaded, you will see a new tab in the Workspace named *sim* that displays the hierarchical structure of the design (Figure 3-7). You can navigate within the hierarchy by clicking on any line with a ’+’ (expand) or ’-’ (contract) icon. You will also see a tab named *Files* that displays all files included in the design.

Figure 3-7. Workspace and Objects Panes Showing a Verilog Design



Instance	Design unit	Design unit type
test_counter	test_counter(fast)	Module
dut	counter(fast)	Module
increment	counter(fast)	Function

Name	Value	Kind	Mode
reset	x	Register	Internal
count	xxxxxxxx	Net	Internal
clk	x	Register	Internal

The Objects pane opens, by default, when a design is loaded. The Objects pane shows the names and current values of data objects in the current region (selected in the Workspace). Data objects include signals, nets, registers, constants and variables not declared in a process, generics, parameters.

You may open other windows and panes with the **View** menu. See Navigating the Interface.

Run the Simulation

Now you will open the Wave window, add signals to it, then run the simulation.

1. Open the Wave debugging window.

   a. Enter **view wave** at the command line.

      You can also use the **View** > **Wave** menu selection to open a Wave window.

      The Wave window is one of several windows available for debugging. To see a list of the other debuggin windows, select the **View** menu. You may need to move or resize the windows to your liking. Window panes within the Main window can be zoomed to occupy the entire Main window or undocked to stand alone. For details, see Navigating the Interface.

2. Add signals to the Wave window.

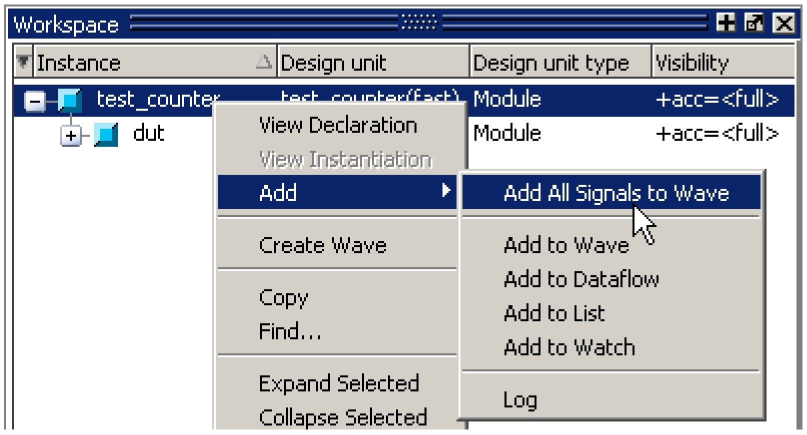
   a. In the Workspace pane, select the **sim** tab.

   b. Right-click *test_counter* to open a popup context menu.

   c. Select **Add** > **Add All Signals to 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.

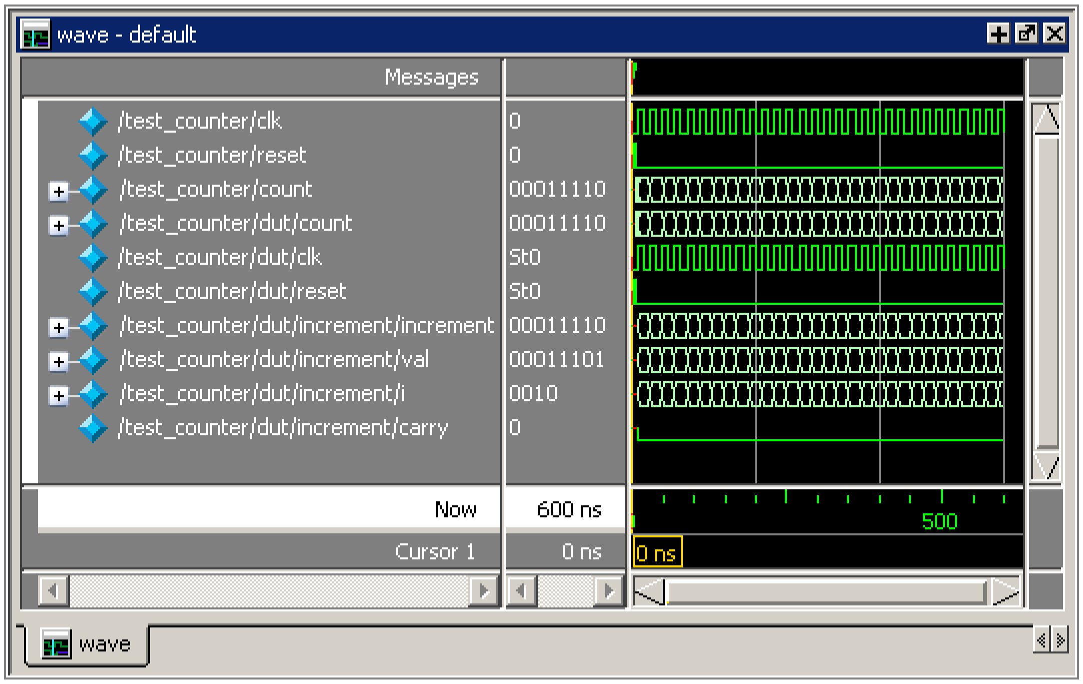
   a. Click the Run icon in the Main or Wave window toolbar.

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

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

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

Figure 3-9. Waves Drawn in Wave Window



Signal	Value
/test_counter/clk	0
/test_counter/reset	0
/test_counter/count	00011110
/test_counter/dut/count	00011110
/test_counter/dut/clk	St0
/test_counter/dut/reset	St0
/test_counter/dut/increment/increment	00011110
/test_counter/dut/increment/val	00011101
/test_counter/dut/increment/i	0010
/test_counter/dut/increment/carry	0

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

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

d. Click the Break icon. The simulation stops running.

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 lines with red line numbers.

1. Open *counter.v* in the Source window.

   a. Select the **Files** tab in the Main window Workspace.

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

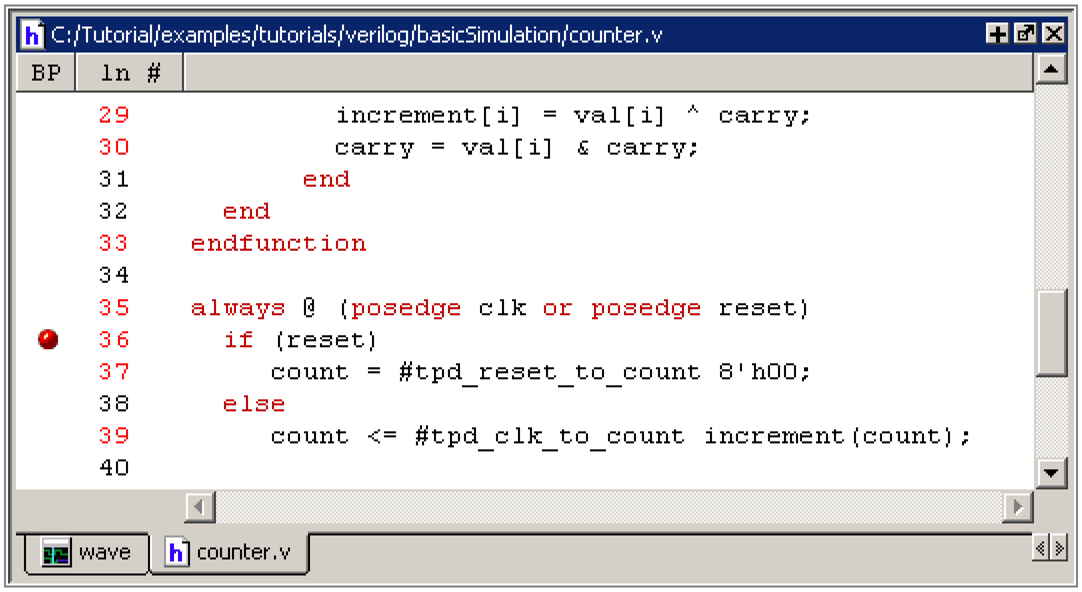
   c. Double-click *counter.v* to open it in the Source window.

2. Set a breakpoint on line 36 of *counter.v* (or, line 36 of *counter.vhd* if you are simulating the VHDL files).

   a. Scroll to line 36 and click in the BP (breakpoint) column next to the line number.

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

Figure 3-10. Setting Breakpoint in Source Window


C:/Tutorial/examples/tutorials/verilog/basicSimulation/counter.v
wave
counter.v

```
BP  ln #
     29             increment[i] = val[i] ^ carry;
     30             carry = val[i] & carry;
     31           end
     32       end
     33     endfunction
     34
     35     always @ (posedge clk or posedge reset)
●    36       if (reset)
     37          count = #tpd_reset_to_count 8'h00;
     38       else
     39          count <= #tpd_clk_to_count increment(count);
     40
```

3. Disable, enable, and delete the breakpoint.

   a. Click the red ball to disable the breakpoint. It will become a black ball.

b. Click the black ball again to re-enable the breakpoint. It will become a red ball.

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

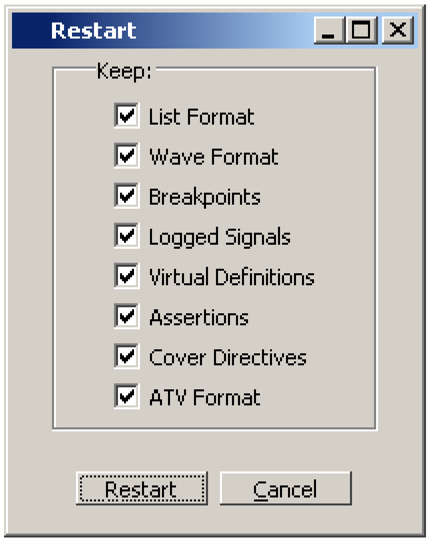
d. Click in the BP column next to line number 36 again to re-create the breakpoint.

4. Restart the simulation.

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

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

Figure 3-11. Setting Restart Functions



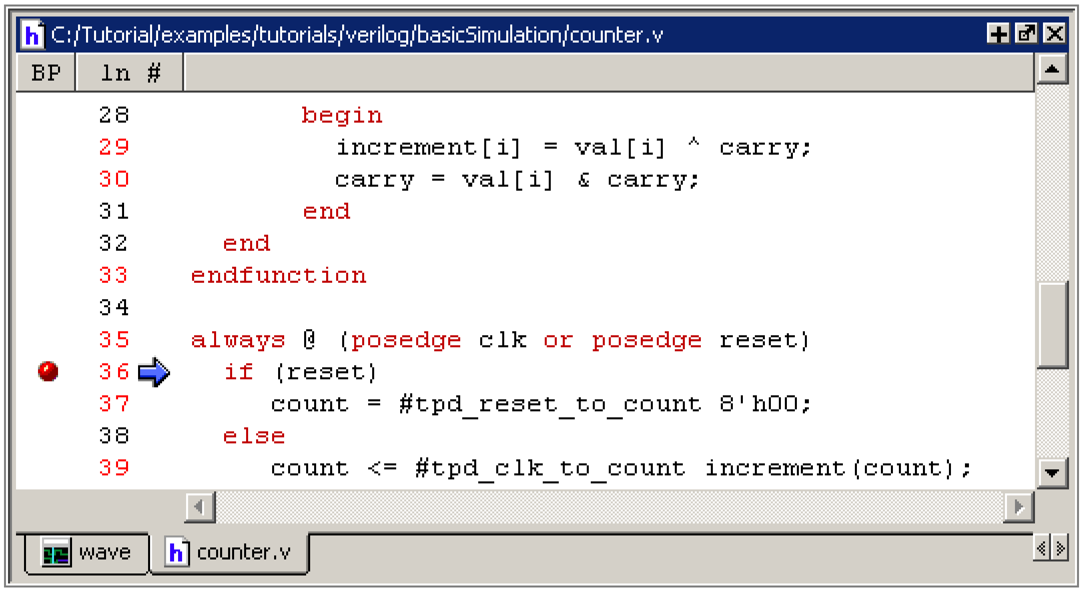
- [✓] List Format
- [✓] Wave Format
- [✓] Breakpoints
- [✓] Logged Signals
- [✓] Virtual Definitions
- [✓] Assertions
- [✓] Cover Directives
- [✓] ATV Format

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

c. Click the Run -All icon.

The simulation runs until the breakpoint is hit. When the simulation hits the breakpoint, it stops running, highlights the line with a blue arrow in the Source view (Figure 3-12), and issues a Break message in the Transcript pane.

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

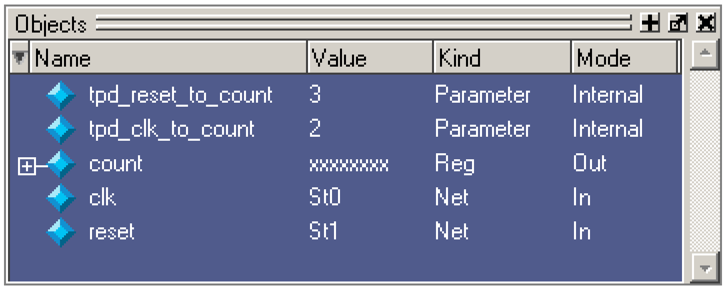

C:/Tutorial/examples/tutorials/verilog/basicSimulation/counter.v
BP
ln #
wave
counter.v

```
28          begin
29            increment[i] = val[i] ^ carry;
30            carry = val[i] & carry;
31          end
32      end
33    endfunction
34
35    always @ (posedge clk or posedge reset)
36      if (reset)
37         count = #tpd_reset_to_count 8'h00;
38      else
39         count <= #tpd_clk_to_count increment(count);
```

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).

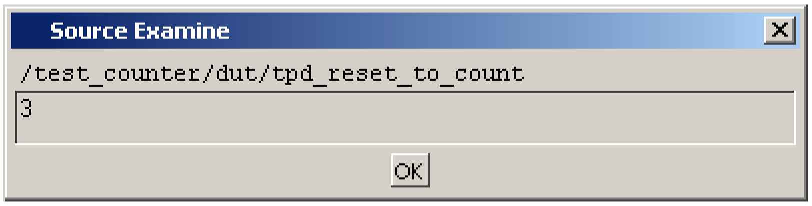
Figure 3-13. Values Shown in Objects Window



Name	Value	Kind	Mode
tpd_reset_to_count	3	Parameter	Internal
tpd_clk_to_count	2	Parameter	Internal
count	xxxxxxxx	Reg	Out
clk	St0	Net	In
reset	St1	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
- 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 3-14)

Figure 3-14. Parameter Name and Value in Source Examine Window



```
/test_counter/dut/tpd_reset_to_count
3
```

- use the **examine** command at the VSIM> prompt to output a variable value to the Main window Transcript (i.e., `examine count`)

5. Try out the step commands.

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

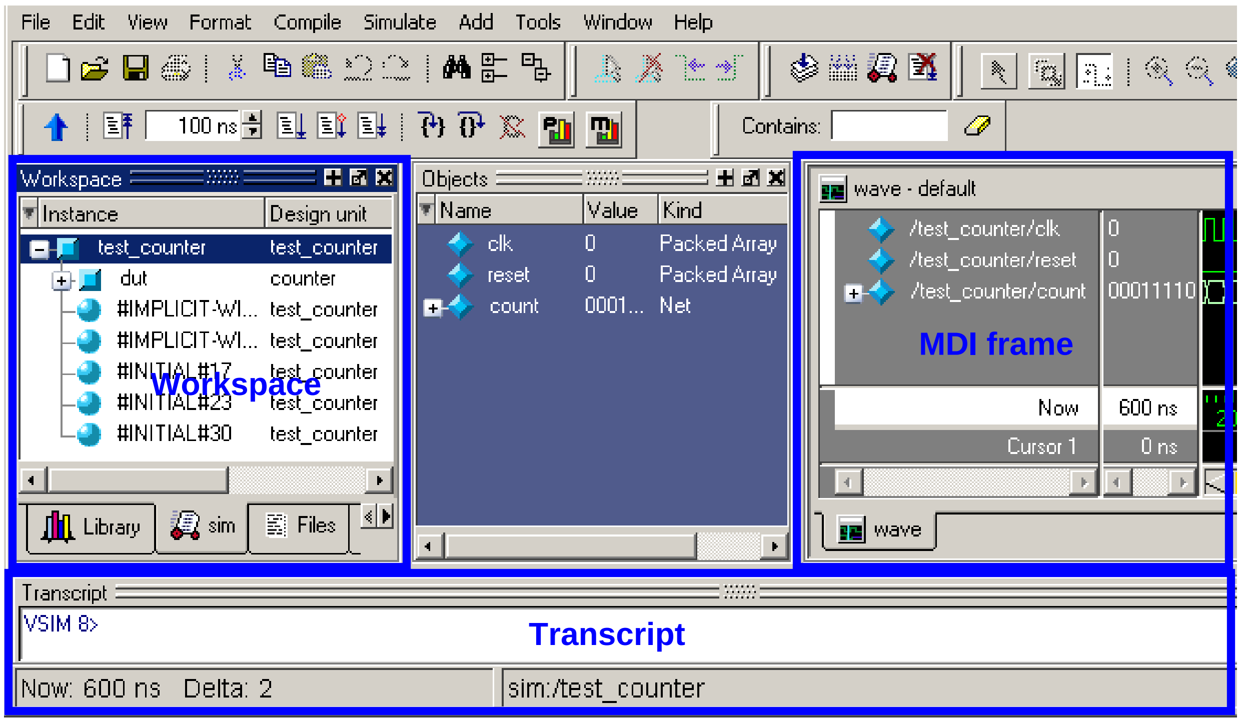
   This single-steps the debugger.

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

Navigating the Interface

The Main window is composed of a number of "panes" and sub-windows that display various types of information about your design, simulation, or debugging session. You can also access other tools from the Main window that display in stand-alone windows (e.g., the Dataflow window).

Figure 3-15. The Main Window



The following table describes some of the key elements of the Main window.

Table 3-1. The Main Window

Window/pane	**Description**
Workspace	This pane comprises multiple tabs that contain various sorts of information about the current project or design. Once a design is loaded, additional tabs will appear. Refer to the section Workspace in the User's Manual for more information.
Transcript	The Transcript pane provides a command-line interface and serves as an activity log including status and error messages. Refer to the section Transcript Window in the User's Manual for more information.

Table 3-1. The Main Window

Window/pane	Description
MDI frame	The Multiple Document Interface (MDI) frame holds windows for which there can be multiple instances. These include Source editor windows, Wave windows, and Memory content windows. Refer to the section Multiple Document Interface (MDI) Frame in the User's Manual for more information.

Here are a few important points to keep in mind about the ModelSim interface:

- Windows/panes can be resized, moved, zoomed, undocked, etc. and the changes are persistent.

  You have a number of options for re-sizing, re-positioning, undocking/redocking, and generally modifying the physical characteristics of windows and panes. When you exit ModelSim, the current layout is saved so that it appears the same the next time you invoke the tool. Refer to the Main Window section in the User's Manual for more information.

- Menus are context sensitive.

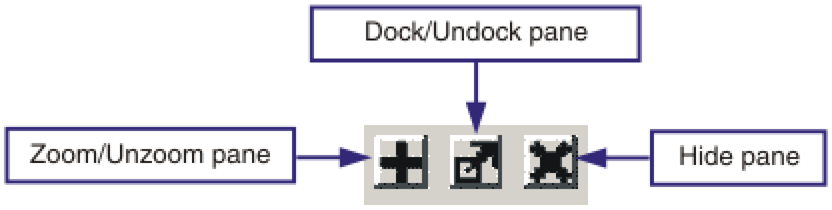
  The menu items that are available and how certain menu items behave depend on which pane or window is active. For example, if the *sim* tab in the Workspace is active and you choose Edit from the menu bar, the Clear command is disabled. However, if you click in the Transcript pane and choose Edit, the Clear command is enabled. The active pane is denoted by a blue title bar.

Let us try a few things.

1. Zoom and undock panes.

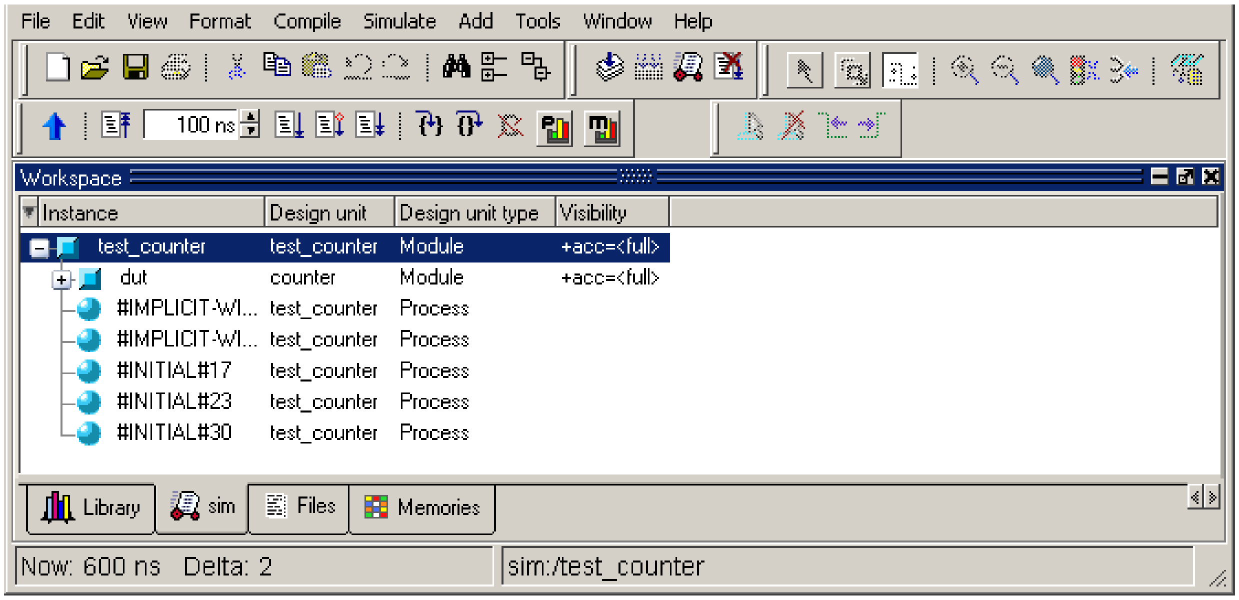
   a. Click the Zoom/Unzoom icon in the upper right corner of the Workspace pane (Figure 3-16).

Figure 3-16. Window/Pane Control Icons



The pane fills the entire Main window (Figure 3-17).

Figure 3-17. zooming in on Workspace Pane


File Edit View Format Compile Simulate Add Tools Window Help
100 ns
Workspace
Library
sim
Files
Memories

Instance	Design unit	Design unit type	Visibility
test_counter	test_counter	Module	+acc=<full>
dut	counter	Module	+acc=<full>
#IMPLICIT-WI...	test_counter	Process	
#IMPLICIT-WI...	test_counter	Process	
#INITIAL#17	test_counter	Process	
#INITIAL#23	test_counter	Process	
#INITIAL#30	test_counter	Process	

Now: 600 ns Delta: 2 sim:/test_counter

b. Click the Unzoom icon in the Workspace.

c. Click the Undock icon in the upper right corner of the Transcript pane.

The Transcript becomes a stand-alone window.

d. Click the Dock icon on the Transcript.

e. Click the Hide pane icon in the Workspace.

f. Select **View** > **Workspace** from the menus to re-open the Workspace.

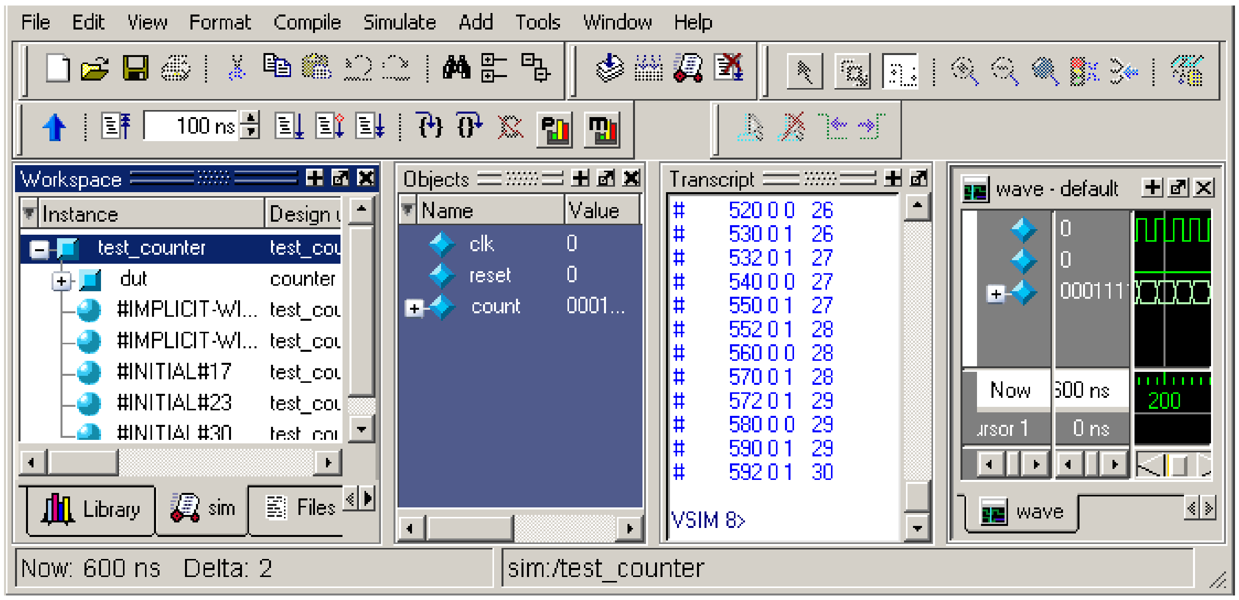
2. Move and resize panes.

a. Hover your mouse pointer in the center of the Transcript title bar, where the two parallel lines are interrupted by 3 lines of small dots. This is the handle for the pane. When the cursor is over the pane handle it becomes a four-headed arrow.

b. Click and drag the Transcript up and to the right until you see a gray outline on the right-hand side of the MDI frame.

When you let go of the mouse button, the Transcript is moved and the MDI frame and Workspace panes shift to the left (Figure 3-18).

Figure 3-18. Panes Rearranged in Main Window



```
#     520 0 0  26
#     530 0 1  26
#     532 0 1  27
#     540 0 0  27
#     550 0 1  27
#     552 0 1  28
#     560 0 0  28
#     570 0 1  28
#     572 0 1  29
#     580 0 0  29
#     590 0 1  29
#     592 0 1  30

VSIM 8>
```

Now: 600 ns   Delta: 2   sim:/test_counter

c. Select **Layout** > **Reset**.

The layout returns to its original setting.

Tip: Moving panes can get confusing, and you may not always obtain the results you expect. Practice moving a pane around, watching the gray outline to see what happens when you drop it in various places. Your layout will be saved when you exit ModelSim and will reappear when you next open ModelSim. (It's a good idea to close all panes in the MDI frame at the end of each lesson in this tutorial so only files relevant to each lesson will be displayed.)

As you practice, notice that the MDI frame cannot be moved in the same manner as the panes. It does not have a handle in its header bar.

Selecting **Layout** > **Reset** is the easiest way to rectify an undesired layout.

d. Hover your mouse pointer on the border between two panes so it becomes a double-headed arrow. ⟷

e. Click-and-drag left and right or up and down to resize the pane.

f. Select **Layout** > **Reset**.

3. Observe context sensitivity of menu commands.

a. Click anywhere in the Workspace.

b. Select the Edit menu and notice that the **Clear** command is disabled.

c. Click in the Transcript and select **Edit > Clear**.

   This command applies to the Transcript pane but not the Workspace pane.

d. Click on a design object in the sim tab of the Workspace and select **File** > **Open**.

e. Notice that the Open dialog filters to show Log files (**.wlf*).

f. Now click on a filename in the Files tab of the Workspace and select **File** > **Open**.

   Notice that the Open dialog filters to show HDL file types instead.

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

Introduction

In this lesson you will practice creating a project.

At a minimum, projects contain a work library and a session state that is stored in a *.mpf* file. A project may also consist of:

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

Design Files for this Lesson

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

Verilog – *<install_dir>/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.

Related Reading

User's Manual Chapter: Projects.

Create a New Project

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).

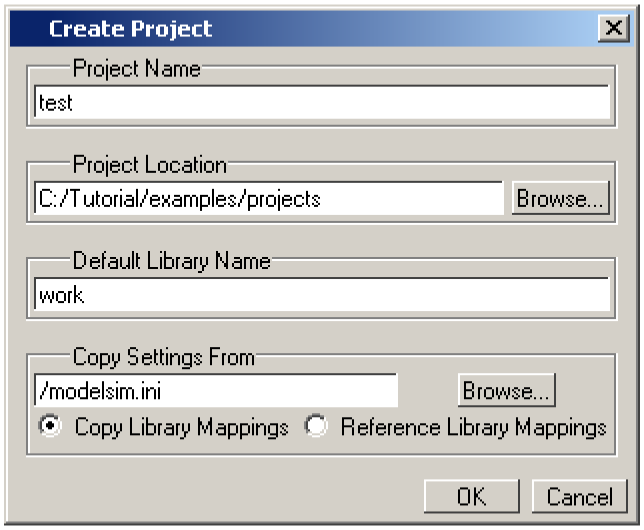
   Verilog: Copy *counter.v* and *tcounter.v* files from */<install_dir>/examples/tutorials/verilog/projects* to the new directory.

   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 not, 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 where you can enter a Project Name, Project Location (i.e., 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 - Project Lab

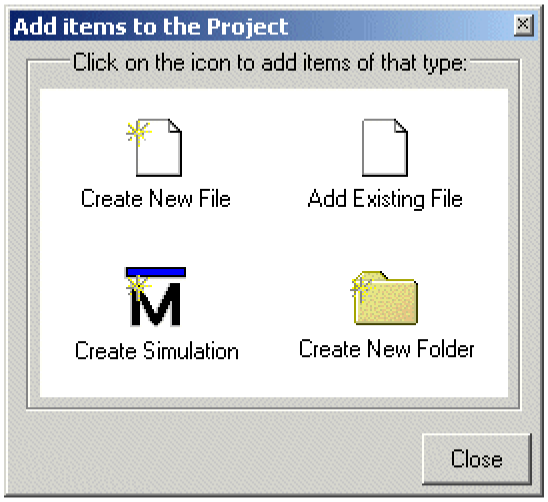


Project Name: test
Project Location: C:/Tutorial/examples/projects
Default Library Name: work
Copy Settings From: /modelsim.ini
[✓ Copy Library Mappings] Reference Library Mappings

Add Objects to the Project

Once you click OK to accept the new project settings, you will see a blank Project tab in the Workspace area of the Main window and the Add items to the Project dialog will appear (Figure 4-2). From this dialog you can create a new design file, add an existing file, add a folder for organization purposes, or create a simulation configuration (discussed below).

Figure 4-2. Adding New Items to a Project

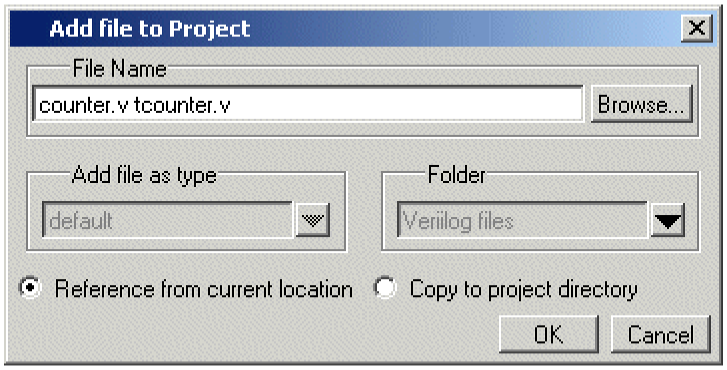


1. Add two existing files.

   a. Click **Add Existing File**.

   This opens the Add file to Project dialog (Figure 4-3). This dialog 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



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 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**.

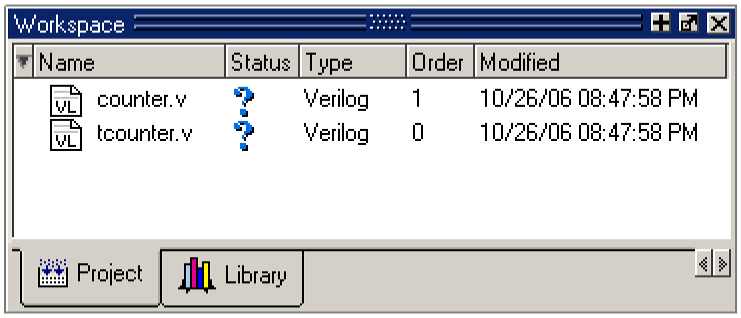
   This closes the “Select files to add to project” dialog and displays the selected files in the “Add file to Project” dialog (Figure 4-3).

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

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

You should now see two files listed in the Project tab of the Workspace pane (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 (e.g., Verilog or VHDL), compilation order, and modified date.

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



Name	Status	Type	Order	Modified
counter.v	?	Verilog	1	10/26/06 08:47:58 PM
tcounter.v	?	Verilog	0	10/26/06 08:47:58 PM

Changing Compile Order (VHDL)

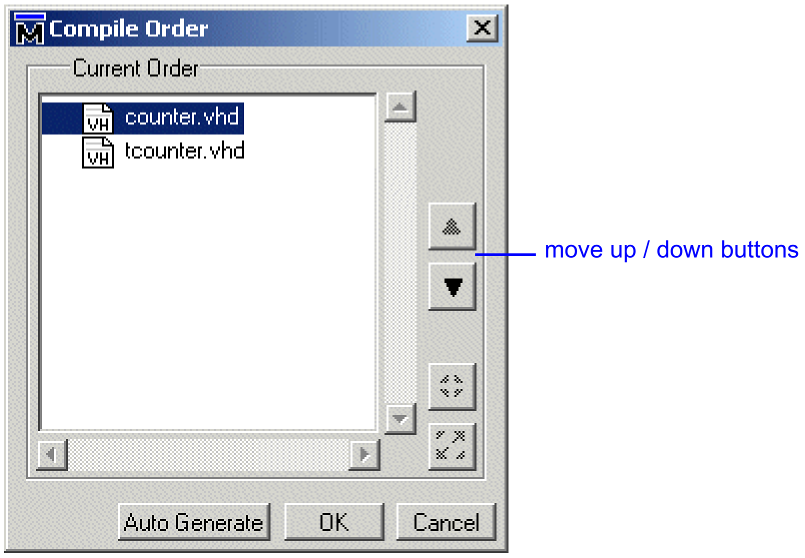
By default ModelSim performs default binding of VHDL designs when you load the design with vsim. However, you can elect to perform default binding at compile time. (For details, refer to the section Default Binding in the User's Manual.) If you elect to do default binding at compile, then the compile order is important. Follow these steps to change compilation order within a project.

1. Change the compile order.

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

This opens the Compile Order dialog box (Figure 4-5).

Figure 4-5. Compile Order Dialog



b. Click the **Auto Generate** button.

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

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

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

Compile the Design

1. Compile the files.

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

   ModelSim compiles both files and changes the symbol in the Status column to a 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 pane.

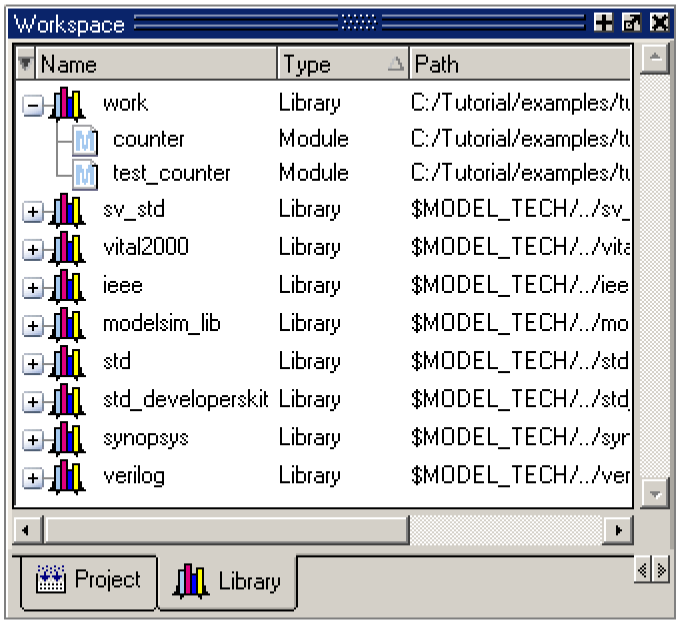
2. View the design units.

   a. Click the **Library** tab in the workspace (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 Tab with Expanded Library



Name	Type	Path
work	Library	C:/Tutorial/examples/tu
counter	Module	C:/Tutorial/examples/tu
test_counter	Module	C:/Tutorial/examples/tu
sv_std	Library	$MODEL_TECH/../sv_
vital2000	Library	$MODEL_TECH/../vita
ieee	Library	$MODEL_TECH/../iee
modelsim_lib	Library	$MODEL_TECH/../mo
std	Library	$MODEL_TECH/../std
std_developerskit	Library	$MODEL_TECH/../std
synopsys	Library	$MODEL_TECH/../syn
verilog	Library	$MODEL_TECH/../ver

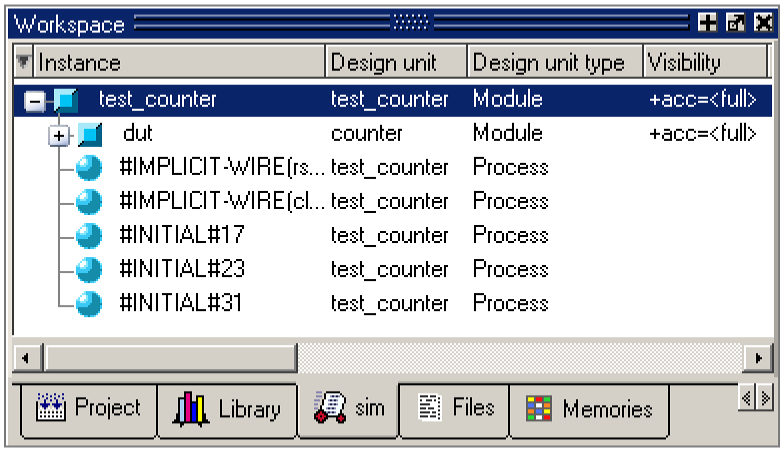
Load the Design

1. Load the *test_counter* design unit.

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

   You should see 3 new tabs in the Main window Workspace. The *sim* tab displays the structure of the *test_counter* design unit (Figure 4-7). The *Files* tab contains information about the underlying source files. The *Memories* tab lists all memories in the design.

Figure 4-7. Structure Tab for a Loaded Design



Instance	Design unit	Design unit type	Visibility
test_counter	test_counter	Module	+acc=<full>
dut	counter	Module	+acc=<full>
#IMPLICIT-WIRE(rs...	test_counter	Process	
#IMPLICIT-WIRE(cl...	test_counter	Process	
#INITIAL#17	test_counter	Process	
#INITIAL#23	test_counter	Process	
#INITIAL#31	test_counter	Process	

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'll 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.

Add Folders

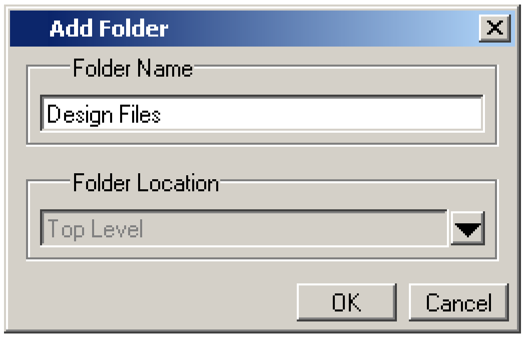
As shown previously in Figure 4-2, the Add items to the Project dialog has an option for adding folders. If you have already closed that dialog, you can use a menu command to add a folder.

1. Add a new folder.

   a. Right-click inside the Projects tab of the Workspace and select **Add to Project** > **Folder**.

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

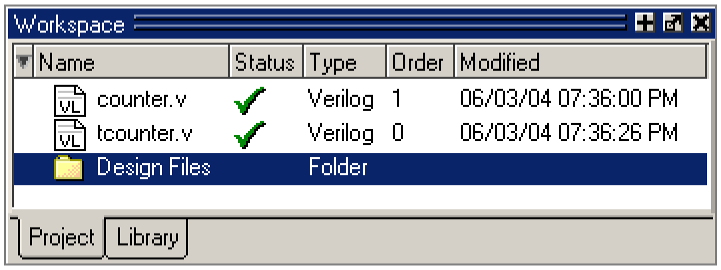
Figure 4-8. Adding New Folder to Project


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

   c. Click **OK**.

   d. Select the Project tab to see the new folder (Figure 4-9).

Figure 4-9. A Folder Within a Project



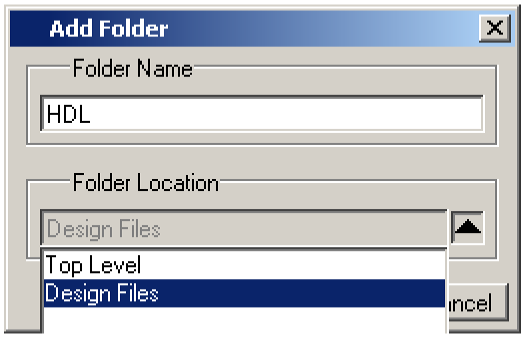
Name	Status	Type	Order	Modified
counter.v	✓	Verilog	1	06/03/04 07:36:00 PM
tcounter.v	✓	Verilog	0	06/03/04 07:36:26 PM
Design Files		Folder		

2. Add a sub-folder.

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

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

Figure 4-10. Creating Subfolder

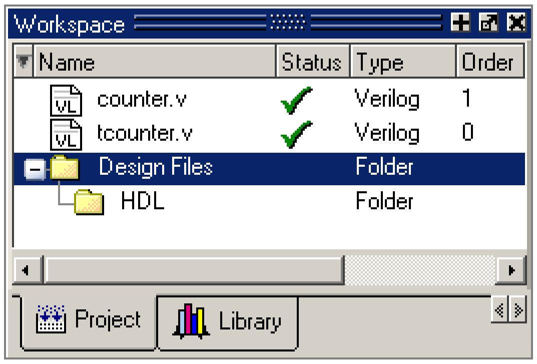


   c. Click the **Folder Location** drop-down arrow and select *Design Files*.

   d. Click **OK**.

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

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



Name	Status	Type	Order
counter.v	✓	Verilog	1
tcounter.v	✓	Verilog	0
Design Files		Folder	
HDL		Folder	

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

Moving Files to Folders

If you don’t place files into a folder when you first add the files to the project, you can move them into a folder using the properties dialog.

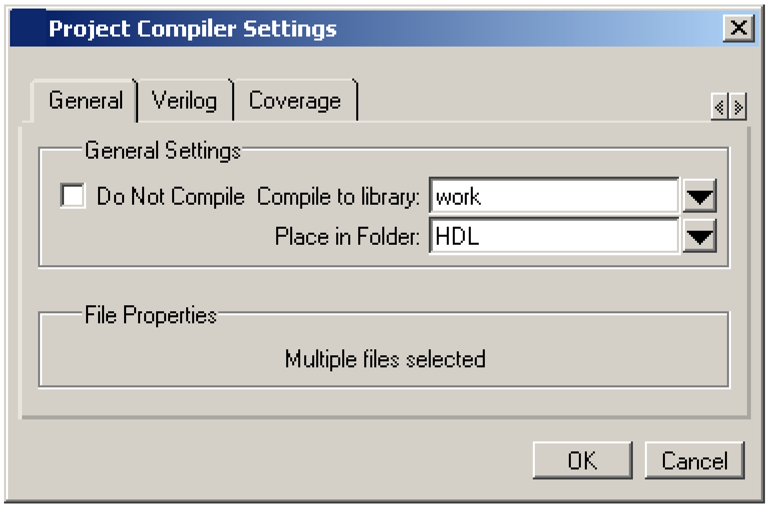
1. Move *tcounter.v* and *counter.v* to the *HDL* folder.

   a. Select both *counter.v* and *tcounter.v* in the Project tab of the Workspace.

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

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

Figure 4-12. Changing File Location via the Project Compiler Settings Dialog



Compile to library: work
Place in Folder: HDL
File Properties
Multiple files selected

   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.

Simulation Configurations

A Simulation Configuration associates a design unit(s) and its simulation options. For example, let’s 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.

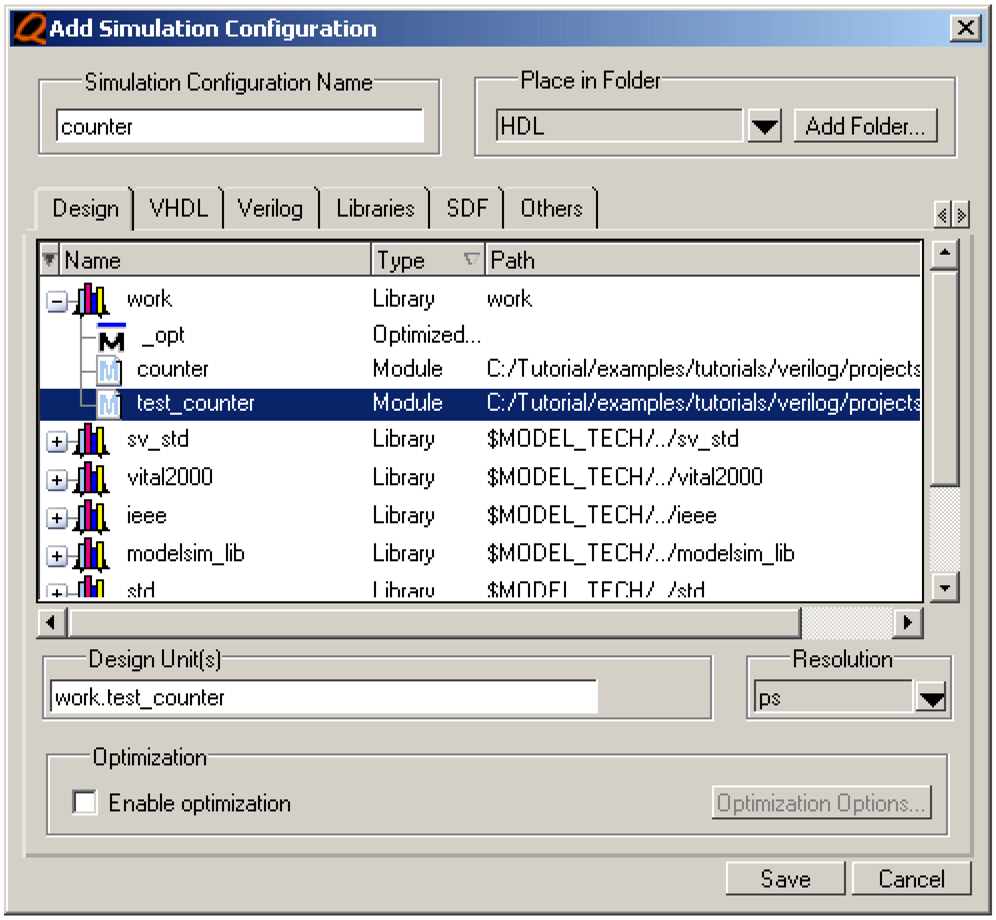
The configuration is then listed in the Project tab and you can double-click it to load *tcounter.v* along with its options.

1. Create a new Simulation Configuration.

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

   This opens the Add Simulation Configuration dialog (Figure 4-13). The tabs in this dialog present a myriad of 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



Simulation Configuration Name: counter

Place in Folder: HDL

Name	Type	Path
work	Library	work
_opt	Optimized...	
counter	Module	C:/Tutorial/examples/tutorials/verilog/projects
test_counter	Module	C:/Tutorial/examples/tutorials/verilog/projects
sv_std	Library	$MODEL_TECH/../sv_std
vital2000	Library	$MODEL_TECH/../vital2000
ieee	Library	$MODEL_TECH/../ieee
modelsim_lib	Library	$MODEL_TECH/../modelsim_lib
std	Library	$MODEL_TECH/../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*.

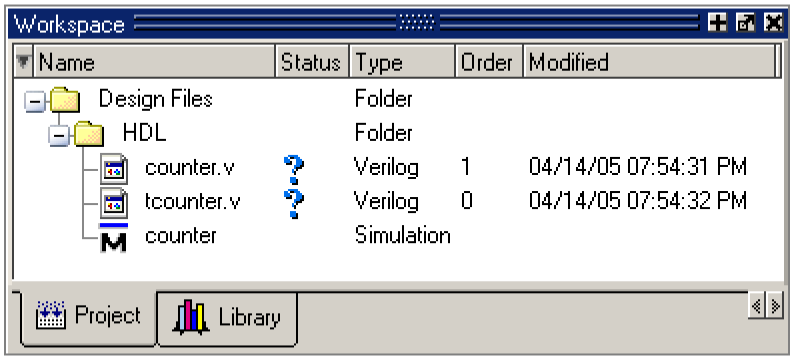
   e. Click the **Resolution** drop-down and select *ps*.

f. For Verilog, click the Verilog tab and check **Enable hazard checking (-hazards)**.

g. Click **Save**.

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

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



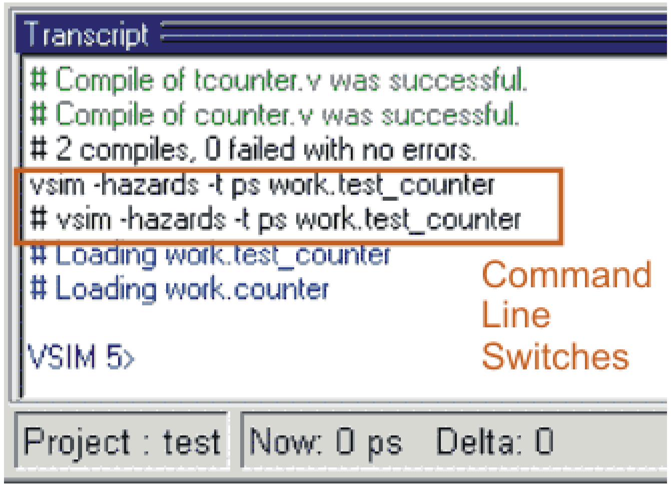
Name	Status	Type	Order	Modified
Design Files		Folder		
HDL		Folder		
counter.v	?	Verilog	1	04/14/05 07:54:31 PM
tcounter.v	?	Verilog	0	04/14/05 07:54:32 PM
counter		Simulation		

2. Load the Simulation Configuration.

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

In the Transcript pane of the Main window, the **vsim** (the ModelSim simulator) invocation shows the **-hazards** and **-t ps** switches (Figure 4-15). These are the command-line equivalents of the options you specified in the Simulate dialog.

Figure 4-15. Transcript Shows Options for Simulation Configurations



```
# Compile of tcounter.v was successful.
# Compile of counter.v was successful.
# 2 compiles, 0 failed with no errors.
vsim -hazards -t ps work.test_counter
# vsim -hazards -t ps work.test_counter
# Loading work.test_counter
# Loading work.counter

VSIM 5>
```

```
Project : test    Now: 0 ps    Delta: 0
```

Lesson Wrap-Up

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

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

2. Select the Project tab in the Main window Workspace.
3. Right-click in this tab to open a popup menu and select **Close Project**.
4. Click **OK**.

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

Chapter 5
Working With Multiple Libraries

Introduction

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 testbench into it. Finally, you will link to the library containing the counter and then run the simulation.

Design Files for this Lesson

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

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

Related Reading

User's Manual Chapter: Design Libraries.

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. See Permanently Mapping VHDL Resource Libraries.

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 testbench.

Create a new directory called *testbench* that will hold the testbench 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 appears, click **Close**.

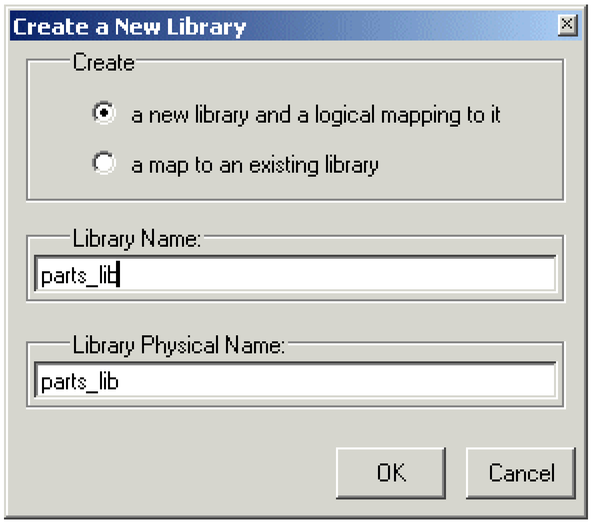
   b. Select **File > Change Directory** and change to the *resource_library* directory you created in step 1.

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



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

Library Name: parts_lib

Library Physical Name: parts_lib

The Library Physical Name field is filled out automatically.

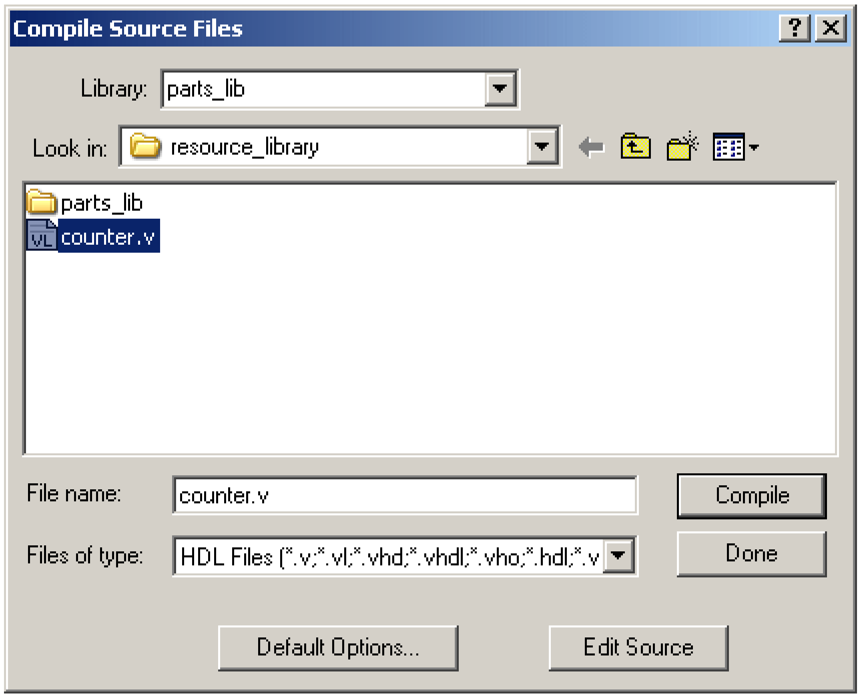
Once you click OK, ModelSim creates a directory for the library, lists it in the Library tab of the Workspace, and modifies the *modelsim.ini* file to record this new library for the future.

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



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 testbench.

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 testbench to the project.

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

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

   c. Click **Open**.

   d. Click **OK**.

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

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

3. Compile the testbench.

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

Linking to the Resource Library

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

ModelSim responds differently for Verilog and VHDL in this situation.

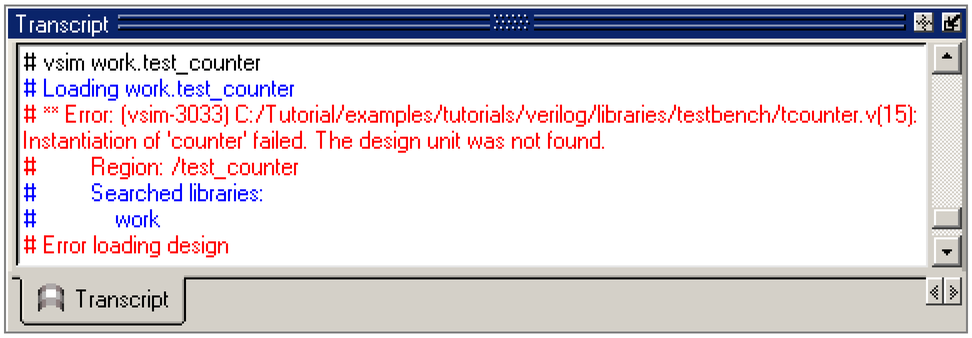
Verilog

1. Simulate a Verilog design with a missing resource library.

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

   The Main window Transcript reports an error (Figure 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 Main Window



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

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

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

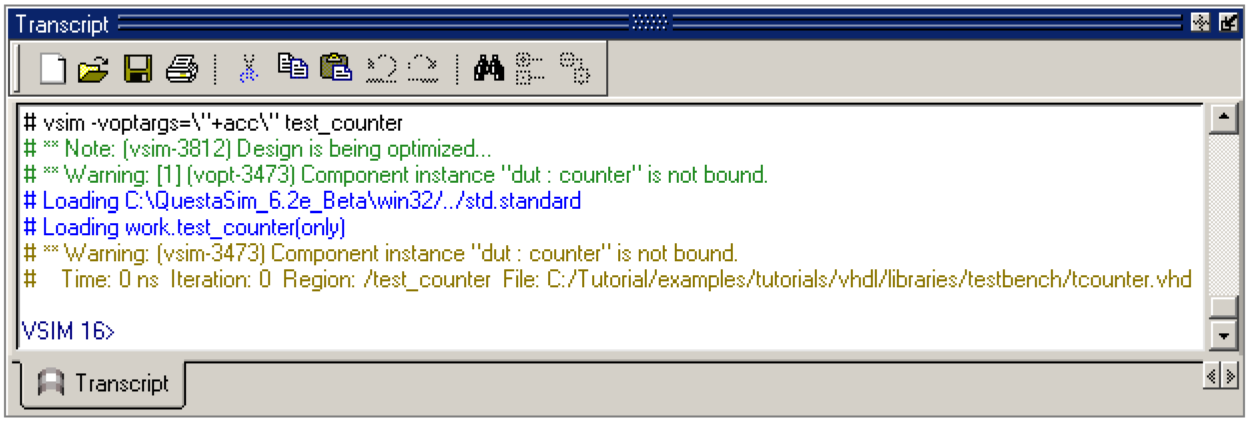
VHDL

1. Simulate a VHDL design with a missing resource library.

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

   The Main window Transcript reports a warning (Figure 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



```
# vsim -voptargs=\"+acc\" test_counter
# ** Note: (vsim-3812) Design is being optimized...
# ** Warning: [1] (vopt-3473) Component instance "dut : counter" is not bound.
# Loading C:\QuestaSim_6.2e_Beta\win32/../std.standard
# Loading work.test_counter(only)
# ** Warning: (vsim-3473) Component instance "dut : counter" is not bound.
#    Time: 0 ns  Iteration: 0  Region: /test_counter  File: C:/Tutorial/examples/tutorials/vhdl/libraries/testbench/tcounter.vhd

VSIM 16>
```

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.

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

Linking in Verilog

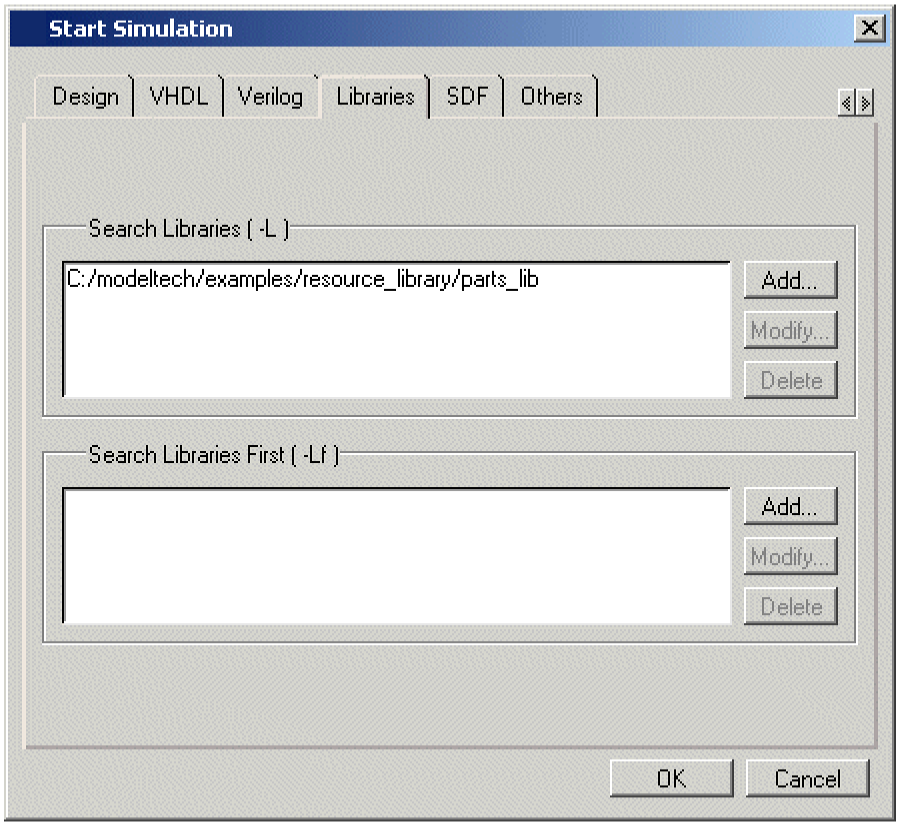
Linking in Verilog requires that you specify a "search library" when you invoke the simulator.

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

      The dialog 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



C:/modeltech/examples/resource_library/parts_lib

Linking in VHDL

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

1. Create a logical mapping to *parts_lib*.

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

   b. In the Create a New Library dialog, select **a map to an existing library**.

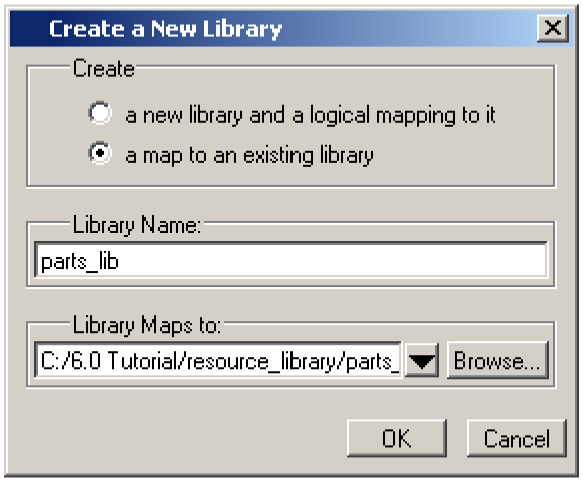
   c. Type **parts_lib** in the Library Name field.

   d. Click Browse to open the Select Library dialog and browse to *parts_lib* in the *resource_library* directory you created earlier in the lesson.

   e. Click OK to select the library and close the Select Library dialog.

   f. The Create a New Library dialog should look similar to the one shown in Figure 5-6. Click **OK** to close the dialog.

Figure 5-6. Mapping to the parts_lib Library



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

Library Name: parts_lib

Library Maps to: C:/6.0 Tutorial/resource_library/parts_

2. Add LIBRARY and USE statements to *tcounter.vhd*.

   a. In the Library tab of the Main window, click the '+' icon next to the *work* library.

   b. Right-click *test_counter* in the work library and select **Edit**.

   c. This opens the file in the Source window.

   d. Right-click in the Source window and uncheck Read Only.

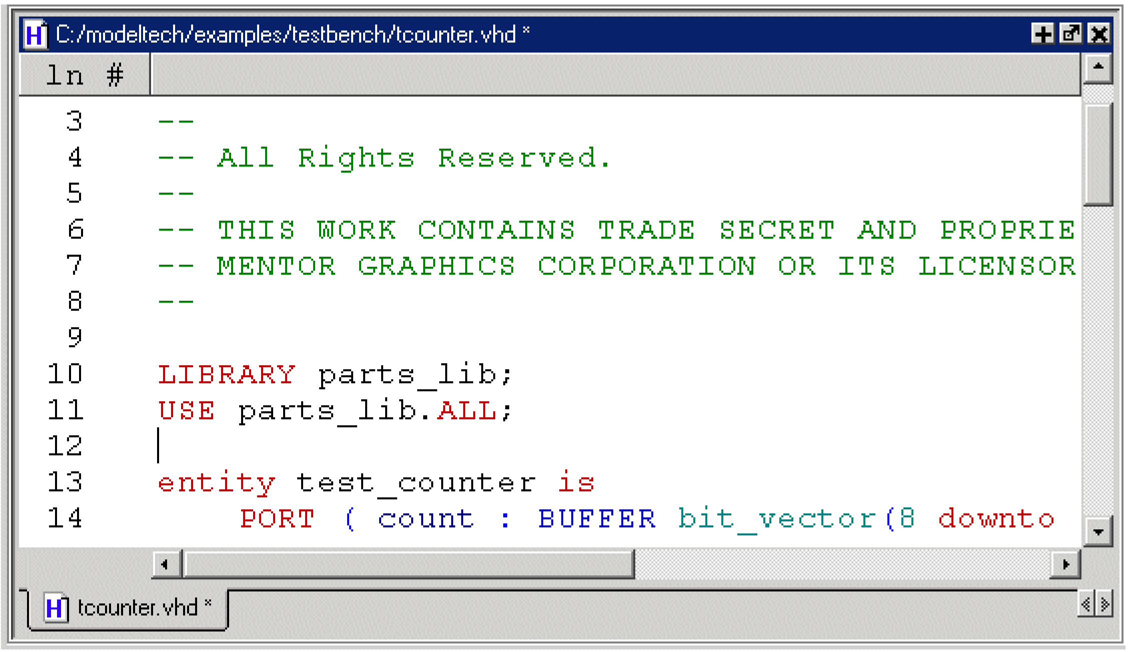
   e. Add these two lines to the top of the file:

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

   The testbench source code should now look similar to that shown in Figure 5-7.

   f. Select **File** > **Save**.

Figure 5-7. Adding LIBRARY and USE Statements to the Testbench


C:/modeltech/examples/testbench/tcounter.vhd *

```
--
-- All Rights Reserved.
--
-- THIS WORK CONTAINS TRADE SECRET AND PROPRIE
-- MENTOR GRAPHICS CORPORATION OR ITS LICENSOR
--

LIBRARY parts_lib;
USE parts_lib.ALL;

entity test_counter is
    PORT ( count : BUFFER bit_vector(8 downto
```

3. Recompile and simulate.

   a. In the Project tab of the Workspace, right-click *tcounter. vhd* and select **Compile** > **Compile Selected**.

   b. In the Library tab, double-click *test_counter to load the design*.

   c. The design loads without errors.

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 won't actually practice it in this tutorial, here are the steps for editing the file:

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

   ```
   parts_lib = C:/libraries/parts_lib
   ```

5. Save the file.

6. Change the file attributes so the file is "read-only" again.

Lesson Wrap-Up

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

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

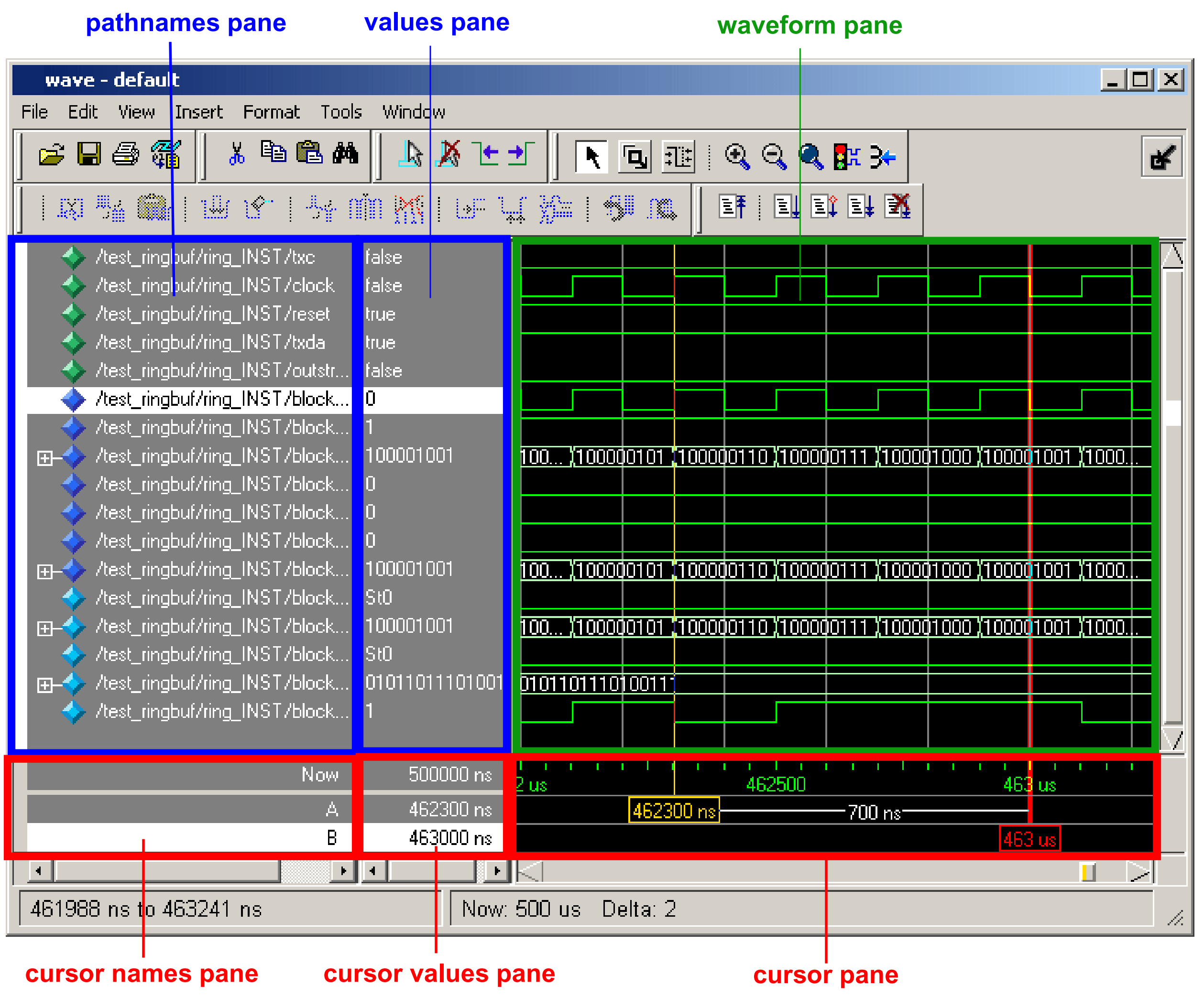
Chapter 6 Analyzing Waveforms

Introduction

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

The Wave window is divided into a number of window panes (Figure 6-1). All window panes in the Wave window can be resized by clicking and dragging the bar between any two panes.

Figure 6-1. Panes of the Wave Window



```
/test_ringbuf/ring_INST/txc          false
/test_ringbuf/ring_INST/clock        false
/test_ringbuf/ring_INST/reset        true
/test_ringbuf/ring_INST/txda         true
/test_ringbuf/ring_INST/outstr...    false
/test_ringbuf/ring_INST/block...     0
/test_ringbuf/ring_INST/block...     1
/test_ringbuf/ring_INST/block...     100001001
/test_ringbuf/ring_INST/block...     0
/test_ringbuf/ring_INST/block...     0
/test_ringbuf/ring_INST/block...     0
/test_ringbuf/ring_INST/block...     100001001
/test_ringbuf/ring_INST/block...     St0
/test_ringbuf/ring_INST/block...     100001001
/test_ringbuf/ring_INST/block...     St0
/test_ringbuf/ring_INST/block...     01011011101001
/test_ringbuf/ring_INST/block...     1

100... 100000101 100000110 100000111 100001000 100001001 1000...
0101101110100111

Now   500000 ns
A     462300 ns
B     463000 ns

2 us   462500   463 us
462300 ns   700 ns   463 us

461988 ns to 463241 ns      Now: 500 us  Delta: 2
```

Related Reading

User’s Manual sections: Wave Window and WLF Files (Datasets) and Virtuals.

Loading a Design

For the examples in this lesson, we have used the design simulated in Basic Simulation.

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

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

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

2. Load the design.

   a. Select **File** > **Change Directory** and open the directory you created in Lesson 2.

      The *work* library should already exist.

   b. Click the ’+’ icon next to the *work* library and double-click *test_counter*.

      ModelSim loads the design and adds *sim* and *Files* tabs to the Workspace.

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.

1. Add objects from the Objects pane.

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

      ModelSim adds several signals to the Wave window.

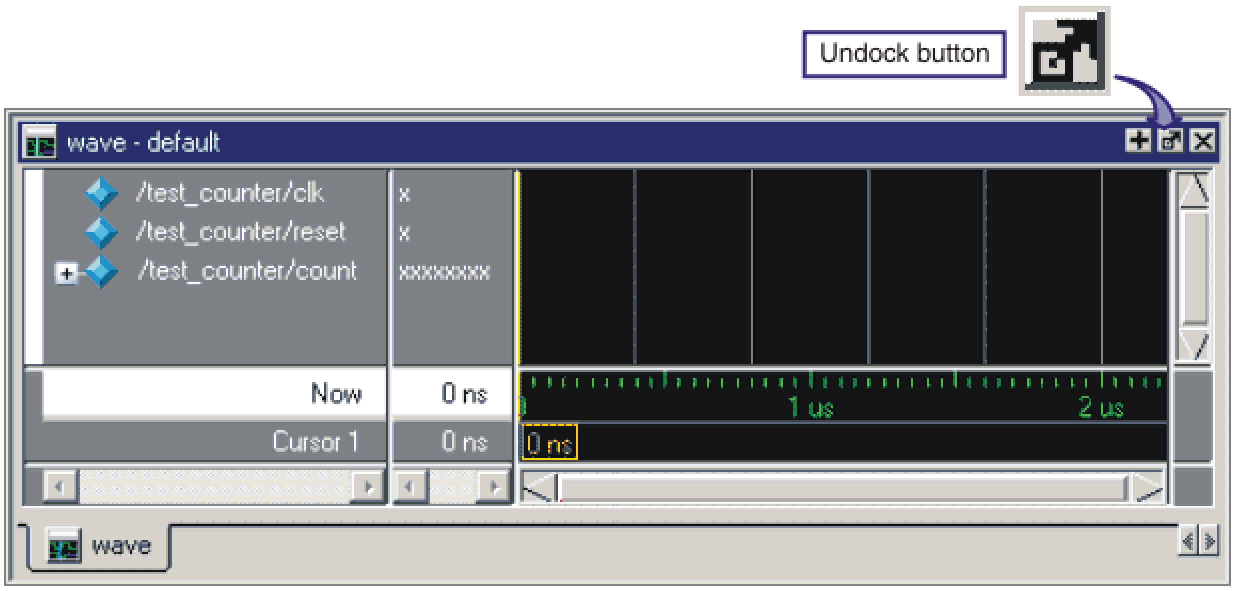
2. Undock the Wave window.

   By default ModelSim opens Wave windows as a tab in the MDI frame of the Main window. You can change the default via the Preferences dialog (**Tools** > **Edit Preferences**). Refer to the section Simulator GUI Preferences in the User’s Manual for more information.

   a. Click the undock button on the Wave pane (Figure 6-2).

      The Wave pane becomes a standalone, un-docked window. You may need to resize the window.

Figure 6-2. Undocking the Wave Window



3. Add objects using drag-and-drop.

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

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

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

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

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

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

4. Add objects using a command.

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

      ModelSim adds all objects from the current region.

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

Zooming the Waveform Display

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

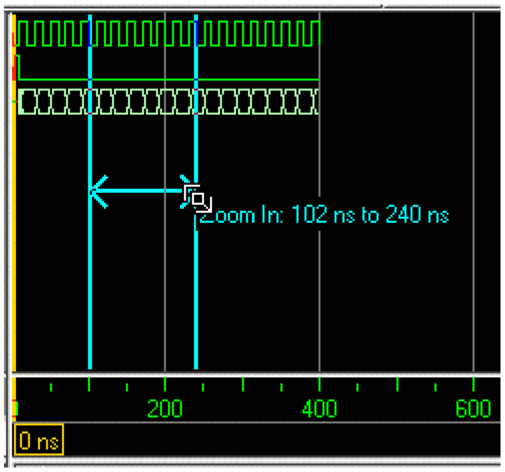
1. Zoom the display using various techniques.

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

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

You should see blue vertical lines and numbers defining an area to zoom in (Figure 6-3).

Figure 6-3. Zooming in with the Mouse Pointer



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

The waveform pane returns to the previous display range.

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

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

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

f. 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 anywhere in the waveform pane brings that cursor to the mouse location.

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

Working with a Single Cursor

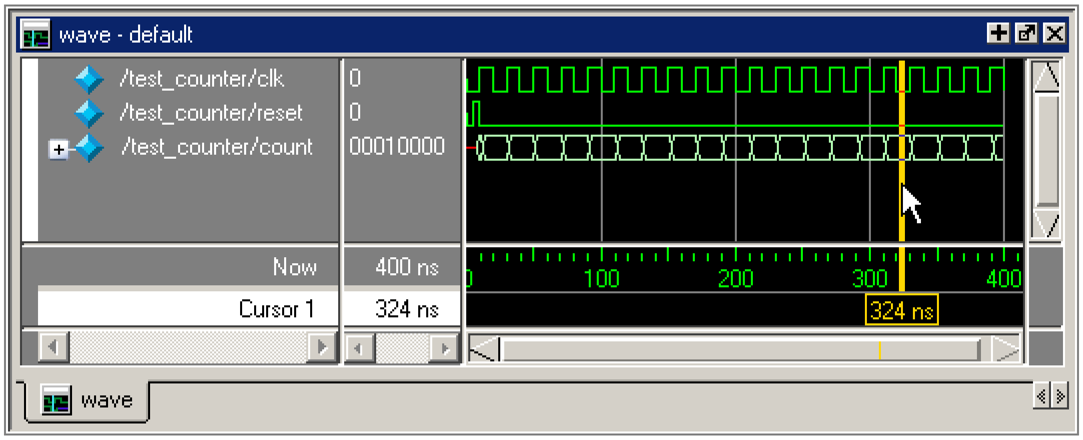
1. Position the cursor by clicking and dragging.

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

   b. Click anywhere in the waveform pane.

A cursor is inserted at the time where you clicked (Figure 6-4).

Figure 6-4. 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, drag the cursor to the right of a transition with the mouse positioned over a waveform.

The cursor "snaps" to the transition. Cursors "snap" to a waveform edge if you click or drag a cursor to within ten pixels of a waveform edge. You can set the snap distance in the Window Preferences dialog (select **Tools** > **Window Preferences**).

e. In the cursor pane, drag the cursor to the right of a transition (Figure 6-4).

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

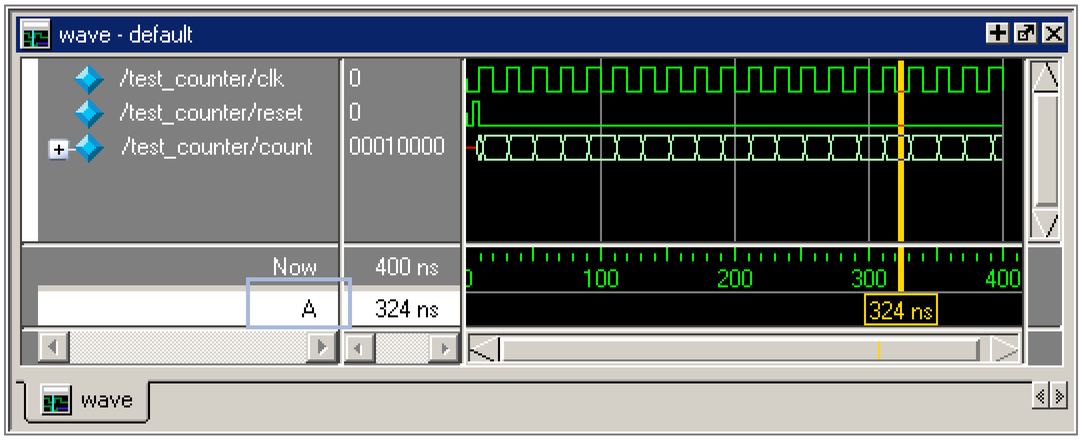
2. Rename the cursor.

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

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

The cursor name changes to "A" (Figure 6-5).

Figure 6-5. 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 currently selected signal.

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

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

Working with Multiple Cursors

1. Add a second cursor.

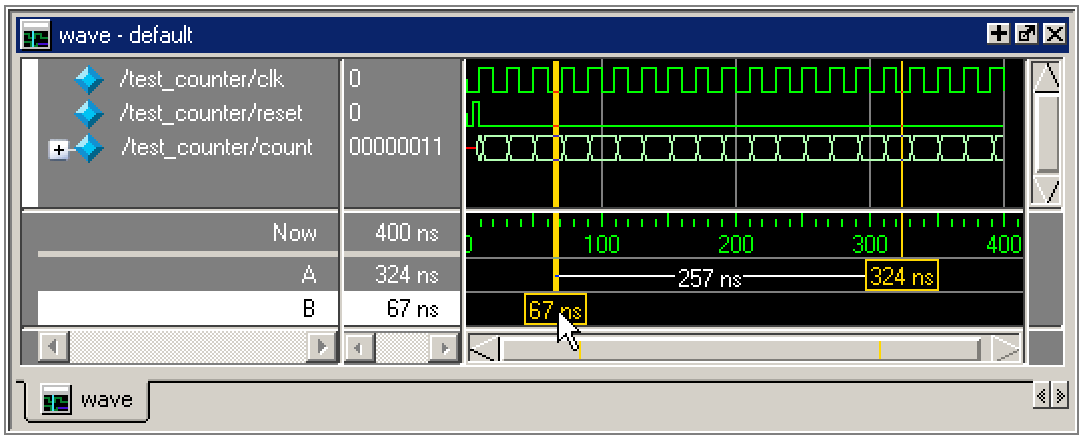
   a. Click the Add Cursor icon on the Wave window toolbar.

   b. Right-click the name of the new cursor and delete the text.

   c. Type **B** and press Enter.

   d. Drag cursor *B* and watch the interval measurement change dynamically (Figure 6-6).

Figure 6-6. Interval Measurement Between Two Cursors

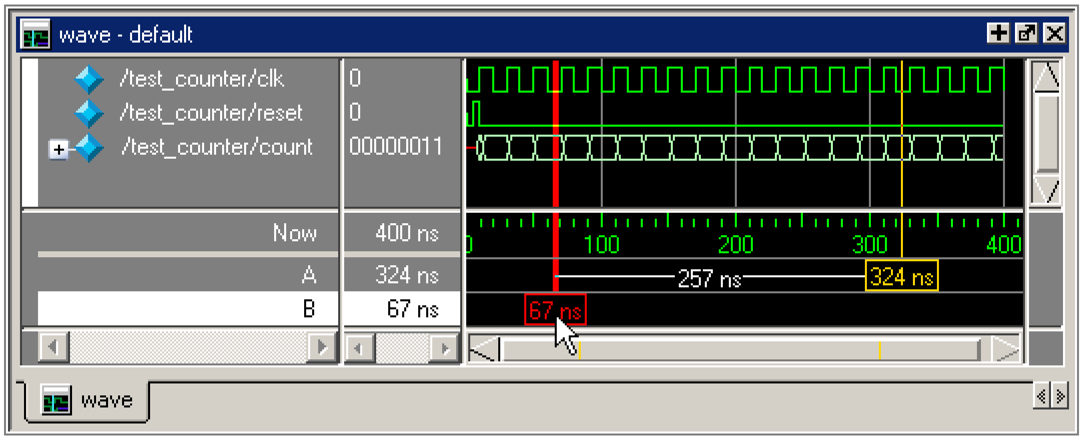


2. Lock cursor *B*.

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

   The cursor color changes to red and you can no longer drag the cursor (Figure 6-7).

Figure 6-7. A Locked Cursor in the Wave Window



3. Delete cursor *B*.

   a. Right-click cursor *B* and select **Delete B**.

Lesson Wrap-Up

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

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

Chapter 7
Viewing And Initializing Memories

Introduction

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

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

Design Files for this Lesson

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

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

Related Reading

User's Manual Section: Memory Panes.

Reference Manul commands: mem display, mem load, mem save, and radix.

Compile and Load the Design

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

   Start by creating a new directory for this exercise (in case other users will be working with these lessons). Create the directory and copy all files from *<install_dir>/examples/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 the previous lesson, ModelSim should already be running. If not, start ModelSim.

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

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

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

3. Create the working library and compile the design.

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

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

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

4. Load the design.

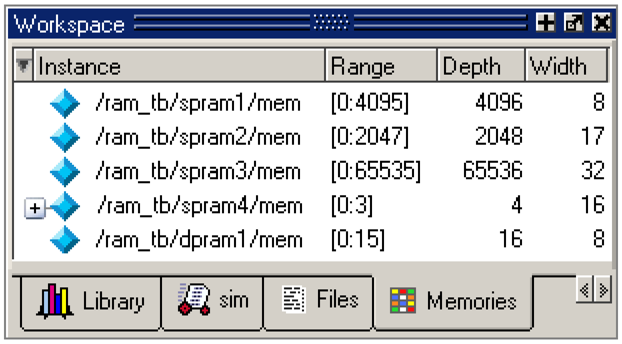
   a. On the Library tab of the Main window Workspace, click the "+" icon next to the *work* library.

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

View a Memory and its Contents

The Memories tab of the Main window lists all memories in the design (Figure 7-1) when the design is loaded; with the range, depth, and width of each memory displayed.

Figure 7-1. Viewing the Memories Tab in the Main Window Workspace



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

VHDL: The radix for enumerated types is Symbolic. To change the radix to binary for the purposes of this lesson, type the following command at the VSIM> prompt:

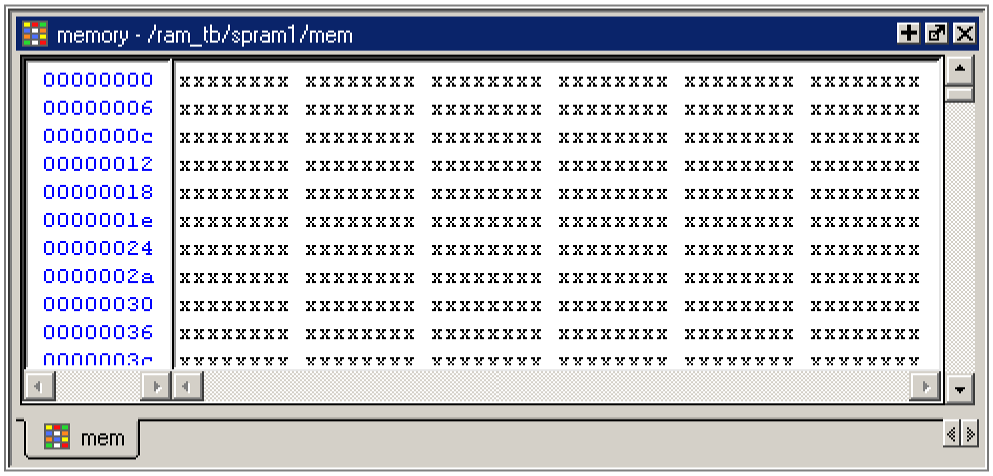
```
radix bin
```

1. Open a Memory instance to show its contents.

   a. Double-click the */ram_tb/spram1/mem* instance in the memories list to view its contents in the MDI frame.

   A **mem** tab is created in the MDI frame to display the memory contents. The data are all **X** (**0** in VHDL) since you have not yet simulated the design. The first column (blue hex characters) lists the addresses (Figure 7-2), and the remaining columns show the data values.

Figure 7-2. The mem Tab in the MDI Frame Shows Addresses and Data



```
00000000  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000006  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000000c  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000012  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000018  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000001e  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000024  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000002a  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000030  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
00000036  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
0000003c  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
```

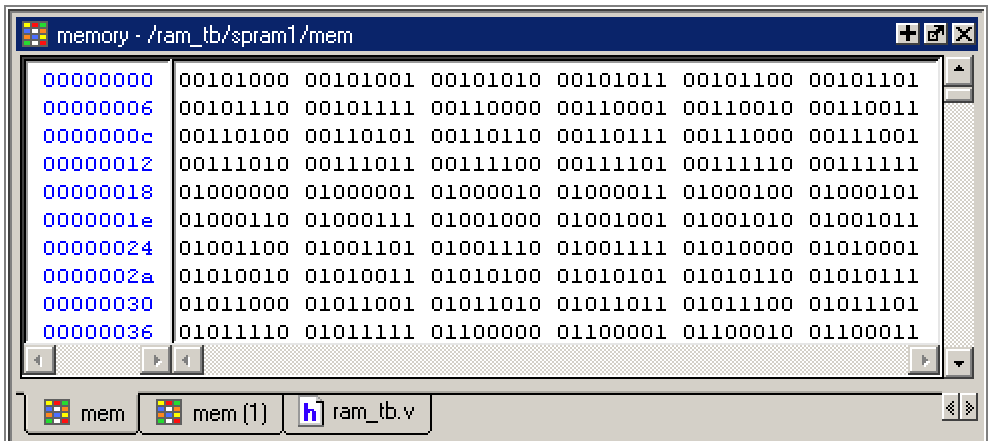
b. Double-click instance */ram_tb/spram2/mem* in the Memories tab of the Workspace, This creates a new tab in the MDI frame called **mem(1)** that contains the addresses and data for the *spram2* instance. Each time you double-click a new memory instance in the Workspace, a new tab is created for that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

2. Simulate the design.

a. Click the **run -all** icon in the Main window.

b. Click the **mem** tab of the MDI frame to bring the */ram_tb/spram1/mem* to the foreground. The data fields now show values (Figure 7-3).

Figure 7-3. The Memory Display Updates with the Simulation



```
00000000  00101000 00101001 00101010 00101011 00101100 00101101
00000006  00101110 00101111 00110000 00110001 00110010 00110011
0000000c  00110100 00110101 00110110 00110111 00111000 00111001
00000012  00111010 00111011 00111100 00111101 00111110 00111111
00000018  01000000 01000001 01000010 01000011 01000100 01000101
0000001e  01000110 01000111 01001000 01001001 01001010 01001011
00000024  01001100 01001101 01001110 01001111 01010000 01010001
0000002a  01010010 01010011 01010100 01010101 01010110 01010111
00000030  01011000 01011001 01011010 01011011 01011100 01011101
00000036  01011110 01011111 01100000 01100001 01100010 01100011
```

VHDL:
In the Transcript pane, you will see NUMERIC_STD warnings that can be ignored and

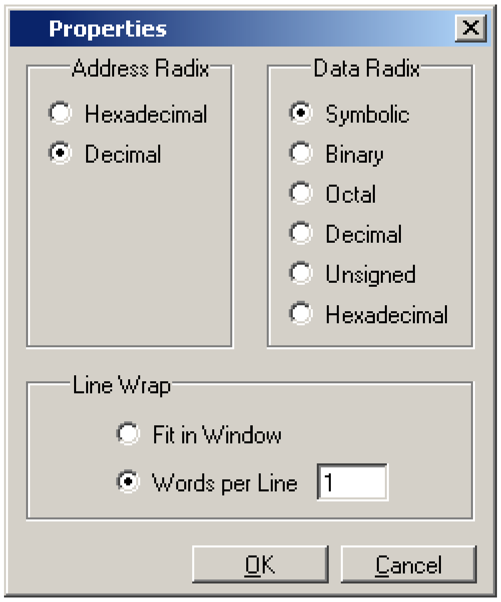
an assertion failure that is functioning to stop the simulation. The simulation itself has not failed.

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

    a. Right-click anywhere in the Memory Contents pane and select **Properties**.

    b. The Properties dialog box opens (Figure 7-4).

Figure 7-4. 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

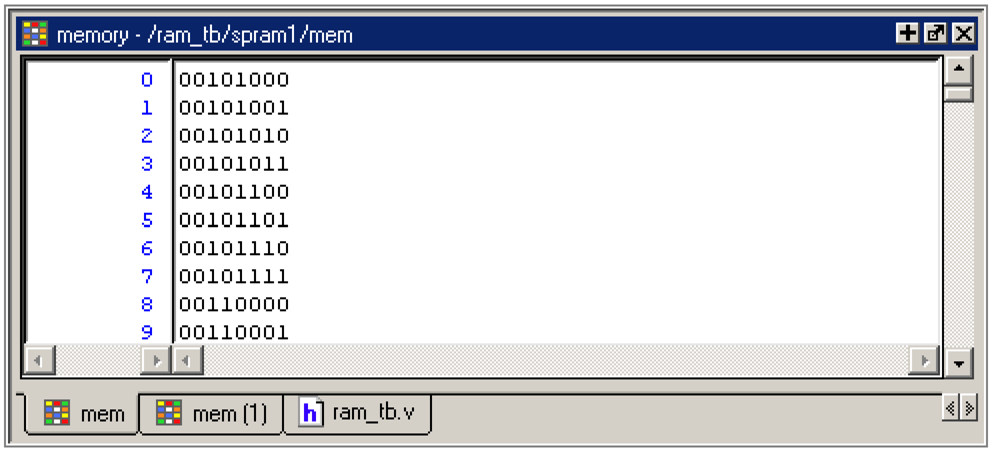
    c. For the **Address Radix,** select **Decimal**. This changes the radix for the addresses only.

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

    e. Click OK.

You can see the results of the settings in Figure 7-5. If the figure doesn't 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 7-5. New Address Radix and Line Length



```
memory - /ram_tb/spram1/mem
 0 00101000
 1 00101001
 2 00101010
 3 00101011
 4 00101100
 5 00101101
 6 00101110
 7 00101111
 8 00110000
 9 00110001
```

Navigate Within the Memory

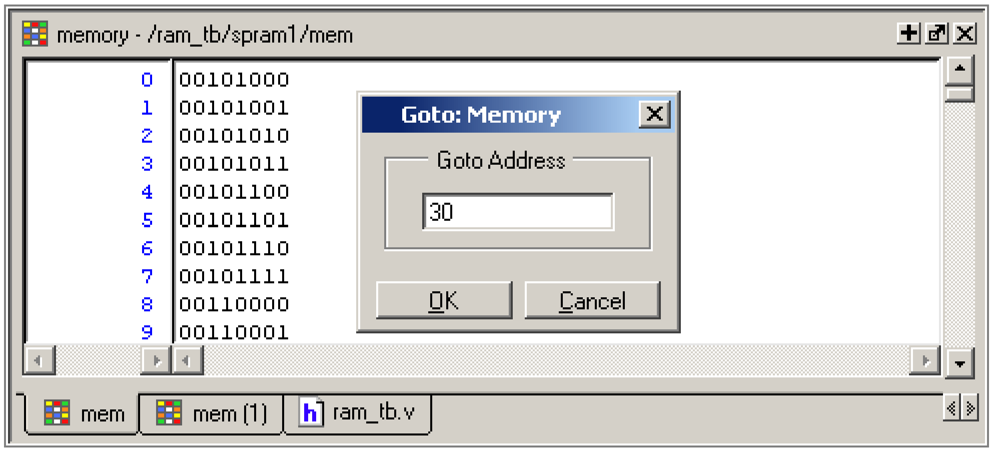
You can navigate to specific memory address locations, or to locations containing particular data patterns. First, you will go to a specific address.

1. Use Goto to find a specific address.

   a. Right-click anywhere in address column and select **Goto** (Figure 7-6).

   The Goto dialog box opens in the data pane.

Figure 7-6. Goto Dialog



```
memory - /ram_tb/spram1/mem
 0 00101000
 1 00101001
 2 00101010
 3 00101011
 4 00101100
 5 00101101
 6 00101110
 7 00101111
 8 00110000
 9 00110001
```

Goto: Memory
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.

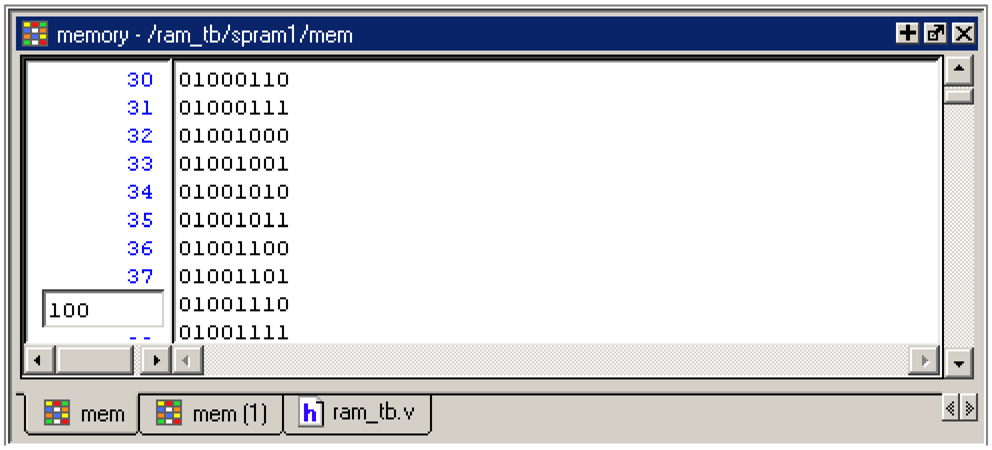
2. Edit the address location directly.

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

b. Double click address 38 in the address column.

c. Enter address 100 (Figure 7-7).

Figure 7-7. Editing the Address Directly



memory - /ram_tb/spram1/mem

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

d. Press <Enter> on your keyboard.

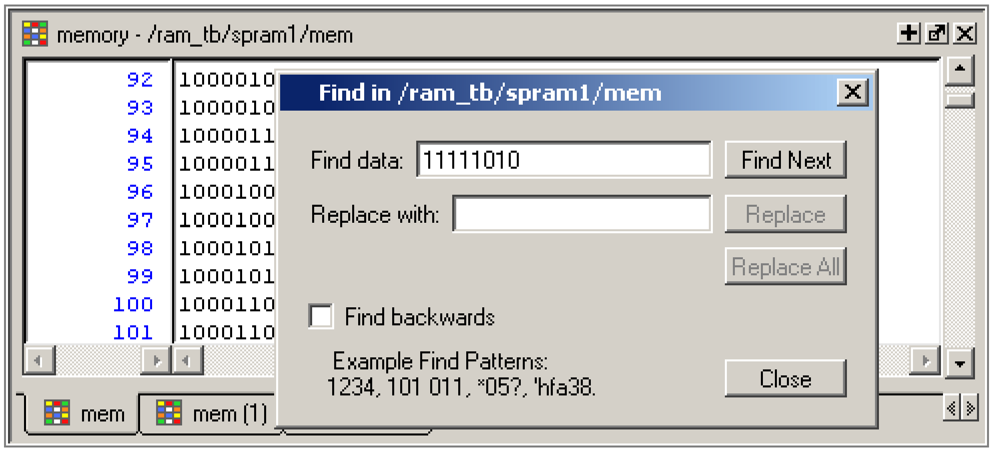
The pane scrolls to that address.

3. Now, let’s find a particular data entry.

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

The Find in dialog box opens (Figure 7-8).

Figure 7-8. Searching for a Specific Data Value



memory - /ram_tb/spram1/mem

```
92   1000010
93   1000010
94   1000011
95   1000011
96   1000100
97   1000100
98   1000101
99   1000101
100  1000110
101  1000110
```

Find in /ram_tb/spram1/mem

Find data: 11111010

Replace with:

Find backwards

Example Find Patterns:
1234, 101 011, *05?, 'hfa38.

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

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

c. Click **Close** to close the dialog box.

Export Memory Data to a File

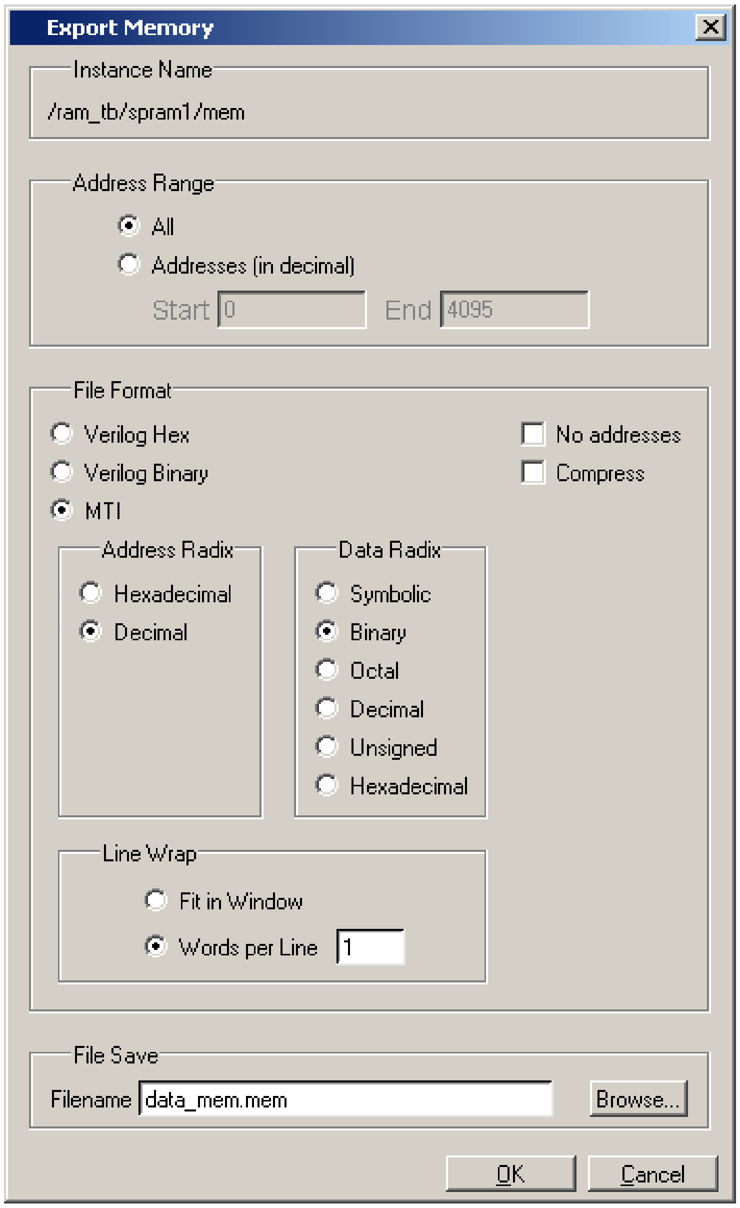
You can save memory data to a file that can be loaded at some later point in simulation.

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 in the MDI frame.

   b. Select **File > Export > Memory Data** to bring up the Export Memory dialog box (Figure 7-9).

Figure 7-9. Export Memory Dialog



Export Memory

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 Line Wrap, set to 1 word per line.

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 **mem(1)** tab in the MDI pane to see the data 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, 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.

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 click **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 Line Wrap, set 1 **Words per Line**.

i. Enter the file name as **reloc.mem**, then click OK to save the memory contents and close the dialog. You will use this file for initialization in the next section.

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, let’s initialize a memory from a file only. You will use one you exported previously, *data_mem.mem*.

1. View instance */ram_tb/spram3/mem*.

a. Double-click the */ram_tb/spram3/mem* instance in the Memories tab.

This will open a new tab – **mem(2)** – in the MDI frame to display the contents of */ram_tb/spram3/mem*. Scan these contents so you can identify changes once the initialization is complete.

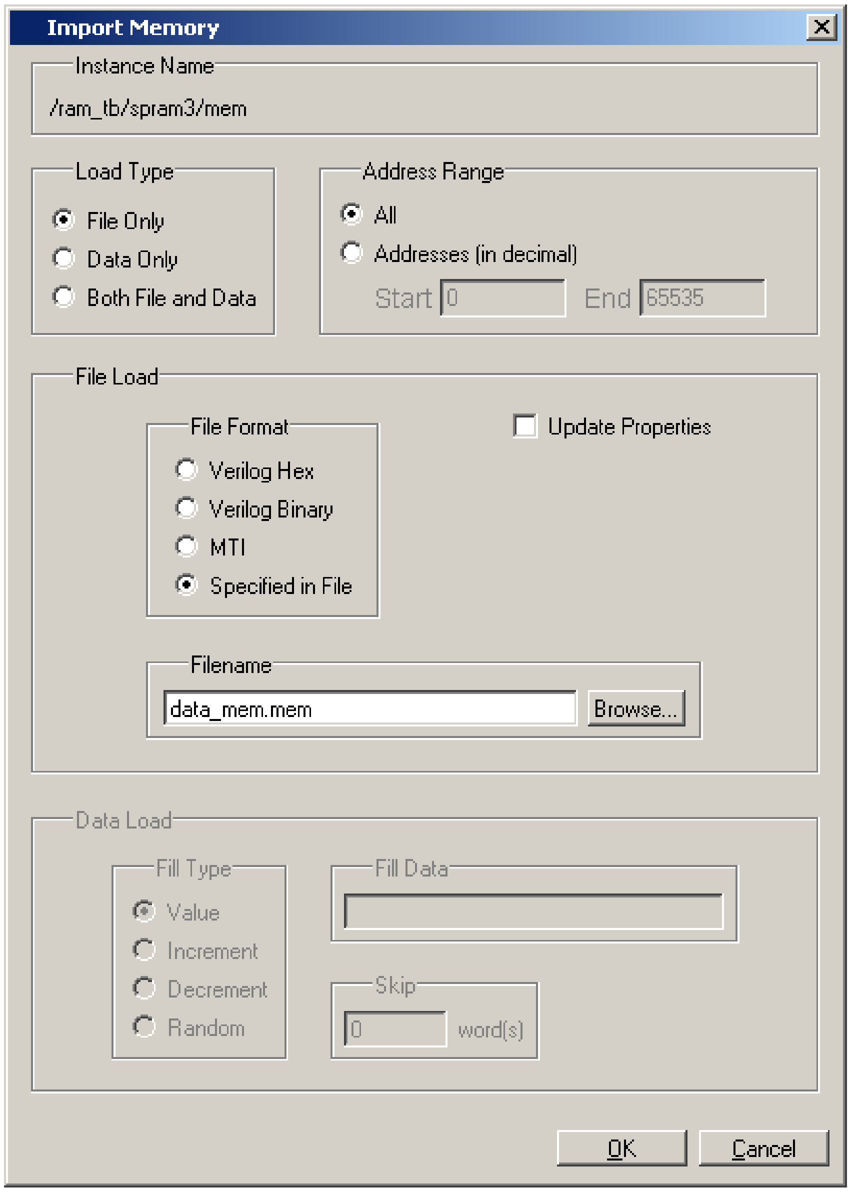
b. Right-click and select **Properties** to bring up the Properties dialog.

c. Change the Address Radix to **Decimal**, Data Radix to **Binary, Line Wrap to 1 Words per Line,** and click OK.

2. Initialize *spram3* from a file.

   a. Right-click anywhere in the data column and select **Import** to bring up the Import Memory dialog box (Figure 7-10).

Figure 7-10. Import Memory Dialog



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 ☐

Filename: data_mem.mem

Data Load

Fill Type: [✓ Value] Increment Decrement Random

Fill Data:

Skip: 0 word(s)

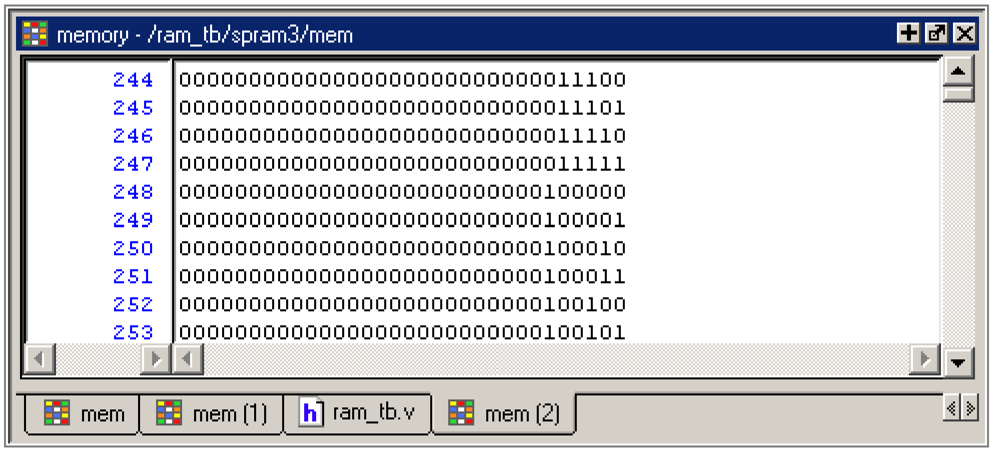
The default Load Type is File Only.

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 7-11).

Figure 7-11. Initialized Memory from File and Fill Pattern



```
244  00000000000000000000000000011100
245  00000000000000000000000000011101
246  00000000000000000000000000011110
247  00000000000000000000000000011111
248  00000000000000000000000000100000
249  00000000000000000000000000100001
250  00000000000000000000000000100010
251  00000000000000000000000000100011
252  00000000000000000000000000100100
253  00000000000000000000000000100101
```

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 the **mem(2)** tab and select **Import** 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**.

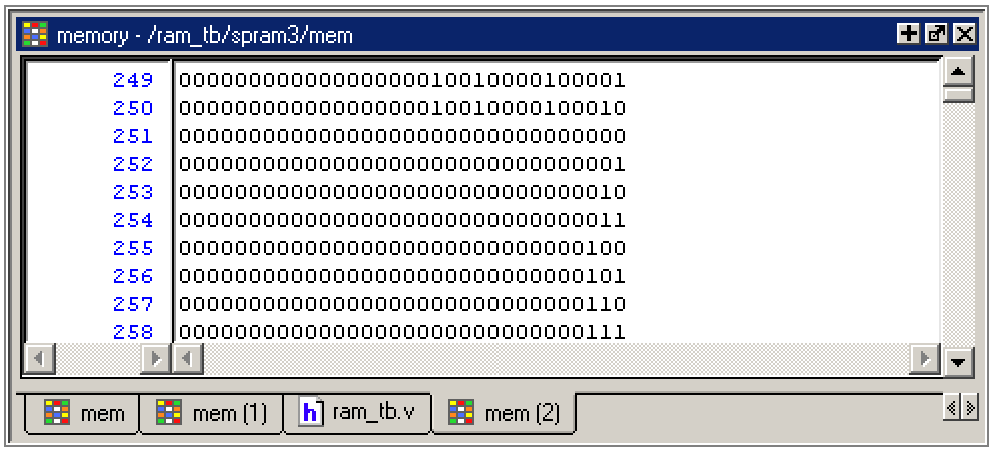
   f. In the Fill Data field, set the seed value of **0** for the incrementing data.

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 7-12).

Figure 7-12. Data Increments Starting at Address 251



memory - /ram_tb/spram3/mem

```
249  00000000000000000010010000100001
250  00000000000000000010010000100010
251  00000000000000000000000000000000
252  00000000000000000000000000000001
253  00000000000000000000000000000010
254  00000000000000000000000000000011
255  00000000000000000000000000000100
256  00000000000000000000000000000101
257  00000000000000000000000000000110
258  00000000000000000000000000000111
```

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

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

Interactive Debugging Commands

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

1. Open a memory instance and change its display characteristics.

   a. Double-click instance */ram_tb/dpram1/mem* in the Memories tab.

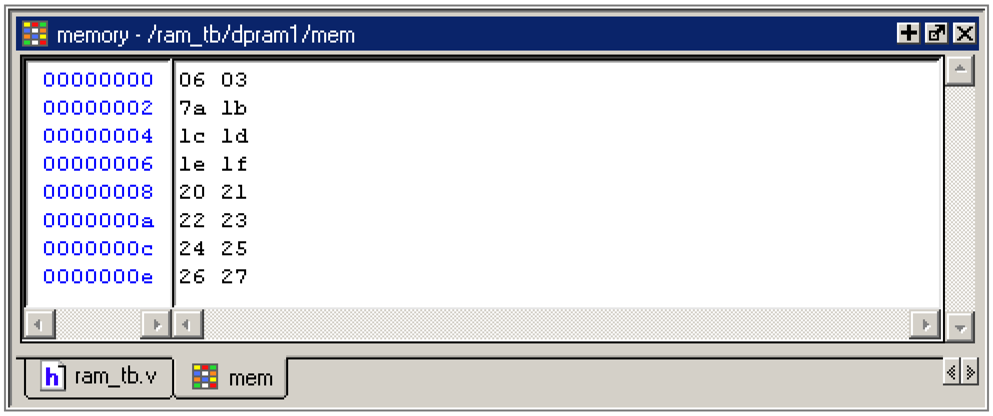
   b. Right-click in the memory contents pane and select **Properties**.

   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 in Figure 7-13.

Figure 7-13. Original Memory Content

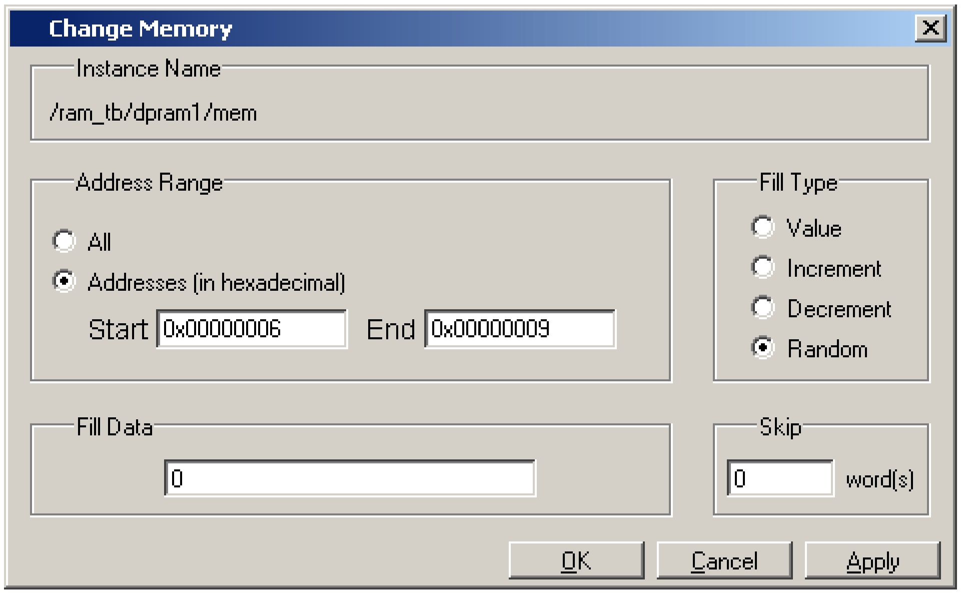

memory - /ram_tb/dpram1/mem

```
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* contents pane and select **Change** to open the Change Memory dialog (Figure 7-14).

Figure 7-14. 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.

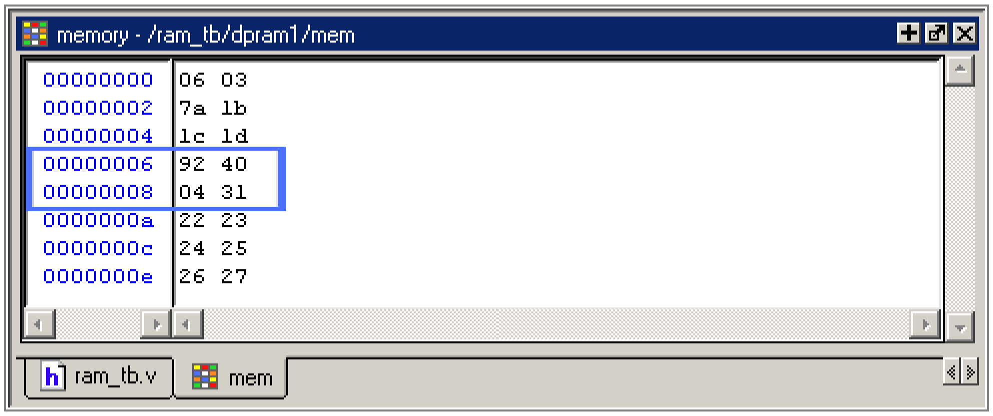
   c. Select **Random** as the **Fill Type**.

   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 7-15).

Figure 7-15. Random Content Generated for a Range of Addresses



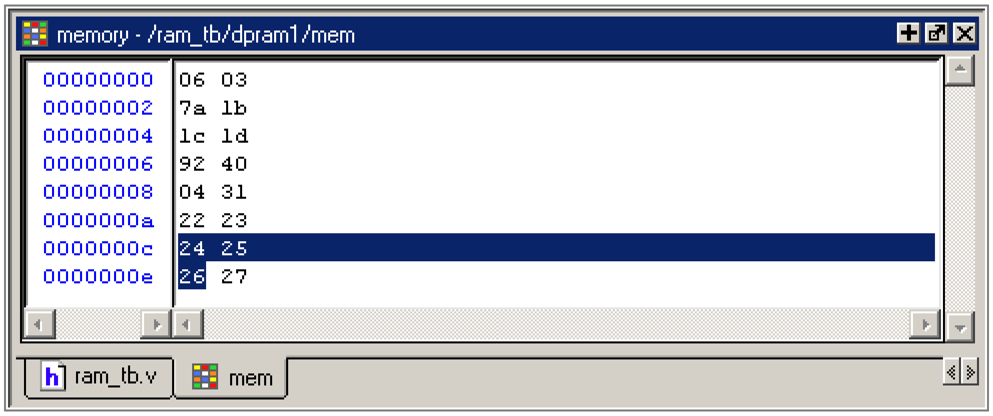
```
00000000 06 03
00000002 7a 1b
00000004 1c 1d
00000006 92 40
00000008 04 31
0000000a 22 23
0000000c 24 25
0000000e 26 27
```

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 7-16.

Figure 7-16. Changing Memory Contents by Highlighting

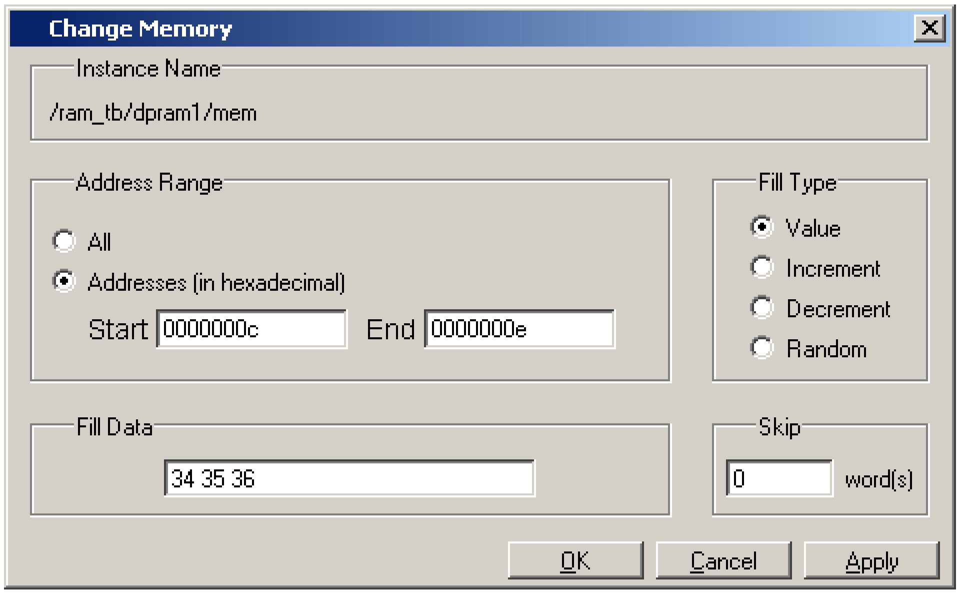


```
00000000 06 03
00000002 7a 1b
00000004 1c 1d
00000006 92 40
00000008 04 31
0000000a 22 23
0000000c 24 25
0000000e 26 27
```

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

      This brings up the Change memory dialog box (Figure 7-17). Note that the Addresses field is already populated with the range you highlighted.

Figure 7-17. Entering Data to Change



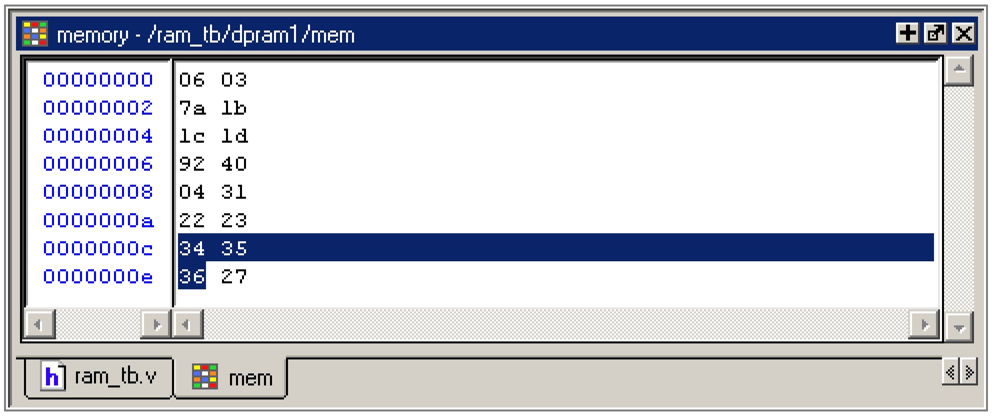
c. Select **Value** as the Fill Type.

d. Enter the data values into the Fill Data field as follows: **34 35 36**

e. Click **OK**.

The data in the address locations change to the values you entered (Figure 7-18).

Figure 7-18. 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  34 35
0000000e  36 27
```

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 <Enter> 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 we need to end the current simulation.

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

Chapter 8
Automating Simulation

Introduction

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

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

Note

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

Related Reading

User's Manual Chapter: Tcl and Macros (DO Files).

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

Creating a Simple DO File

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

1. Load the *test_counter* design unit.
   a. If necessary, start ModelSim.
   b. Change to the directory you created in Lesson 2.
   c. In the Library tab of the Workspace pane, double-click the *test_counter* design unit to load it.

2. Enter commands to add signals to the Wave window, force signals, and run the simulation.

   a. Select **File** > **New** > **Source** > **Do** to create a new DO file.

   b. Enter the following commands into the source window:

   ```
   add wave count
   add wave clk
   add wave reset
   force -freeze clk 0 0, 1 {50 ns} -r 100
   force reset 1
   run 100
   force reset 0
   run 300
   force reset 1
   run 400
   force reset 0
   run 200
   ```

3. Save the file.

   a. Select **File** > **Save As**.

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

4. Load the simulation again and use the DO file.

   a. Enter **quit -sim** at the VSIM> prompt.

   b. Enter **vsim test_counter** at the ModelSim> prompt.

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

   ModelSim executes the saved commands and draws the waves in the Wave window.

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

Running in Command-Line Mode

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

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

   Start by creating a new directory for this exercise. Create the directory and copy 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 *

# read in stimulus
do stim.do

# output results
write list counter.lst

# quit the simulation
quit -f
```

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

4. Run the batch-mode simulation.

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

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

5. View the list output.

   a. Open *counter.lst* and view the simulation results. Output produced by the Verilog version of the design should look like the following:

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

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

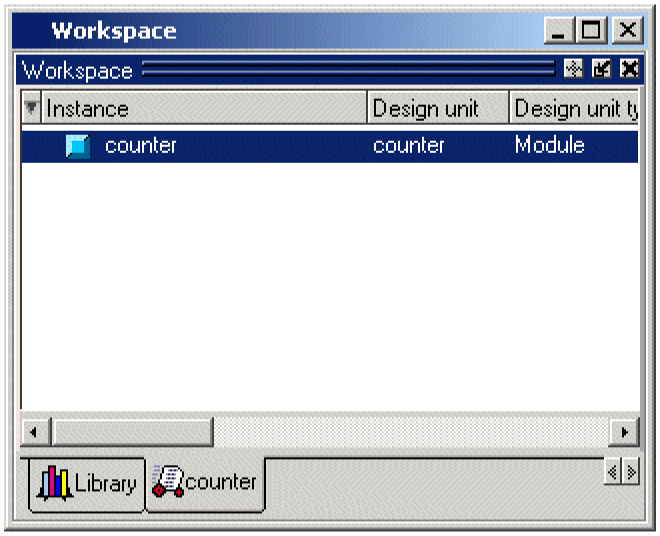
Note

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 DOS/ UNIX prompt.

      The GUI opens and a dataset tab named "counter" is displayed in the Workspace (Figure 8-1).

Figure 8-1. A Dataset in the Main Window Workspace



Instance	Design unit	Design unit ty
counter	counter	Module

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

      The waveforms display in the Wave window.

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

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.

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

1. Create the script.

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

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

   These commands do the following:

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

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

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

   These commands do the following:

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

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

   Save it into the directory you created in Basic Simulation.

2. Load the *test_counter* design unit.

    a. Start ModelSim.

    b. Select **File > Change Directory** and change to the directory you saved the DO file to in step 1c above.

    c. In the Library tab of the Main window, expand the *work* library and double-click the *test_counter* design unit.

3. Execute the DO file and run the design.

    a. Type **do add_bkmrk.do** at the VSIM> prompt.

    b. Type **run 1500 ns** at the VSIM> prompt.

    The simulation runs and the DO file creates two bookmarks.

    c. If the Wave window is docked in the Main window, click somewhere in the Wave window and select **View > Wave > Bookmarks > bm1**. If the window is undocked, select **View > Bookmarks > bm1** in the Wave window.

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

Lesson Wrap-Up

This concludes this lesson.

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

Index

End-User License Agreement

The latest version of the End-User License Agreement is available on-line at:
www.mentor.com/terms_conditions/enduser.cfm

IMPORTANT INFORMATION

USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE. USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. ANY ADDITIONAL OR DIFFERENT PURCHASE ORDER TERMS AND CONDITIONS SHALL NOT APPLY.

END-USER LICENSE AGREEMENT ("Agreement")

This is a legal agreement concerning the use of Software between you, the end user, as an authorized representative of the company acquiring the license, and Mentor Graphics Corporation and Mentor Graphics (Ireland) Limited acting directly or through their subsidiaries (collectively "Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by you and an authorized representative of Mentor Graphics, this Agreement and the applicable quotation contain the parties' entire understanding relating to the subject matter and supersede all prior or contemporaneous agreements. If you do not agree to these terms and conditions, promptly return or, if received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

1. **GRANT OF LICENSE.** The software programs, including any updates, modifications, revisions, copies, documentation and design data ("Software"), are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form; (b) for your internal business purposes; (c) for the license term; and (d) on the computer hardware and at the site authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or services purchased, apply to the following: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be technically implemented through the use of authorization codes or similar devices); and (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions.

2. **EMBEDDED SOFTWARE.** If you purchased a license to use embedded software development ("ESD") Software, if applicable, Mentor Graphics grants to you a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESD compilers, including the ESD run-time libraries distributed with ESD C and C++ compiler Software that are linked into a composite program as an integral part of your compiled computer program, provided that you distribute these files only in conjunction with your compiled computer program. Mentor Graphics does NOT grant you any right to duplicate, incorporate or embed copies of Mentor Graphics' real-time operating systems or other embedded software products into your products or applications without first signing or otherwise agreeing to a separate agreement with Mentor Graphics for such purpose.

3. **BETA CODE.** Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to you a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and your use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on your feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this section 3 shall survive the termination or expiration of this Agreement.

4. **RESTRICTIONS ON USE.** You may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. You shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. You shall not make Software available in any form to any person other than employees and on-site contractors, excluding Mentor Graphics' competitors, whose job performance requires access and who are under obligations of confidentiality. You shall take appropriate action to protect the confidentiality of Software and ensure that any person permitted access to Software does not disclose it or use it except as permitted by this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, you shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive from Software any source code. You may not sublicense, assign or otherwise transfer Software, this Agreement or the rights under it, whether by operation of law or otherwise ("attempted transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable transfer charges. Any attempted transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and licenses granted under this Agreement. The terms of this Agreement, including without limitation, the licensing and assignment provisions shall be binding upon your successors in interest and assigns. The provisions of this section 4 shall survive the termination or expiration of this Agreement.

5. **LIMITED WARRANTY.**

   5.1. Mentor Graphics warrants that during the warranty period Software, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Software will meet your requirements or that operation of Software will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. You must notify Mentor Graphics in writing of any nonconformity within the warranty period. This warranty shall not be valid if Software has been subject to misuse, unauthorized modification or improper installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF SOFTWARE TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF SOFTWARE THAT DOES NOT MEET THIS LIMITED WARRANTY, PROVIDED YOU HAVE OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) SOFTWARE WHICH IS LICENSED TO YOU FOR A LIMITED TERM OR LICENSED AT NO COST; OR (C) EXPERIMENTAL BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

   5.2. THE WARRANTIES SET FORTH IN THIS SECTION 5 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO SOFTWARE OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

6. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT PAID BY YOU FOR THE SOFTWARE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER. THE PROVISIONS OF THIS SECTION 6 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

7. **LIFE ENDANGERING ACTIVITIES.** NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF SOFTWARE IN ANY APPLICATION WHERE THE FAILURE OR INACCURACY OF THE SOFTWARE MIGHT RESULT IN DEATH OR PERSONAL INJURY. THE PROVISIONS OF THIS SECTION 7 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

8. **INDEMNIFICATION.** YOU AGREE TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE, OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH YOUR USE OF SOFTWARE AS

DESCRIBED IN SECTION 7. THE PROVISIONS OF THIS SECTION 8 SHALL SURVIVE THE EXPIRATION OR TERMINATION OF THIS AGREEMENT.

9. **INFRINGEMENT.**

9.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against you alleging that Software infringes a patent or copyright or misappropriates a trade secret in the United States, Canada, Japan, or member state of the European Patent Office. Mentor Graphics will pay any costs and damages finally awarded against you that are attributable to the infringement action. You understand and agree that as conditions to Mentor Graphics' obligations under this section you must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to defend or settle the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

9.2. If an infringement claim is made, Mentor Graphics may, at its option and expense: (a) replace or modify Software so that it becomes noninfringing; (b) procure for you the right to continue using Software; or (c) require the return of Software and refund to you any license fee paid, less a reasonable allowance for use.

9.3. Mentor Graphics has no liability to you if infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you make, use or sell; (f) any Beta Code contained in Software; (g) any Software provided by Mentor Graphics’ licensors who do not provide such indemnification to Mentor Graphics’ customers; or (h) infringement by you that is deemed willful. In the case of (h) you shall reimburse Mentor Graphics for its attorney fees and other costs related to the action upon a final judgment.

9.4. THIS SECTION IS SUBJECT TO SECTION 6 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.

10. **TERM.** This Agreement remains effective until expiration or termination. This Agreement will immediately terminate upon notice if you exceed the scope of license granted or otherwise fail to comply with the provisions of Sections 1, 2, or 4. For any other material breach under this Agreement, Mentor Graphics may terminate this Agreement upon 30 days written notice if you are in material breach and fail to cure such breach within the 30 day notice period. If Software was provided for limited term use, this Agreement will automatically expire at the end of the authorized term. Upon any termination or expiration, you agree to cease all use of Software and return it to Mentor Graphics or certify deletion and destruction of Software, including all copies, to Mentor Graphics’ reasonable satisfaction.

11. **EXPORT.** Software is subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct products of the products to certain countries and certain persons. You agree that you will not export any Software or direct product of Software in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.

12. **RESTRICTED RIGHTS NOTICE.** Software was developed entirely at private expense and is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement under which Software was obtained pursuant to DFARS 227.7202-3(a) or as set forth in subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable. Contractor/manufacturer is Mentor Graphics Corporation, 8005 SW Boeckman Road, Wilsonville, Oregon 97070-7777 USA.

13. **THIRD PARTY BENEFICIARY.** For any Software under this Agreement licensed by Mentor Graphics from Microsoft or other licensors, Microsoft or the applicable licensor is a third party beneficiary of this Agreement with the right to enforce the obligations set forth herein.

14. **AUDIT RIGHTS.** You will monitor access to, location and use of Software. With reasonable prior notice and during your normal business hours, Mentor Graphics shall have the right to review your software monitoring system and reasonably relevant records to confirm your compliance with the terms of this Agreement, an addendum to this Agreement or U.S. or other local export laws. Such review may include FLEXlm or FLEXnet report log files that you shall capture and provide at Mentor Graphics’ request. Mentor Graphics shall treat as confidential information all of your information gained as a result of any request or review and shall only use or disclose such information as required by law or to enforce its rights under this Agreement or addendum to this Agreement. The provisions of this section 14 shall survive the expiration or termination of this Agreement.

15. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** THIS AGREEMENT SHALL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF THE STATE OF OREGON, USA, IF YOU ARE LOCATED IN NORTH OR SOUTH AMERICA, AND THE LAWS OF IRELAND IF YOU ARE LOCATED OUTSIDE OF NORTH OR SOUTH AMERICA. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply. Notwithstanding the foregoing, all disputes in Asia (except for Japan) arising out of or in relation to this Agreement shall be resolved by arbitration in Singapore before a single arbitrator to be appointed by the Chairman of the Singapore International Arbitration Centre ("SIAC") to be conducted in the English language, in accordance with the Arbitration Rules of the SIAC in effect at the time of the dispute, which rules are deemed to be incorporated by reference in this section 15. This section shall not restrict Mentor Graphics' right to bring an action against you in the jurisdiction where your place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

16. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.

17. **PAYMENT TERMS AND MISCELLANEOUS.** You will pay amounts invoiced, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. Any past due invoices will be subject to the imposition of interest charges in the amount of one and one-half percent per month or the applicable legal rate currently in effect, whichever is lower. Some Software may contain code distributed under a third party license agreement that may provide additional rights to you. Please see the applicable Software documentation for details. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.

    Rev. 060210, Part No. 227900