ModelSim® Tutorial

Software Version 10.5c

Mentor Graphics Corporation
8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777
Telephone: 503.685.7000
Toll-Free Telephone: 800.592.2210
Website: www.mentor.com
SupportNet: supportnet.mentor.com/

Send Feedback on Documentation: supportnet.mentor.com/doc_feedback_form

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

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

Before you Begin

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

Example Designs

ModelSim comes with Verilog and VHDL versions of the designs used in most of these lessons. This allows you to do the tutorial regardless of which license type you have. Though we have tried to minimize the differences between the Verilog and VHDL versions, we could not do so in all cases. In cases where the designs differ (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

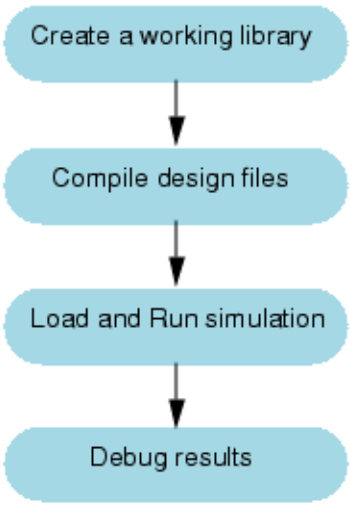
ModelSim is a verification and simulation tool for VHDL, Verilog, SystemVerilog.

This lesson provides a brief conceptual overview of the ModelSim simulation environment. It is divided into topics, which you will learn more about in subsequent 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,” which is the default library name used by the compiler as the default destination for compiled design units.

- Compiling Your Design

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

- Loading the Simulator with Your Design and Running the Simulation

  With the design compiled, you load the simulator with your design by invoking the simulator on a top-level module (Verilog) or a configuration or entity/architecture pair (VHDL).

  Assuming the design loads successfully, the simulation time is set to zero, and you enter a run command to begin simulation.

- Debugging Your Results

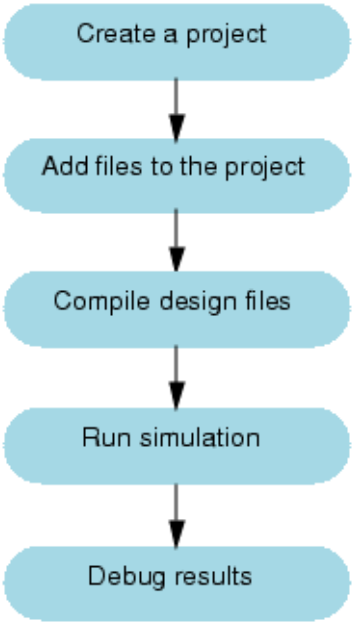
  If you don't get the results you expect, you can use ModelSim's robust debugging environment to track down the cause of the problem.

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.

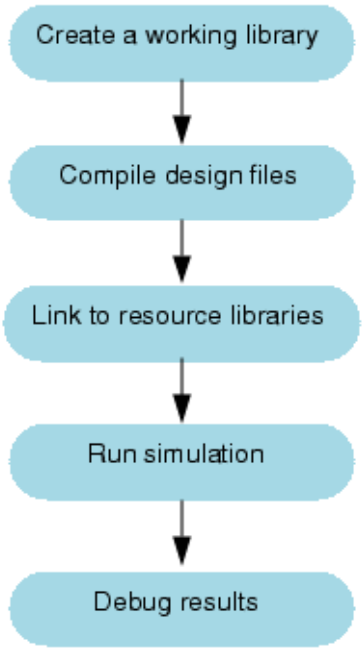
Multiple Library Flow

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 test bench are compiled into the working library, and the design references gate-level models in a separate resource library.

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

Figure 2-3. Multiple Library Flow



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

Debugging Tools

ModelSim offers numerous tools for debugging and analyzing your design.

Several of these tools are covered in subsequent lessons, including:

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

Chapter 3
Basic Simulation

In this lesson you will guide you through the basic simulation flow.

Design Files for this Lesson

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

The pathnames are as follows:

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

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

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

Refer to the following sections for further information:

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.

Procedure

1. Create a new directory and copy the design files for this lesson into it.

   Start by creating a new directory for this exercise (in case other users will be working with these lessons).

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

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

2. Start ModelSim *if necessary*.

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

      Upon opening ModelSim for the first time, you will see the Welcome to ModelSim dialog box. Click **Close**.

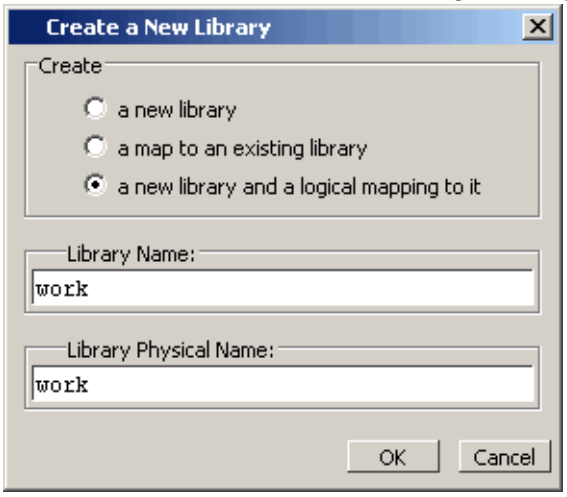
   b. Select **File > Change Directory** and change to the directory you created in step 1.

3. Create the working library.

   a. Select File > New > Library.

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

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



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

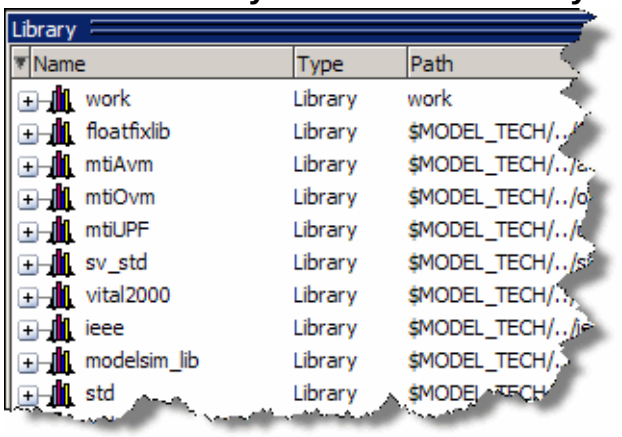
   c. Click **OK**.

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

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



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

```
vlib work
vmap work work
```

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

Compile the Design Units

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

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

Procedure

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

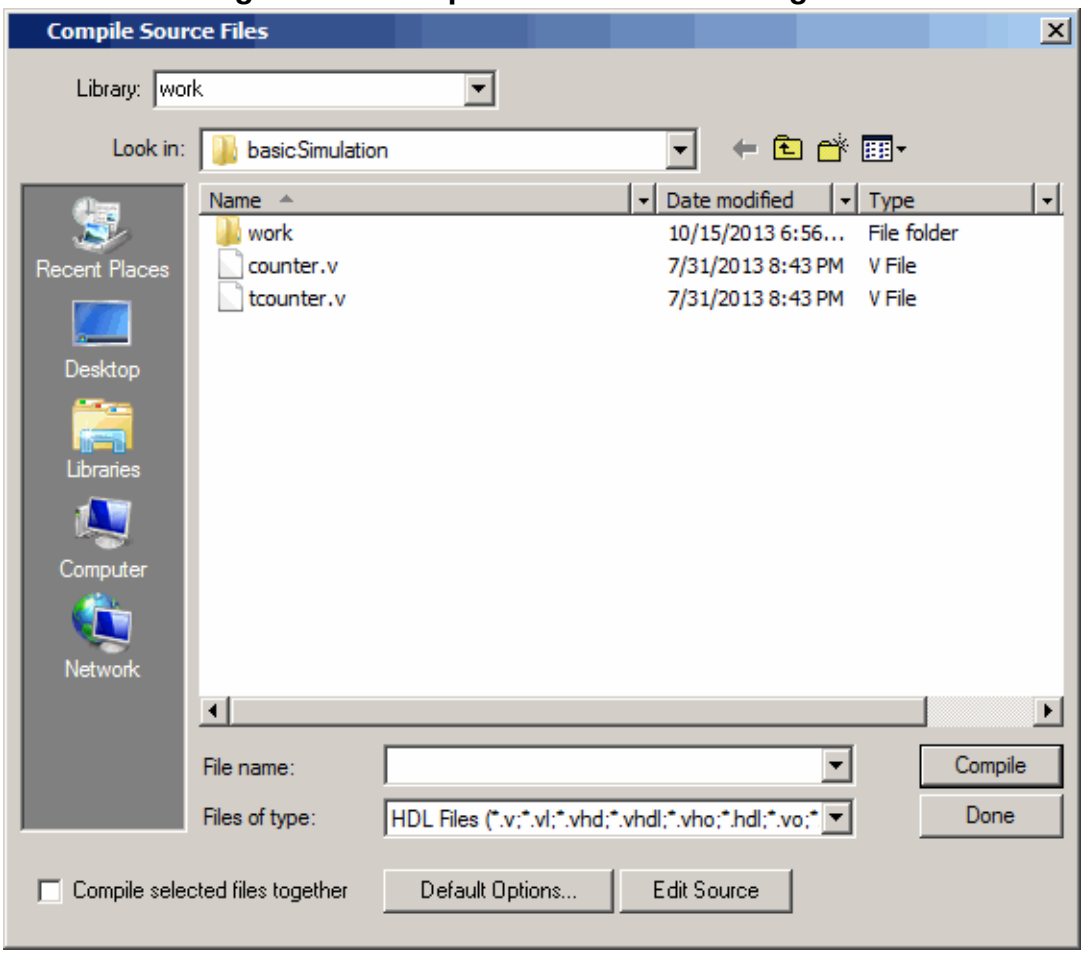
   a. Select **Compile > Compile**. This opens the Compile Source Files dialog box (Figure 3-3).

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

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

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

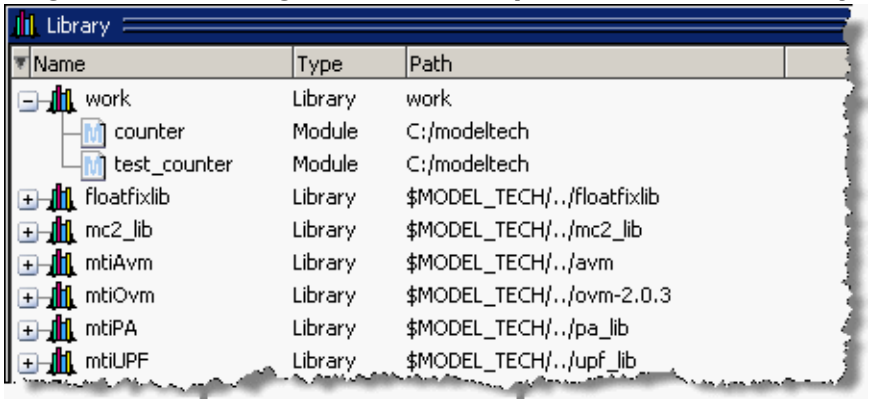
Figure 3-3. Compile Source Files Dialog Box



2. View the compiled design units.

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

Figure 3-4. Verilog Modules Compiled into work Library



Load the Design

Now you're ready to load the design into the simulator.

Procedure

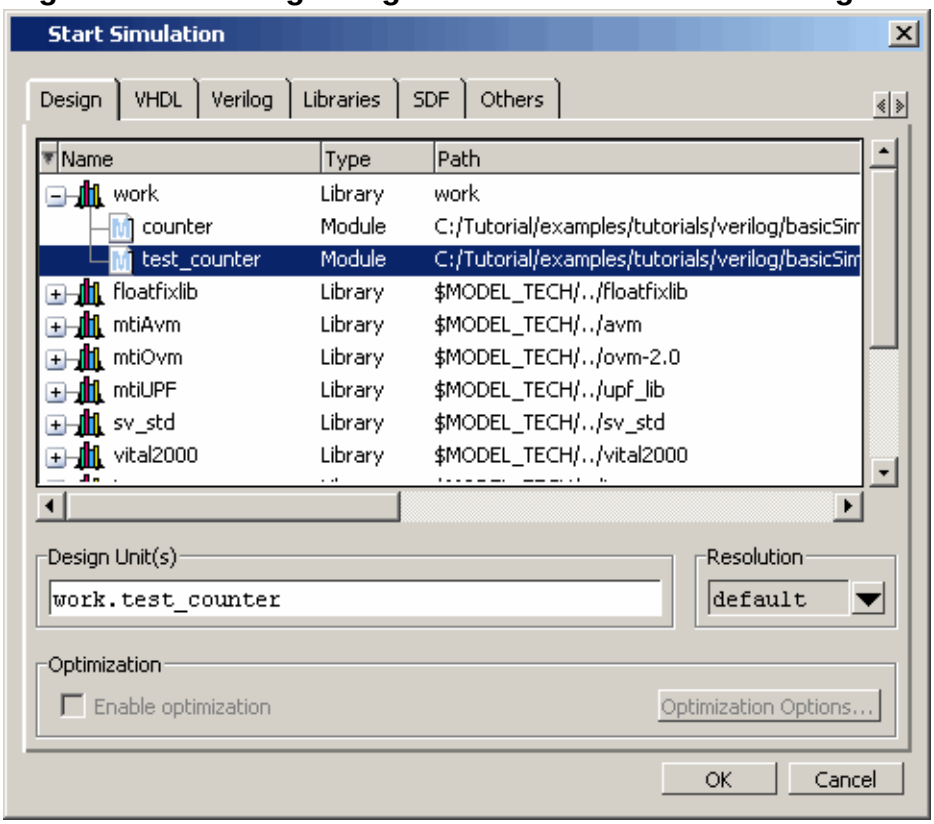
1. Load the *test_counter* module into the simulator.

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

   b. Double-click *test_counter* to load the design.

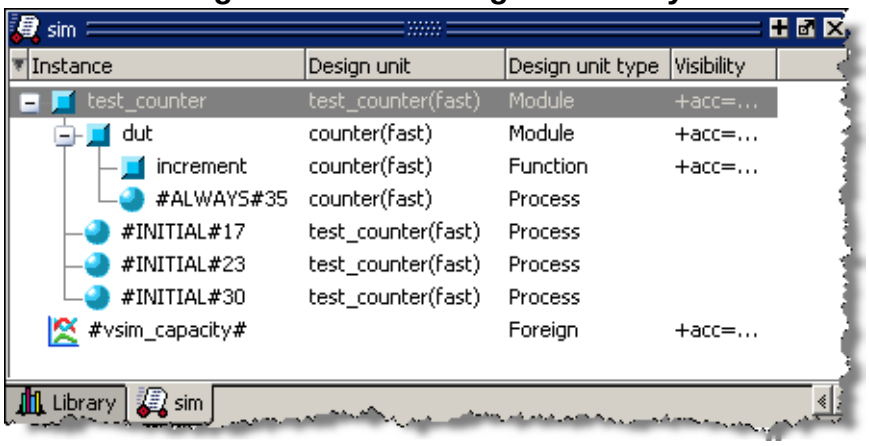
      You can also load the design by selecting Simulate > Start Simulation in the menu bar. This opens the Start Simulation dialog box. With the Design tab selected, click the '+' sign next to the work library to see the counter and test_counter modules. Select the test_counter module and click OK (Figure 3-5).

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



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

Figure 3-6. The Design Hierarchy



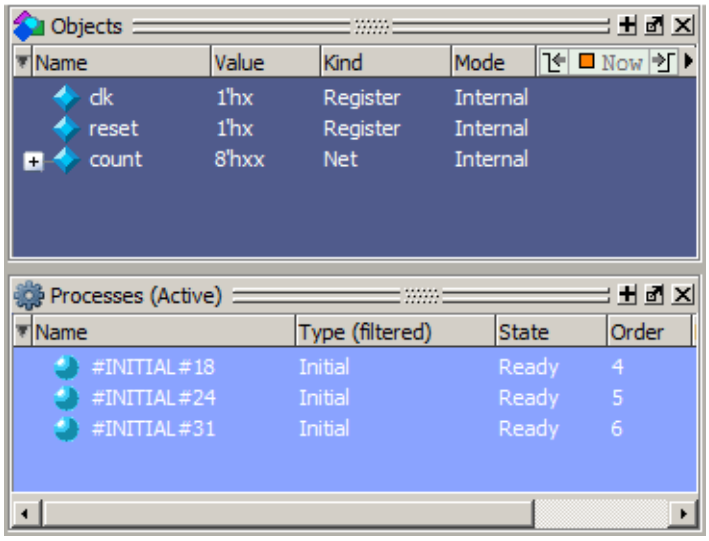
2. Open the Objects and Processes windows.
   a. Select **View** > **Objects** from the menu bar.

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

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

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

Figure 3-7. The Object Window and Processes Window



Run the Simulation

We're ready to run the simulation. But before we do, we'll open the Wave window and add signals to it.

Procedure

1. Open the Wave window.

   a. Enter view wave at the command line.

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

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

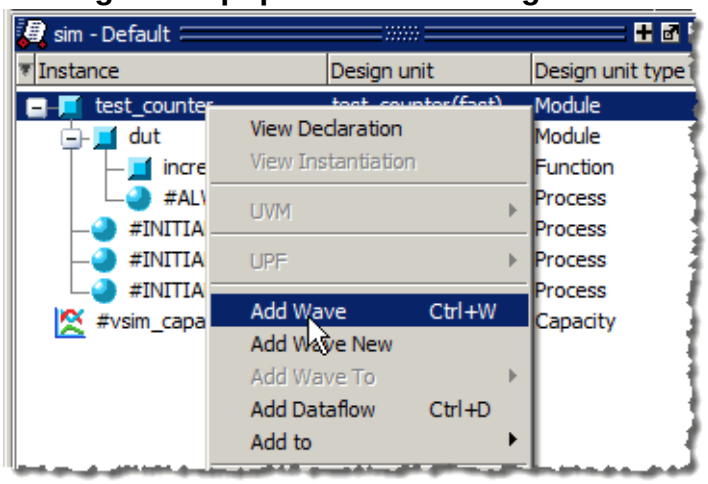
2. Add signals to the Wave window.

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

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

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

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



3. Run the simulation.

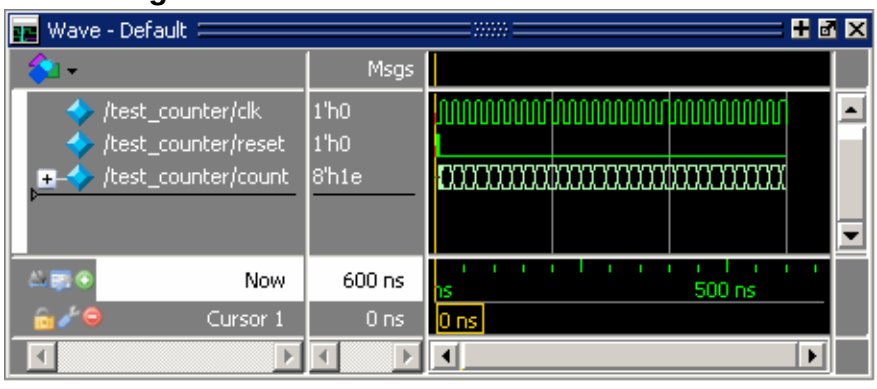
a. Click the Run icon. 

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

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

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

Figure 3-9. Waves Drawn in Wave Window



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

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

d. Click the Break icon  to stop the simulation.

Set Breakpoints and Step through the Source

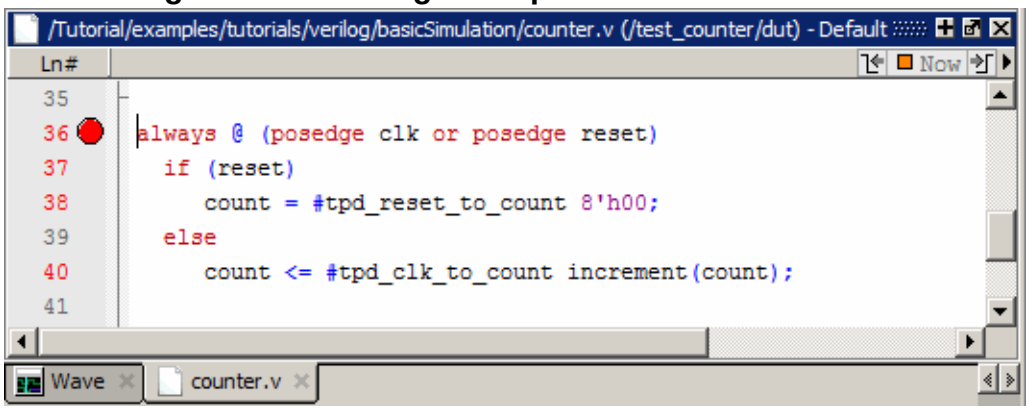
Next you will take a brief look at one interactive debugging feature of the ModelSim environment. You will set a breakpoint in the Source window, run the simulation, and then step through the design under test. Breakpoints can be set only on executable lines, which are indicated with red line numbers.

Procedure

1. Open counter.v in the Source window.
   a. Select **View > Files** to open the Files window.
   b. Click the + sign next to the *sim* filename to see the contents of *vsim.wlf* dataset.
   c. Double-click counter.v (or *counter.vhd* if you are simulating the VHDL files) to open the file in the Source window.
2. Set a breakpoint on line 36 of *counter.v* (or, line 39 of counter.vhd for VHDL).
   a. Scroll to line 36 and click in the Ln# (line number) column next to the line number.

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

Figure 3-10. Setting Breakpoint in Source Window



3. Disable, enable, and delete the breakpoint.
   a. Click the red dot to disable the breakpoint. It will become a gray dot.
   b. Click the gray dot again to re-enable the breakpoint. It will become a red dot.
   c. Click the red dot with your right mouse button and select **Remove Breakpoint 36**.

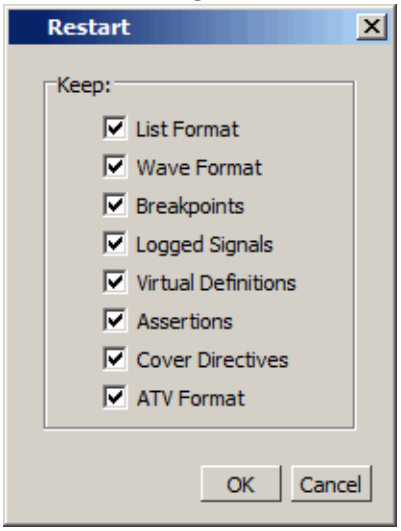
d. Click in the line number column next to line number 36 again to re-create the breakpoint.

4. Restart the simulation.

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

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

Figure 3-11. Setting Restart Functions

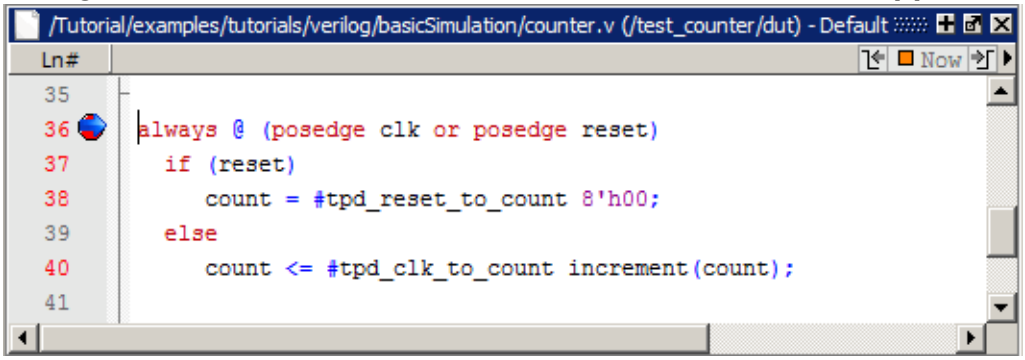


b. Click the **OK** button in the Restart dialog box.

c. Click the Run -All icon. 

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

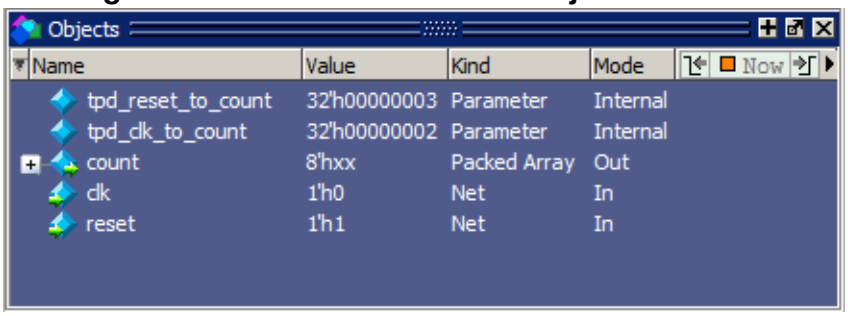
Figure 3-12. Blue Arrow Indicates Where Simulation Stopped.



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

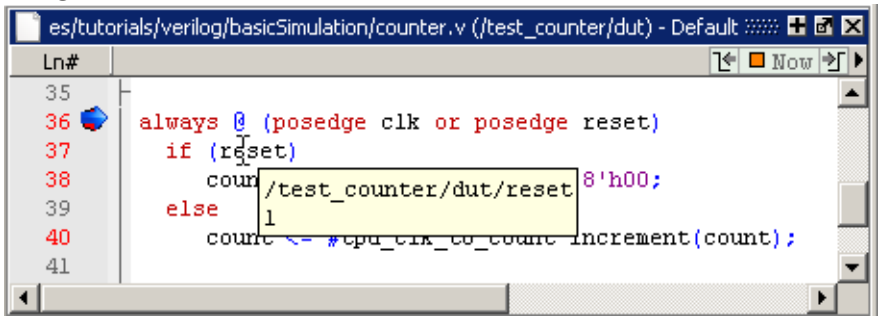
- Look at the values shown in the Objects window (Figure 3-13).

Figure 3-13. Values Shown in Objects Window



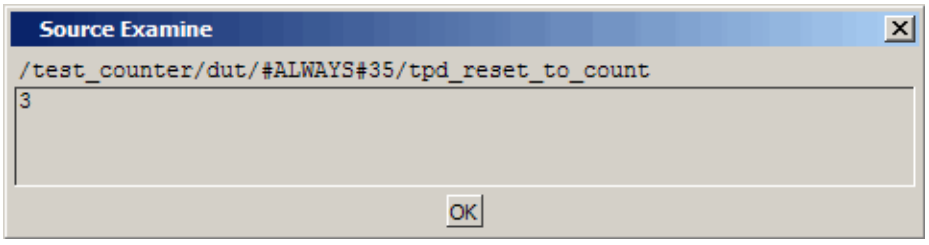
- Set your mouse pointer over a variable in the Source window and a yellow box will appear with the variable name and the value of that variable at the time of the selected cursor in the Wave window (Figure 3-14).

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



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

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



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

5. Try out the step commands.
   a. Click the Step Into icon on the Step toolbar.

This single-steps the debugger.

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

Lesson Wrap-Up

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

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

Chapter 4
Projects

In this lesson you will create a project.

Projects contain a work library and a session state that is stored in an *.mpf* file. Projects may also consist of:

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

Design Files for this Lesson

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

The pathnames are as follows:

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

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

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

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

Project Work Flow

Common tasks for creating and building a new project.

Create a New Project

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

Procedure

1. Create a new directory and copy the design files for this lesson into it.

   Start by creating a new directory for this exercise (in case other users will be working with these lessons).

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

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

2. If you just finished the previous lesson, ModelSim should already be running. If 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 box 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.

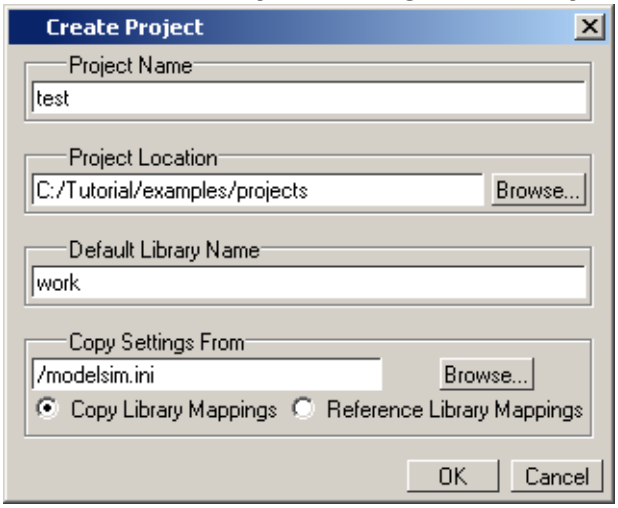
   b. Type **test** in the Project Name field.

   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 Box - Project Lab

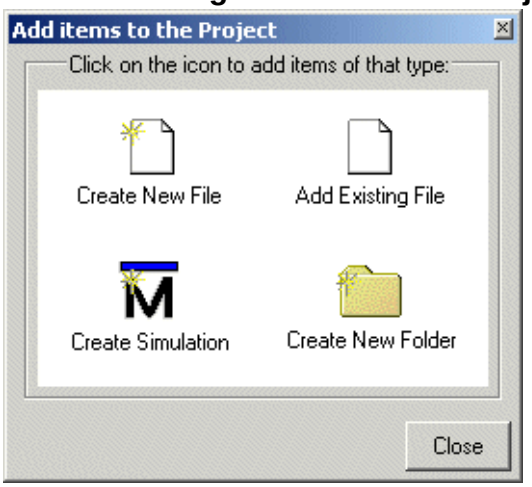


Add Objects to the Project

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

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

Figure 4-2. Adding New Items to a Project



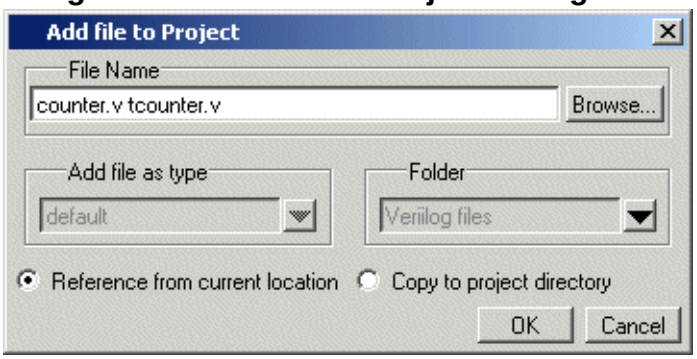
Procedure

Add two existing files.

a. Click **Add Existing File**.

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

Figure 4-3. Add file to Project Dialog Box



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

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

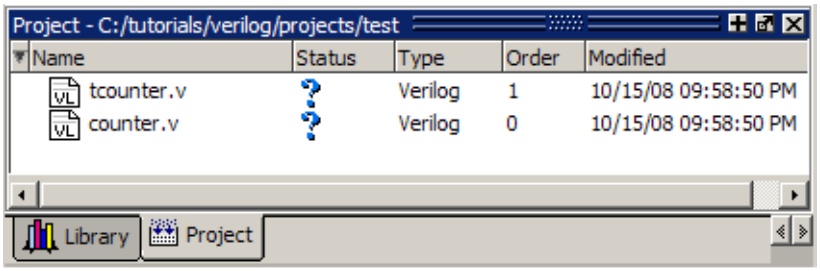
This closes the "Select files to add to project" dialog box and displays the selected files in the "Add file to Project" dialog box (Figure 4-3).

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

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

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

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



Changing Compile Order (VHDL)

By default ModelSim performs default binding of VHDL designs when you load the design with the **vsim** command. However, you can elect to perform default binding at compile time. If

you elect to do default binding at compile, then the compile order is important. Follow these steps to change compilation order within a project.

Procedure

Change the compile order.

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

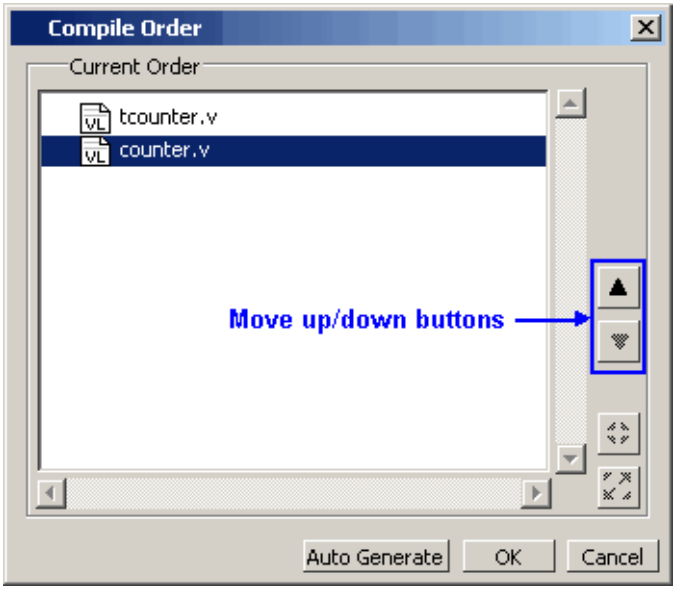
   This opens the Compile Order dialog box.

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

   ModelSim determines the compile order by making multiple passes over the files. It starts compiling from the top; if a file fails to compile due to dependencies, it moves that file to the bottom and then recompiles it after compiling the rest of the files. It continues in this manner until all files compile successfully or until a file(s) 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 (Figure 4-5).

Figure 4-5. Compile Order Dialog Box



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

Related Topics

Default Binding

Compile the Design

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

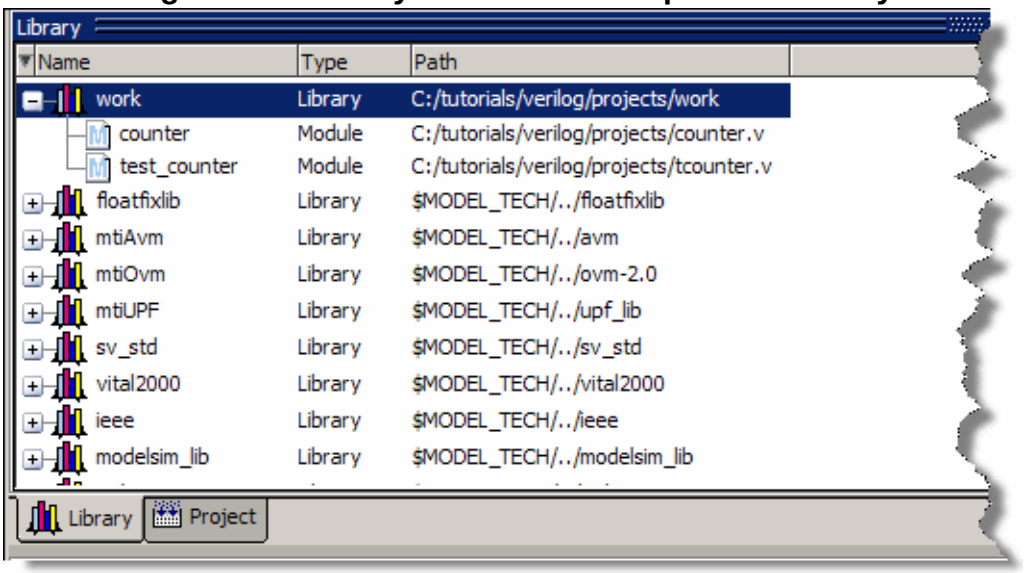
Procedure

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

      ModelSim compiles both files and changes the symbol in the Status column to a green check mark. A check mark means the compile succeeded. If compile fails, the symbol will be a red 'X', and you will see an error message in the Transcript window.
2. View the design units.
   a. Click the **Library** tab (Figure 4-6).
   b. Click the '+' icon next to the *work* library.

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

Figure 4-6. Library Window with Expanded Library



Load the Design

Now we're ready to load the design into the simulator.

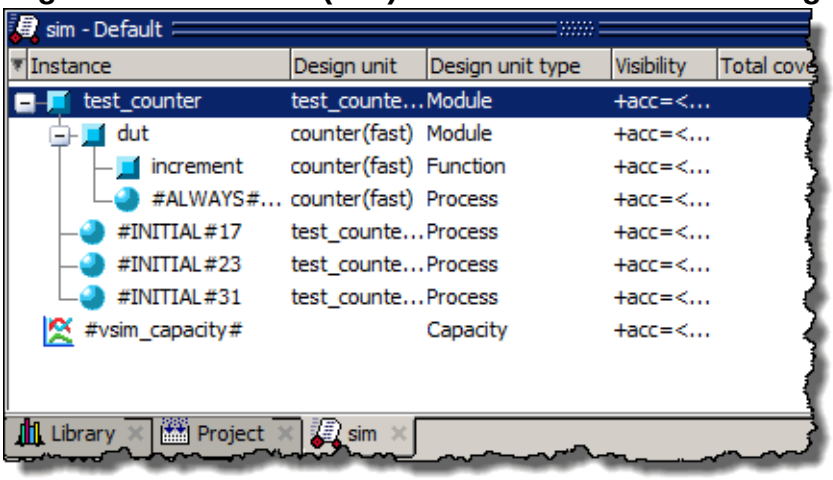
Procedure

1. Load the *test_counter* design unit.

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

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

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



   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.

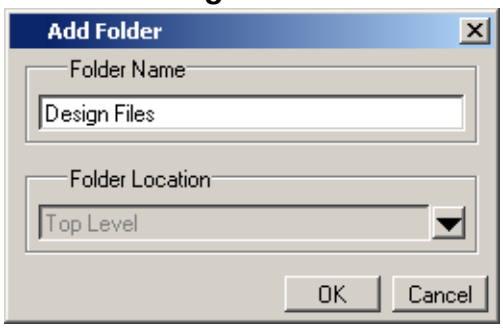
Adding Folders

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

Procedure

1. Add a new folder.
   a. Right-click in the Projects window and select **Add to Project > Folder**.
   b. Type **Design Files** in the **Folder Name** field (Figure 4-8).

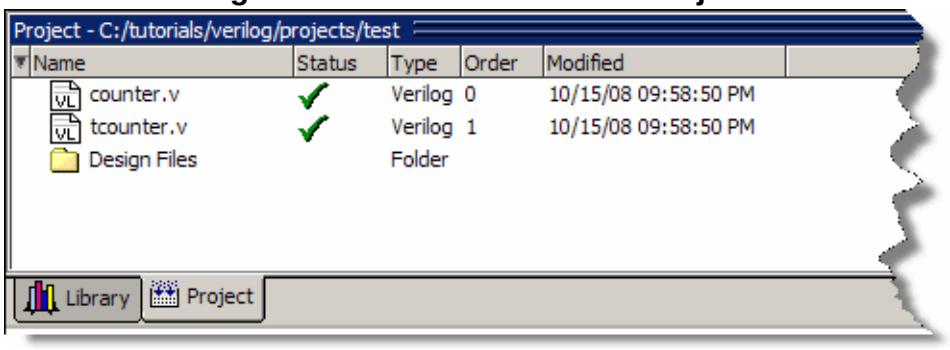
Figure 4-8. Adding New Folder to Project



   c. Click **OK**.

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

Figure 4-9. A Folder Within a Project

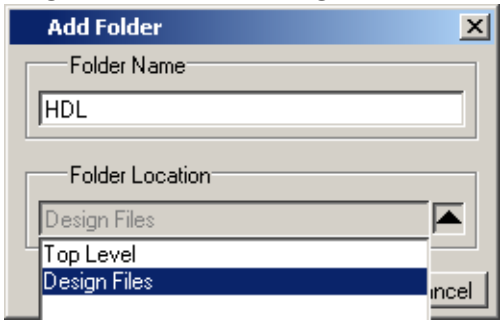


2. Add a sub-folder.

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

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

Figure 4-10. Creating Subfolder

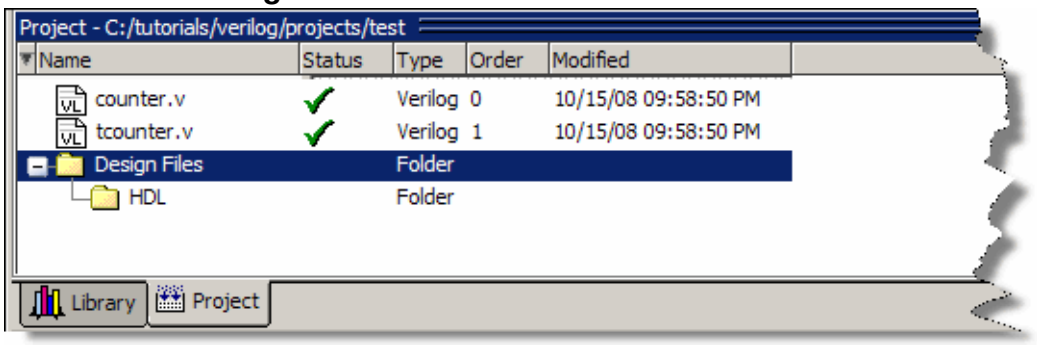


   c. Click the **Folder Location** drop-down arrow and select *Design Files*.

   d. Click **OK**.

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

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



   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 Project Compiler Settings dialog box.

Procedure

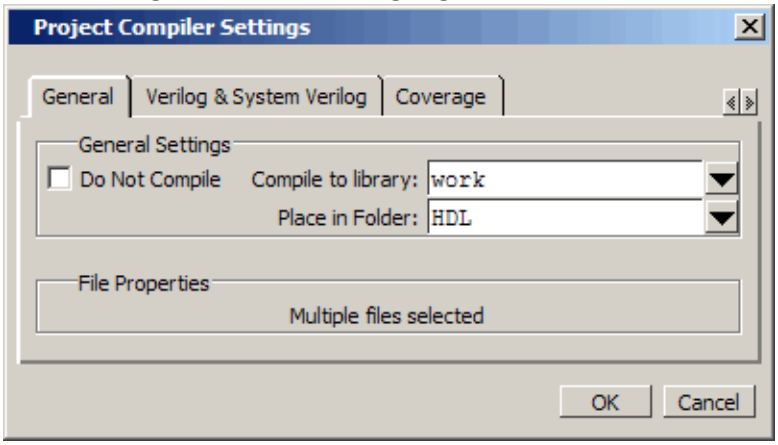
Move *tcounter.v* and *counter.v* to the *HDL* folder.

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

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

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

Figure 4-12. Changing File Location



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

d. Click **OK**.

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

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

Using Simulation Configurations

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

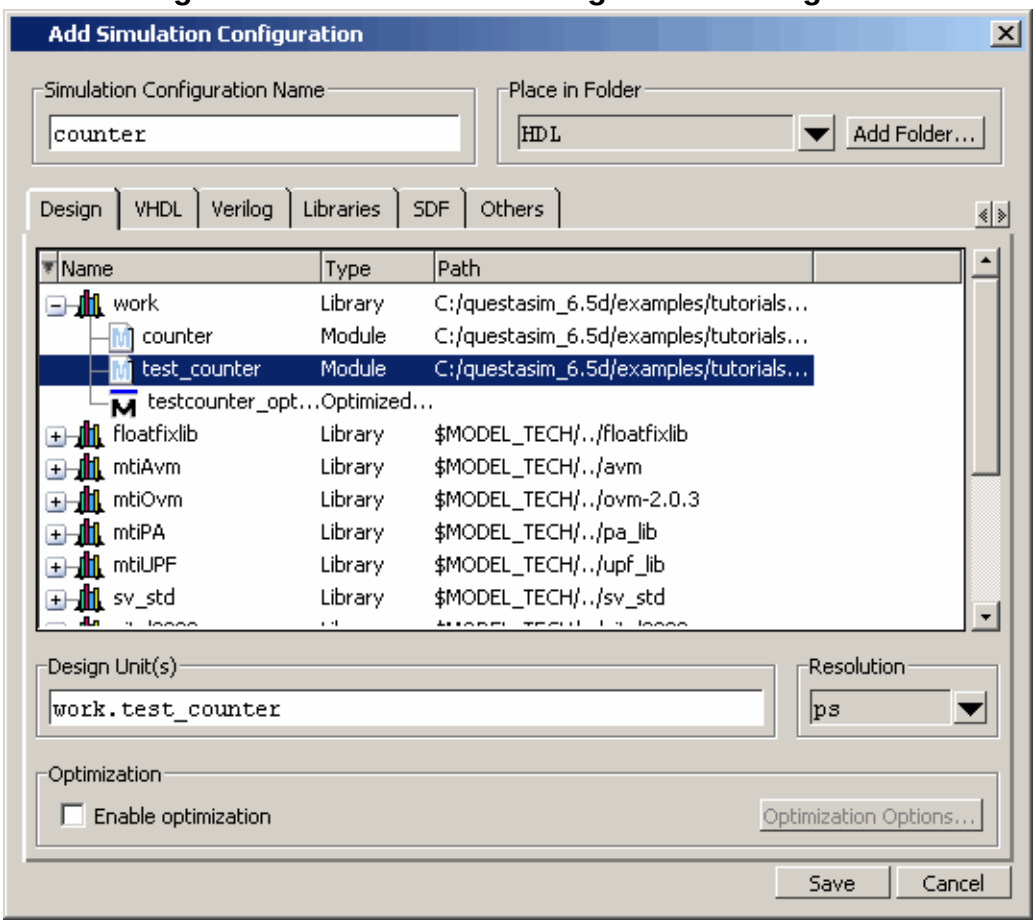
Procedure

1. Create a new Simulation Configuration.

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

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

Figure 4-13. Simulation Configuration Dialog Box



   b. Type **counter** in the **Simulation Configuration Name** field.

   c. Select *HDL* from the **Place in Folder** drop-down.

   d. Click the '+' icon next to the *work* library and select *test_counter*.

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

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

g. Click **Save**.

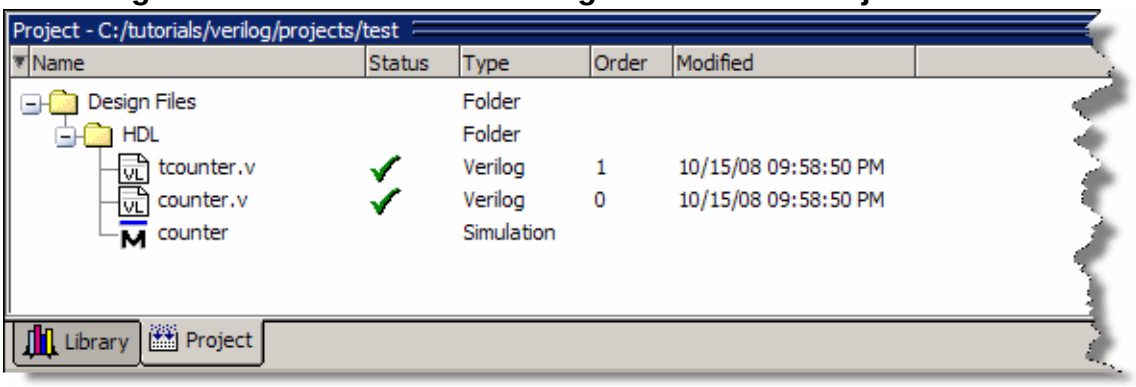
The files *tcounter.v* and *counter.v* show question mark icons in the status column because they have changed location since they were last compiled and need to be recompiled.

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

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

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

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

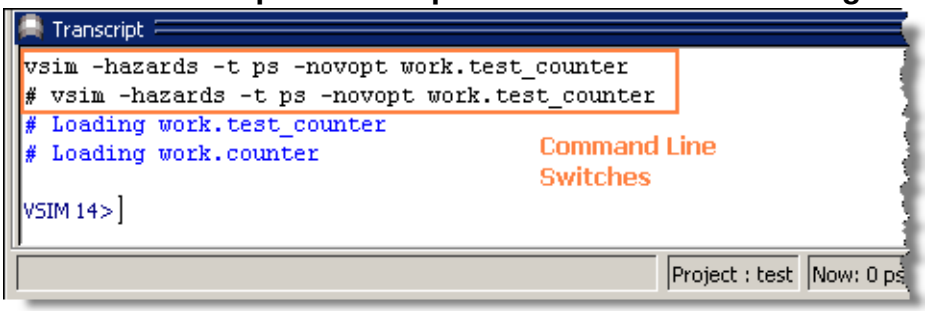


2. Load the Simulation Configuration.

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

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

Figure 4-15. Transcript Shows Options for Simulation Configurations



Lesson Wrap-Up

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

1. Select **Simulate > End Simulation**. Click Yes.
2. In the Project window, right-click and select **Close Project**.

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

Chapter 5
Working With Multiple Libraries

In this lesson you will practice working with multiple libraries. You might have multiple libraries to organize your design, to access IP from a third-party source, or to share common parts between simulations.

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

Design Files for this Lesson

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

The pathnames are as follows:

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

For further information, refer to the 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.

For additional information, see Permanently Mapping VHDL Resource Libraries.

Procedure

1. Create a directory for the resource library.

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

2. Create a directory for the test bench.

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

   You are creating two directories in this lesson to mimic the situation where you receive a resource library from a third-party. As noted earlier, we will link to the resource library in the first directory later in the lesson.

3. Start ModelSim and change to the resource_library directory.

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

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

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

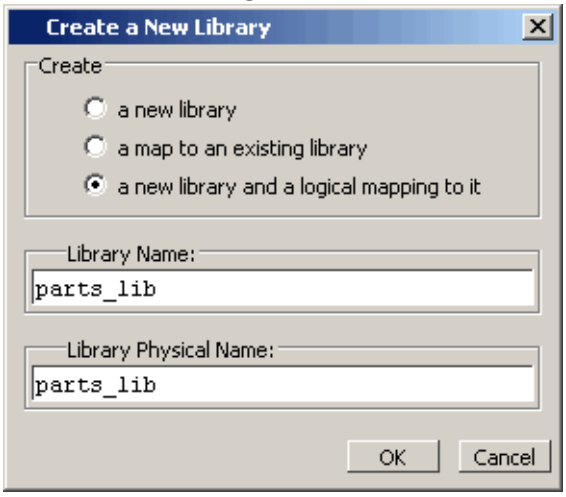
   b. Select **File > Change Directory** and change to the resource_library directory you created in step 1.

4. Create the resource library.

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

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

Figure 5-1. Creating New Resource Library



The Library Physical Name field is filled out automatically.

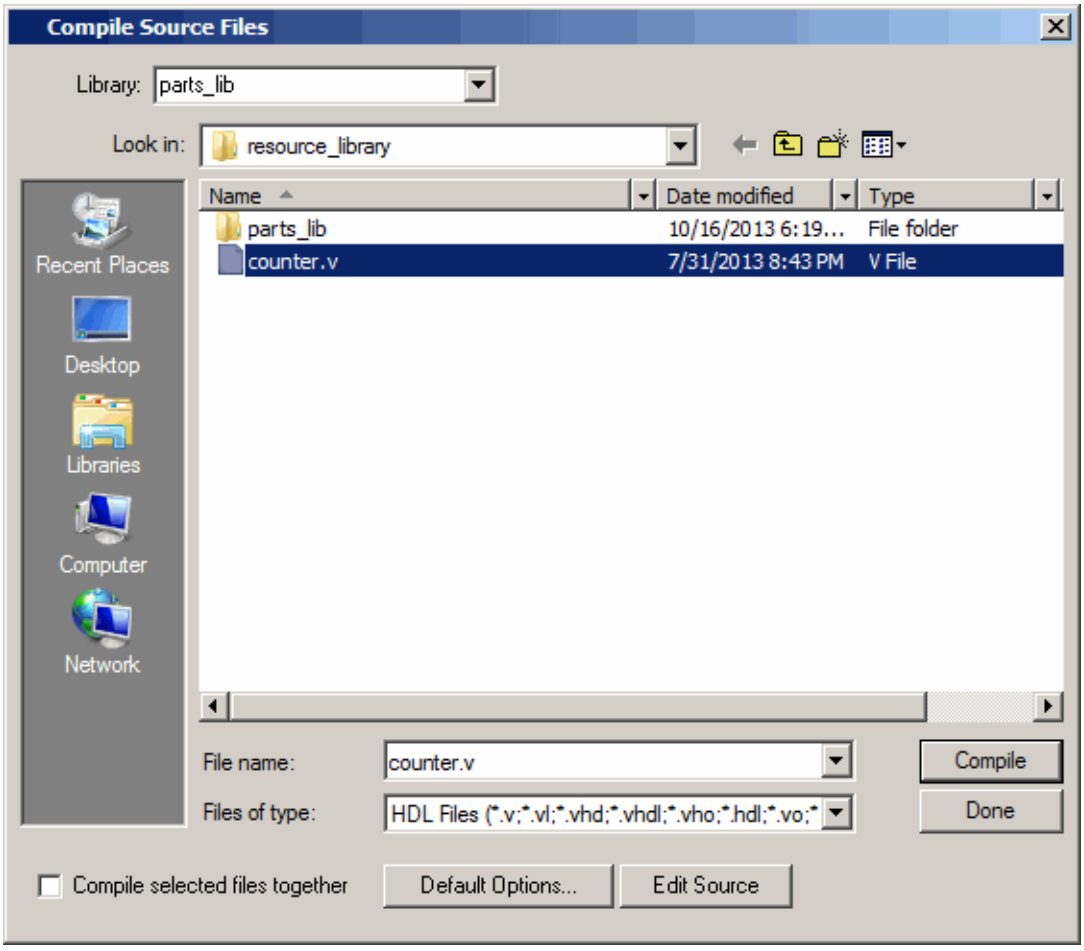
Once you click OK, ModelSim creates a directory for the library, lists it in the Library window, and modifies the *modelsim.ini* file to record this new library for the future.

5. Compile the counter into the resource library.

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

   b. Select the *parts_lib* library from the Library list (Figure 5-2).

Figure 5-2. Compiling into the Resource Library



   c. Double-click *counter.v* to compile it.

   d. Click **Done**.

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

6. Change to the testbench directory.

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

Creating the Project

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

Procedure

1. Create the project.
   a. Select **File > New > Project**.
   b. Type **counter** in the Project Name field.
   c. Do not change the Project Location field or the Default Library Name field. (The default library name is *work*.)
   d. Make sure "Copy Library Mappings" is selected. The default *modelsim.ini* file will be used.
   e. Click **OK**.
2. Add the test bench to the project.
   a. Click **Add Existing File** in the Add items to the Project dialog box.
   b. Click the **Browse** button and select *tcounter.v* in the "Select files to add to project" dialog box.
   c. Click **Open**.
   d. Click **OK**.
   e. Click **Close** to dismiss the "Add items to the Project" dialog box.

   The *tcounter.v* file is listed in the Project window.
3. Compile the test bench.
   a. Right-click *tcounter.v* and select **Compile > Compile Selected**.

Loading Without Linking Libraries

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

ModelSim responds differently for Verilog and VHDL in this situation.

Verilog

The following procedure is for those working with Verilog designs.

Load the Verilog Test Bench

Now we'll load the Verilog test bench into the simulator.

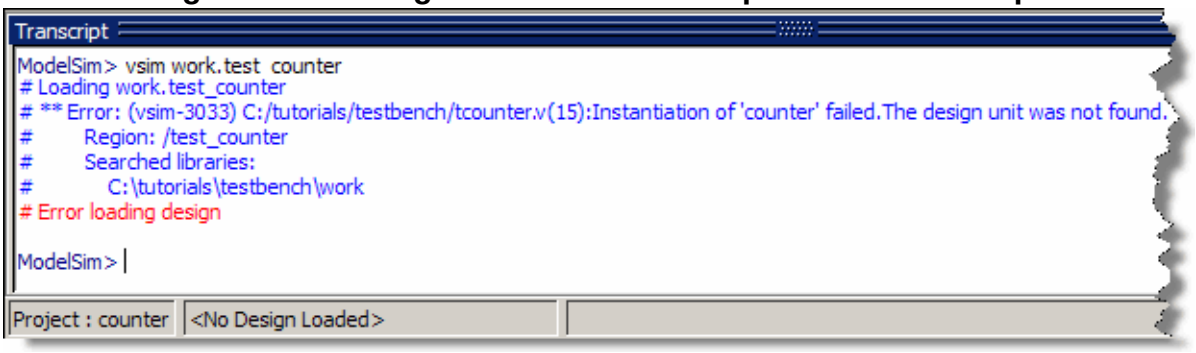
Procedure

Load a Verilog design with a missing resource library.

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

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

Figure 5-3. Verilog Simulation Error Reported in Transcript



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

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

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

VHDL

The following procedure is for those working with VHDL designs.

Load the VHDL Test Bench

Now we'll load the VHDL test bench into the simulator.

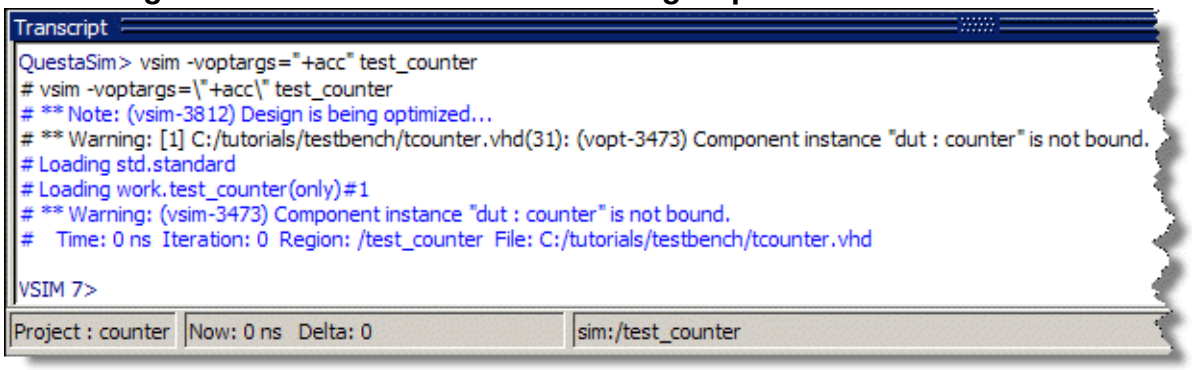
Procedure

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

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

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

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



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

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

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

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

Linking to the Resource Library

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

Procedure

Specify a search library during simulation.

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

b. Click the '+' icon next to the *work* library and select *test_counter*.

c. Click the Libraries tab.

d. Click the Add button next to the Search Libraries field and browse to *parts_lib* in the resource_library directory you created earlier in the lesson.

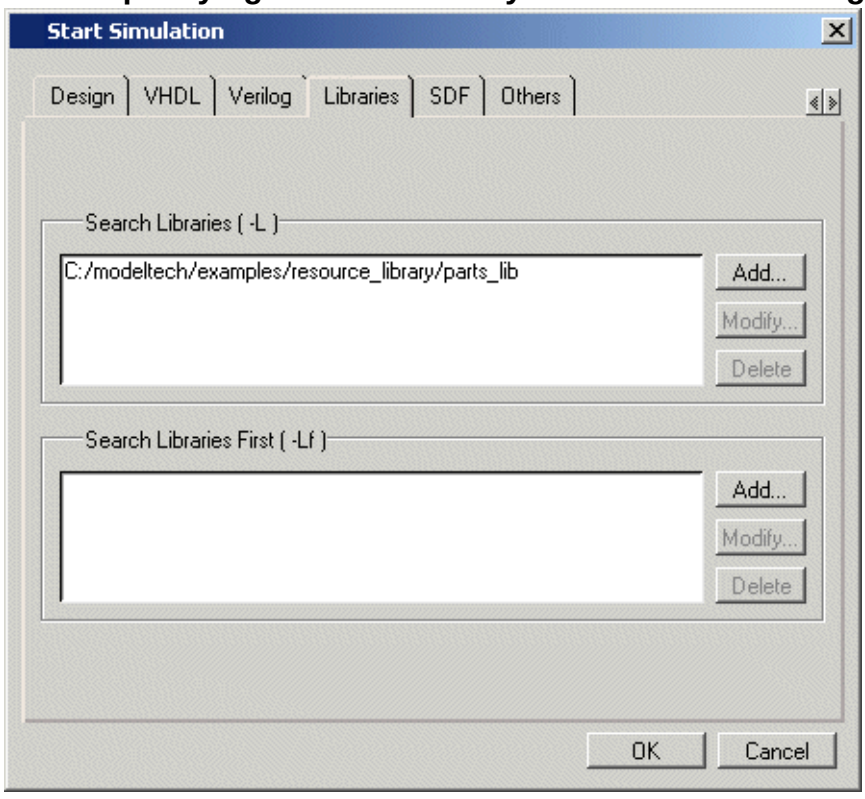
e. Click OK.

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

f. Click OK.

The design loads without errors.

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



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:

Procedure

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

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

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

Lesson Wrap-Up

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

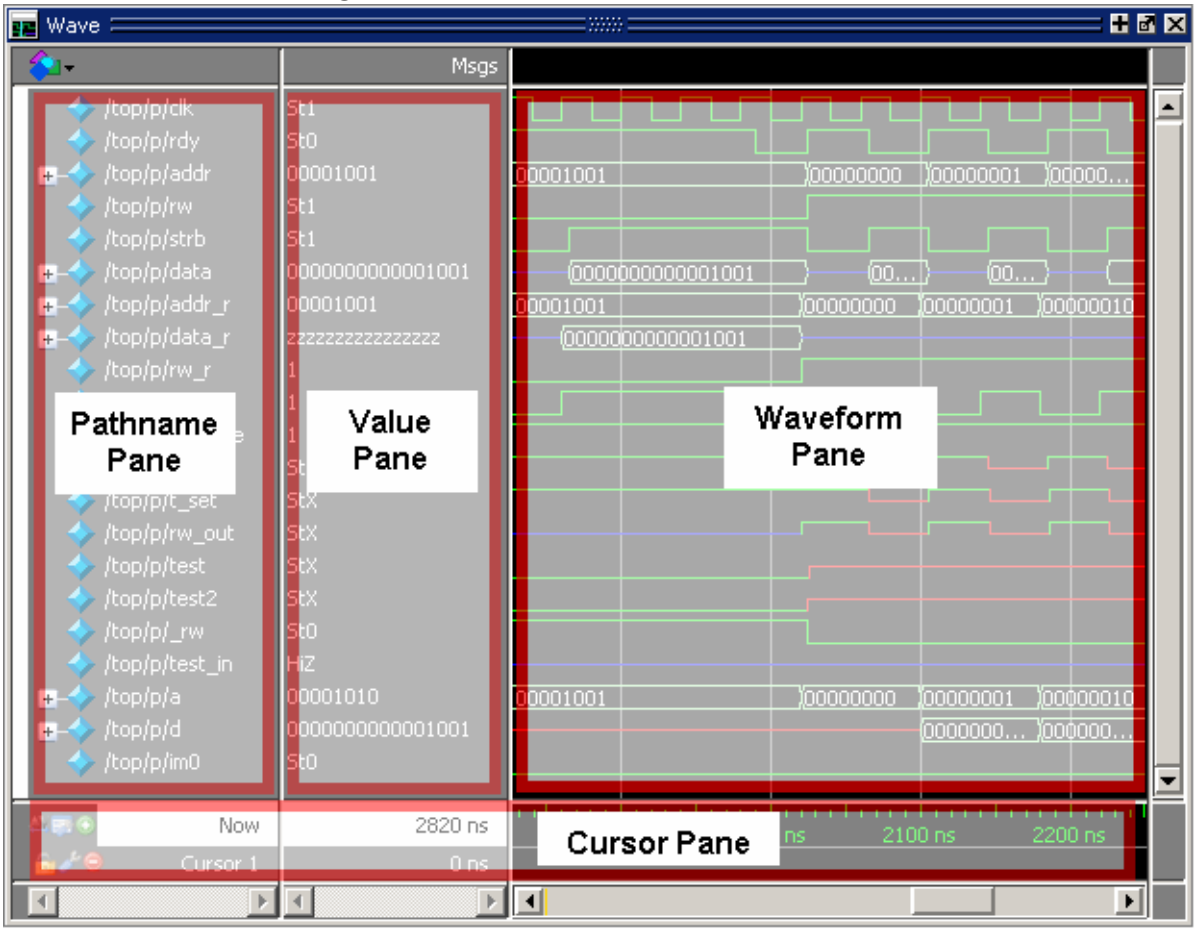
1. Select **Simulate > End Simulation**. Click Yes.
2. Select the Project window to make it active.
3. Select **File > Close**. Click **OK**.

Chapter 6
Analyzing Waveforms

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

The Wave window is divided into a number of panes (Figure 6-1). You can resize the pathnames pane, the values pane, and the waveform pane by clicking and dragging the bar between any two panes.

Figure 6-1. Panes of the Wave Window



Loading a Design

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

Procedure

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

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

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

2. Load the design.

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

   The *work* library should already exist.

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

   ModelSim loads the design and opens a Structure (sim) window.

Add Objects to the Wave Window

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

Procedure

1. Add objects from the Objects window.

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

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

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

2. Undock the Wave window.

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

a. Click the undock icon on the Wave window. 

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

3. Add objects using drag-and-drop.

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

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

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

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

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

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

4. Add objects using the add wave command.

a. Type the following at the VSIM> prompt.

```
add wave *
```

ModelSim adds all objects from the current selected region.

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

Zooming the Waveform Display

There are numerous methods for zooming the Waveform display. This exercise will show you how to zoom using various techniques.

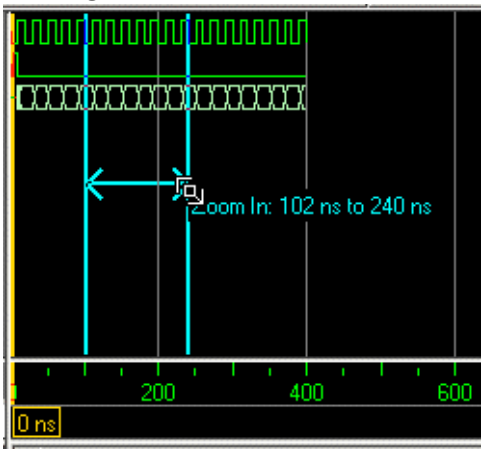
Procedure

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

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

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

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



2. Select **View > Zoom > Zoom Last**.
   a. The waveform display restores the previous display range.
3. Click the Zoom In icon a few times. 
4. In the waveform display, click and drag up and to the right.

   You should see a blue line and numbers defining an area to zoom out.
5. Select **View > Zoom > Zoom Full**.

Using Cursors in the Wave Window

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

You can also:

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

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

Working with a Single Cursor

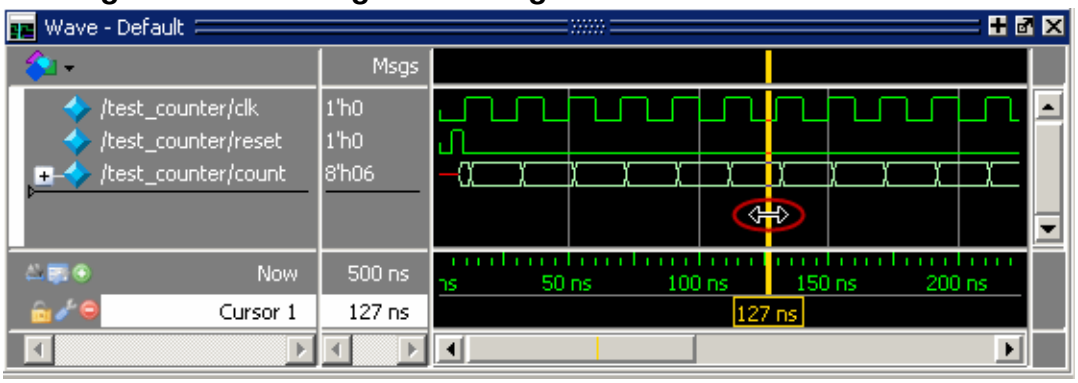
Let's look at the information provided when using a single cursor.

Procedure

1. Position the cursor by clicking in the cursor timeline then dragging.
   a. Click the Select Mode icon on the Wave window toolbar. 
   b. Click anywhere in the cursor timeline.

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

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



   c. Drag the cursor and observe the value pane.

The signal values change as you move the cursor. This is perhaps the easiest way to examine the value of a signal at a particular time.

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

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

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

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

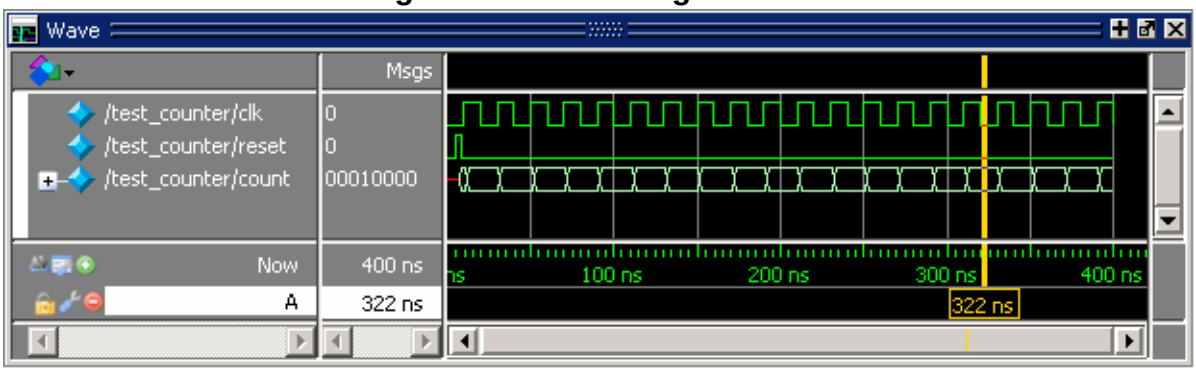
2. Rename the cursor.

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

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

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

Figure 6-4. Renaming a Cursor



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

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

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

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

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

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

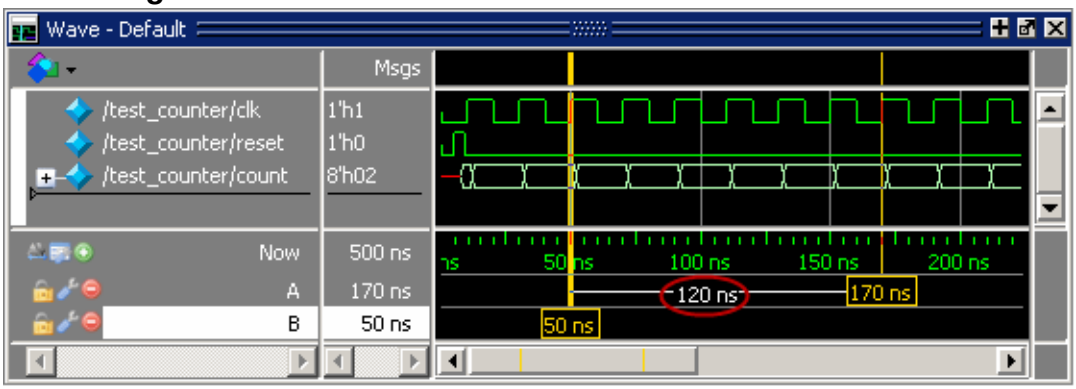
Working with Multiple Cursors

Even more information is available when working with multiple cursors.

Procedure

1. Add a second cursor.
   a. Click the Insert Cursor icon on the Wave window toolbar. 
   b. Right-click the name of the new cursor and delete the text.
   c. Type **B** and press Enter.
   d. Drag cursor *B* and watch the interval measurement change dynamically (Figure 6-5).

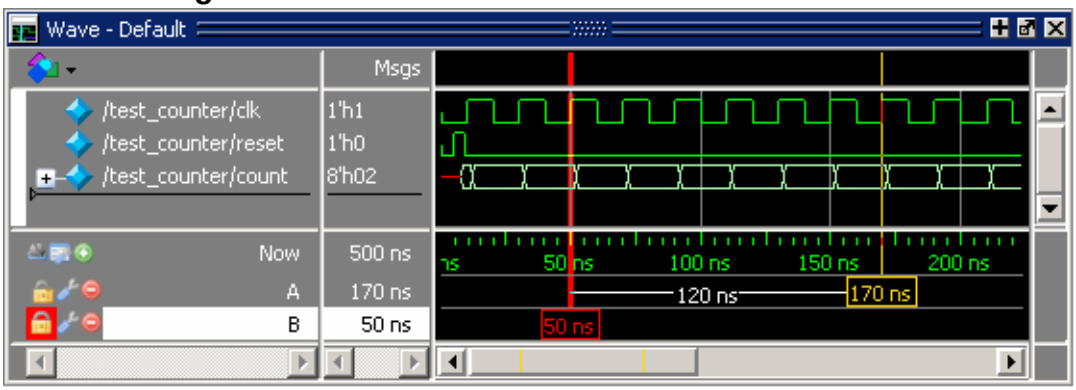
Figure 6-5. Interval Measurement Between Two Cursors



2. Lock cursor *B*.
   a. Right-click the yellow time indicator box associated with cursor *B* (at 56 ns).
   b. Select **Lock B** from the popup menu.

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

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



3. Delete cursor *B*.

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

Lesson Wrap-Up

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

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

Related Topics

Wave Window

Recording Simulation Results With Datasets

Chapter 7
Viewing And Initializing Memories

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

ModelSim defines and lists any of the following as memories:

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

Design Files for this Lesson

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

Example files are located in the following directories:

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

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

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

For further information, refer to the User's Manual Section: Memory List Window and the Reference Manual commands: mem display, mem load, mem save, and radix.

Compile and Load the Design

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

Procedure

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

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

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

2. Start ModelSim and change to the exercise directory.

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

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

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

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

3. Create the working library and compile the design.

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

   b. **Verilog**:
      Type **vlog *.v** at the ModelSim> prompt to compile all verilog files in the design.

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

4. Load the design.

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

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

View a Memory and its Contents

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

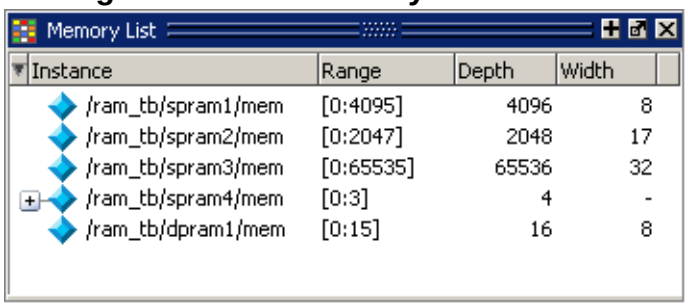
Procedure

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

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

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

Figure 7-1. The Memory List Window

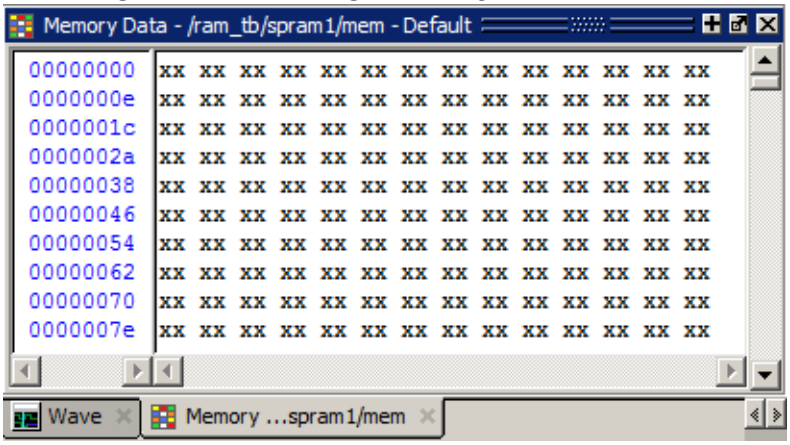


b. Double-click the */ram_tb/spram1/mem* instance in the memory list to view its contents.

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

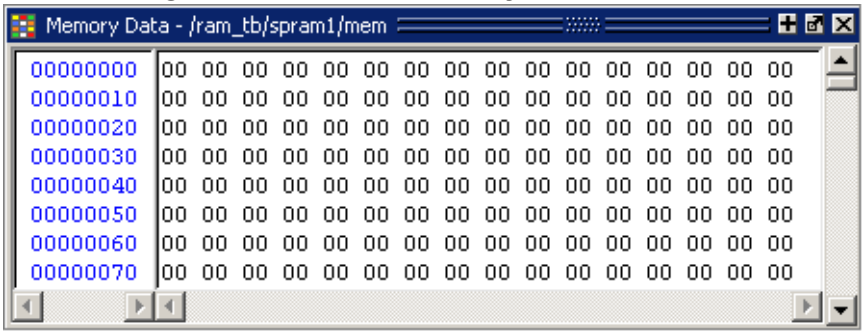
If you are using the Verilog example design, the data is all **X (**Figure 7-2) because you have not yet simulated the design.

Figure 7-2. Verilog Memory Data Window



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

Figure 7-3. VHDL Memory Data Window



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

2. Simulate the design.

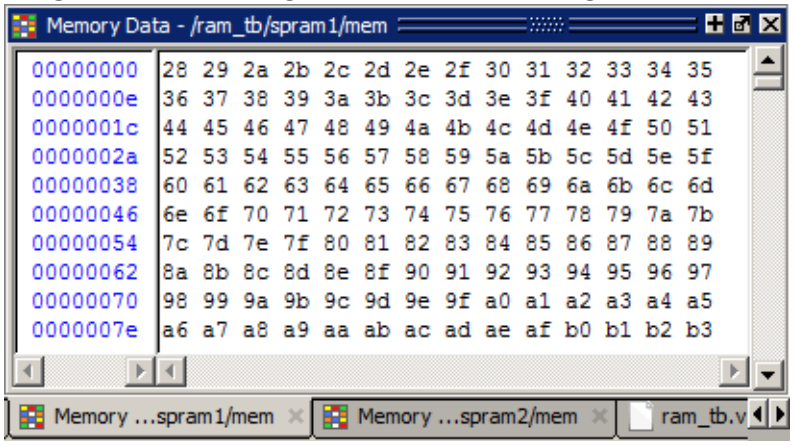
a. Click the **Run -All** icon in the Main window. 

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

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

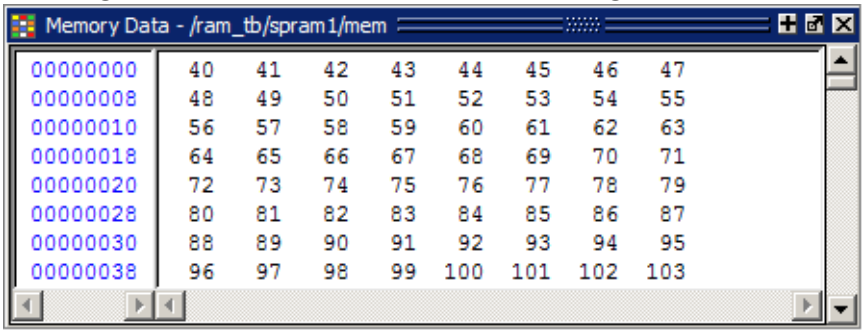
a. Click the **Memory ...spram1/mem** tab to bring that Memory data window to the foreground. The Verilog data fields are shown in Figure 7-4.

Figure 7-4. Verilog Data After Running Simulation



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

Figure 7-5. VHDL Data After Running Simulation

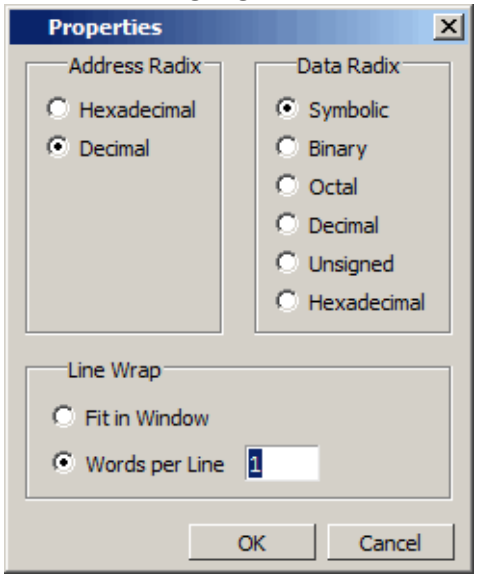


3. Change the address radix and the number of words per line for instance */ram_tb/spram1/mem*.

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

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

Figure 7-6. Changing the Address Radix



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

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

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

   f. Click OK.

You can see the Verilog results of the settings in Figure 7-7 and the VHDL results in Figure 7-8. If the figure 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 7-7. New Address Radix and Line Length (Verilog

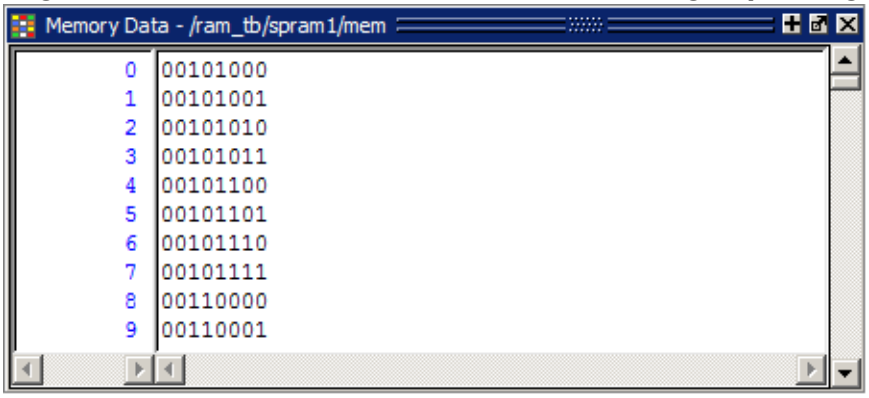
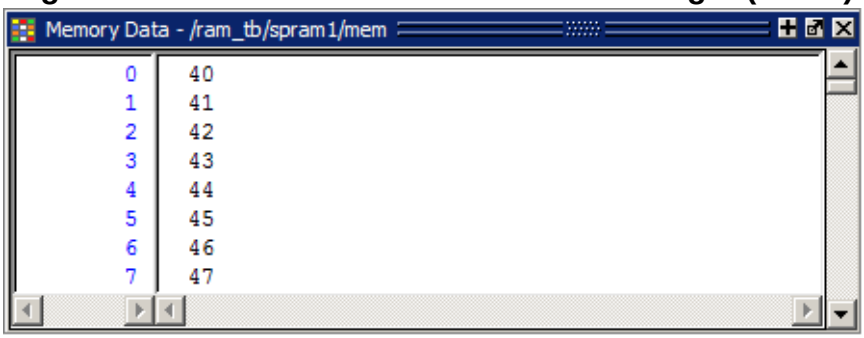


Figure 7-8. New Address Radix and Line Length (VHDL)



Navigate Within the Memory

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

Procedure

1. Use Goto to find a specific address.
   a. Right-click anywhere in address column and select Goto (Figure 7-9).

      The Goto dialog box opens in the data pane.

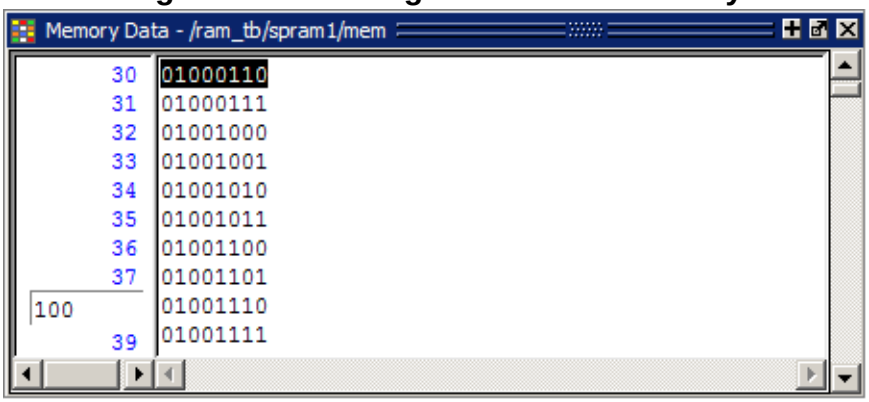
Figure 7-9. Goto Dialog Box

   b. Type **30** in the Goto Address field.
   c. Click **OK**.

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

2. Edit the address location directly.
   a. To quickly move to a particular address, do the following:
      i. Double click address 38 in the address column.
      ii. Enter address 100 (Figure 7-10).

Figure 7-10. Editing the Address Directly



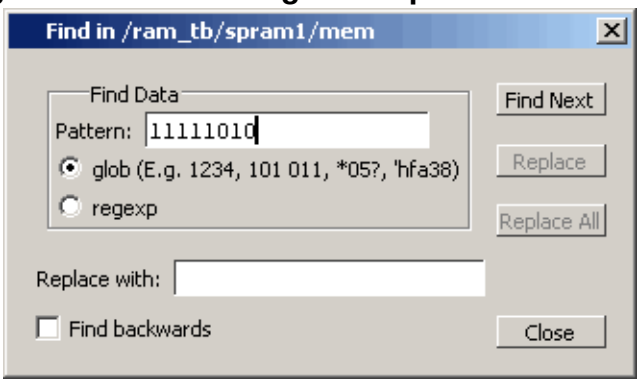
      iii. Press the Enter or Return key on your keyboard.

   The pane jumps to address 100.

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 7-11).

Figure 7-11. Searching for a Specific Data Value



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

      VHDL: Type **250** in the **Find data:** field and click **Find Next**.

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

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

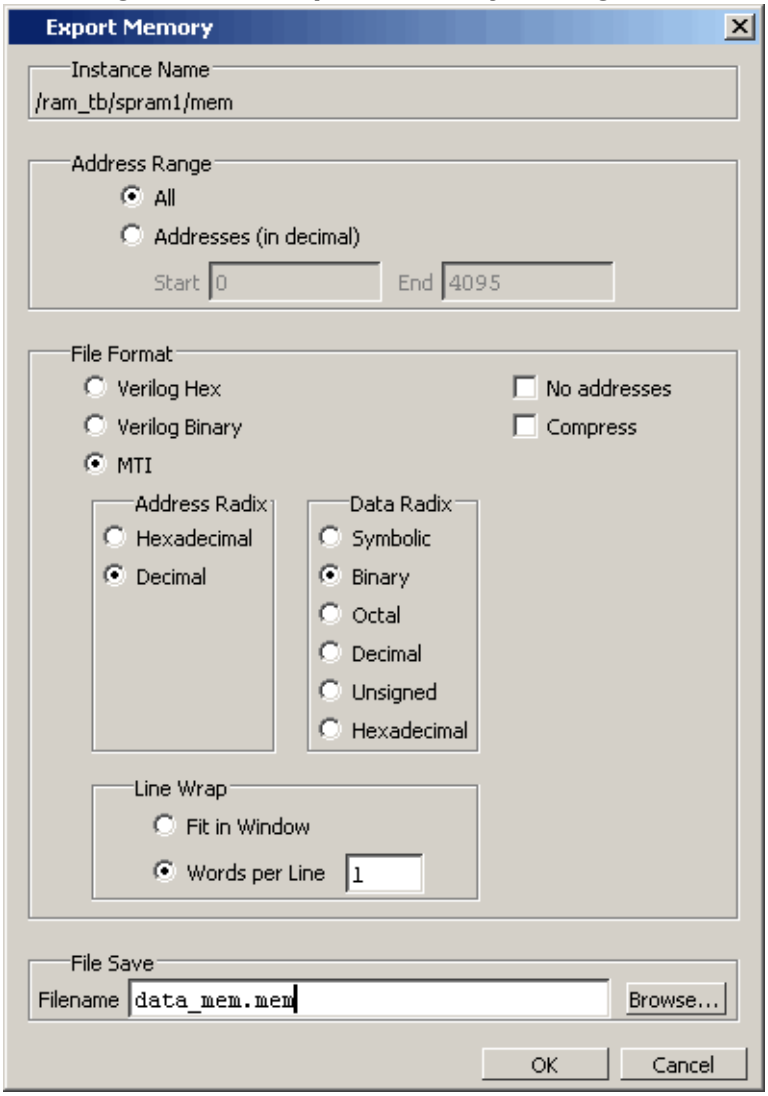
Export Memory Data to a File

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

Procedure

1. Export a memory pattern from the */ram_tb/spram1/mem* instance to a file.
   a. Make sure */ram_tb/spram1/mem* is open and selected.
   b. Select **File > Export > Memory Data** to bring up the Export Memory dialog box (Figure 7-12).

Figure 7-12. Export Memory Dialog Box



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

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

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

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

g. Click OK.

You can view the exported file in any editor.

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

2. Export a relocatable memory pattern file from the */ram_tb/spram2/mem* instance.

   a. Select the Memory Data window for the */ram_tb/spram2/mem* instance.

   b. Right-click on the memory contents to open a popup menu and select Properties.

   c. In the Properties dialog box, set the Address Radix to Decimal; the Data Radix to Binary; and the Line Wrap to 1 Words per Line. Click OK to accept the changes and close the dialog box.

   d. Select **File > Export > Memory Data** to bring up the Export Memory dialog box.

   e. For the Address Range, specify a Start address of **0** and End address of **250**.

   f. For the File Format, select MTI and **No addresses** to create a memory pattern that you can use to relocate somewhere else in the memory, or in another memory.

   g. For Address Radix select Decimal, and for Data Radix select Binary.

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

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

Initialize a Memory

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

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

Procedure

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

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

This will open a new Memory Data window to display the contents of */ram_tb/spram3/mem*. Familiarize yourself with the contents so you can identify changes once the initialization is complete.

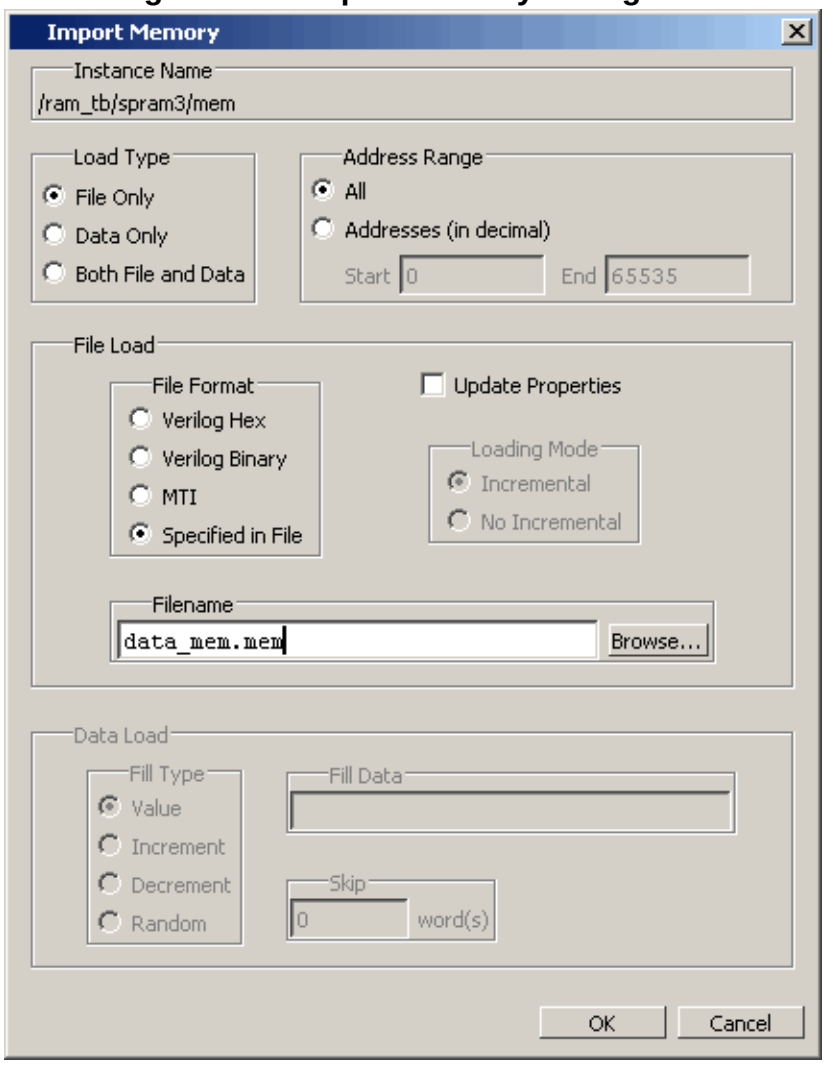
b. Right-click and select **Properties** to bring up the Properties dialog box.

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

2. Initialize *spram3* from a file.

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

Figure 7-13. Import Memory Dialog Box



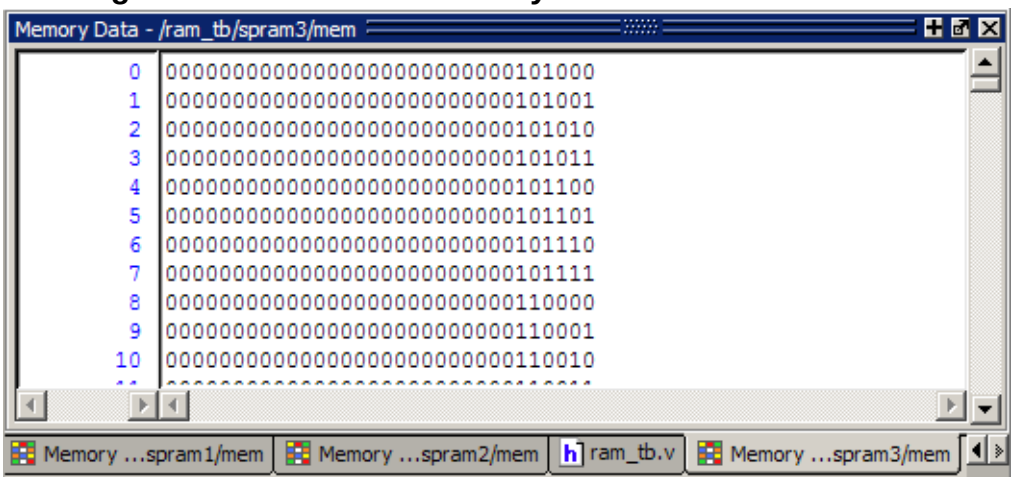
The default Load Type is File Only.

b. Type *data_mem.mem* in the Filename field.

c. Click **OK**.

The addresses in instance */ram_tb/spram3/mem* are updated with the data from *data_mem.mem* (Figure 7-14).

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



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

3. Import the */ram_tb/spram3/mem* instance with a relocatable memory pattern (*reloc.mem*) and a fill pattern.

   a. Right-click in the data column of *spram3* and select **Import Data Patterns** to bring up the Import Memory dialog box.

   b. For Load Type, select **Both File and Data**.

   c. For Address Range, select **Addresses** and enter **0** as the Start address and **300** as the End address.

      This means that you will be loading the file from 0 to 300. However, the *reloc.mem* file contains only 251 addresses of data. Addresses 251 to 300 will be loaded with the fill data you specify next.

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

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

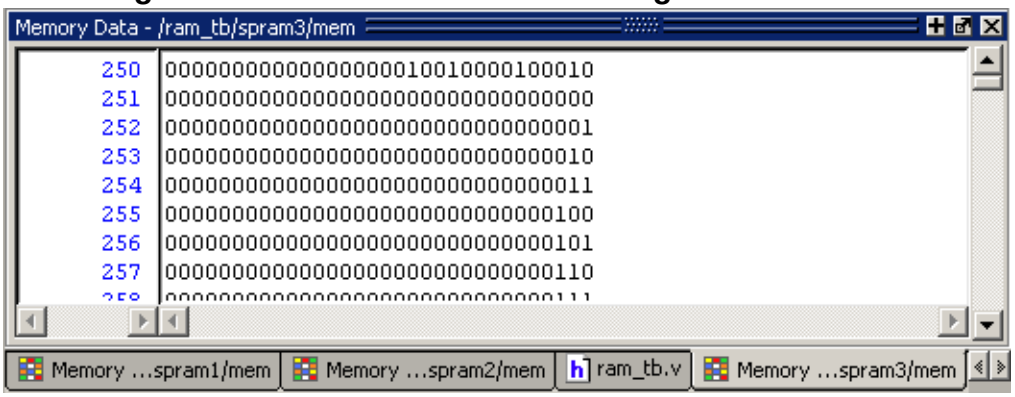
   f. In the Fill Data field, set the seed value of **0** for the incrementing data.

   g. Click **OK**.

   h. View the data near address 250 by double-clicking on any address in the Address column and entering **250**.

You can see the specified range of addresses overwritten with the new data. Also, you can see the incrementing data beginning at address 251 (Figure 7-15).

Figure 7-15. Data Increments Starting at Address 251



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

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

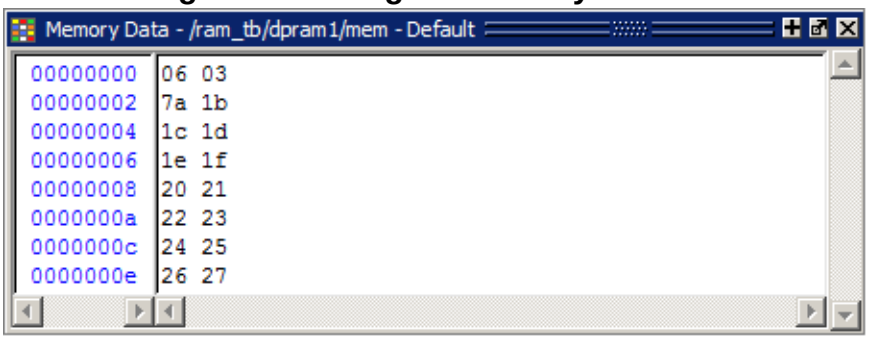
Interactive Debugging Commands

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

Procedure

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

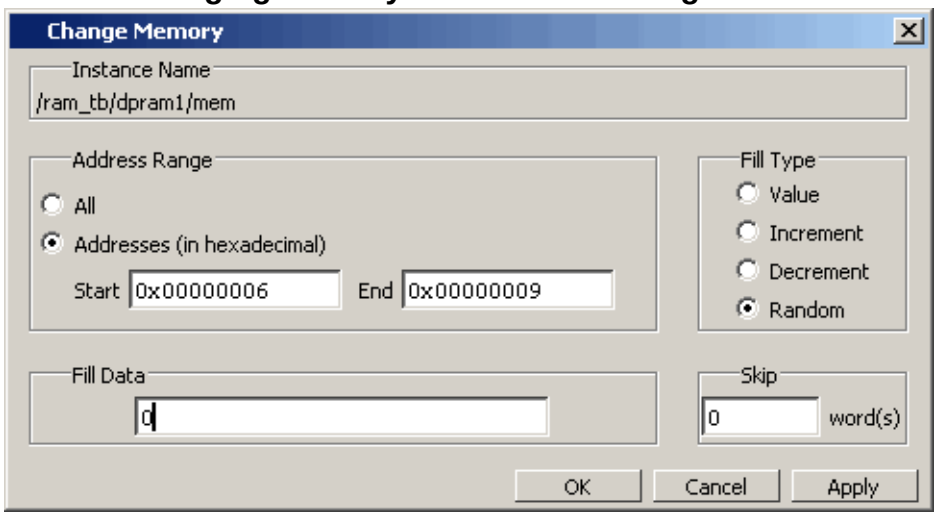
Figure 7-16. Original Memory Content



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

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

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



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

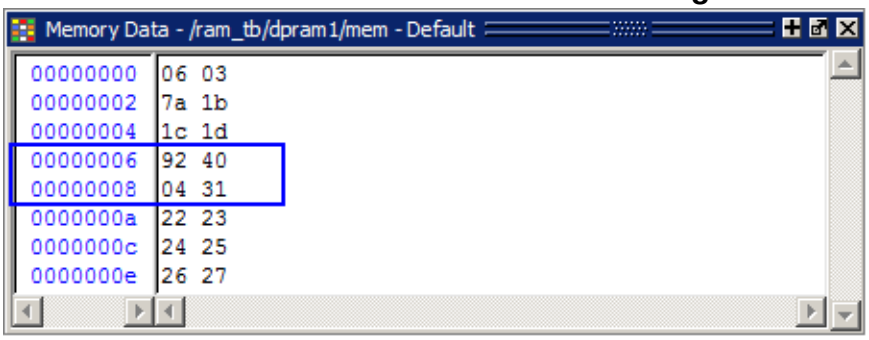
   c. Select **Random** as the **Fill Type**.

   d. Enter **0** as the **Fill Data**, setting the seed for the Random pattern.

   e. Click **OK**.

      The data in the specified range are replaced with a generated random fill pattern (Figure 7-18).

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

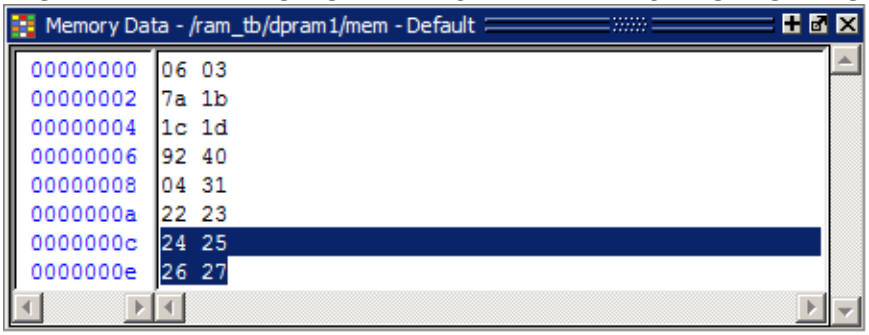


3. Change contents by highlighting.

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

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

Figure 7-19. Changing Memory Contents by Highlighting



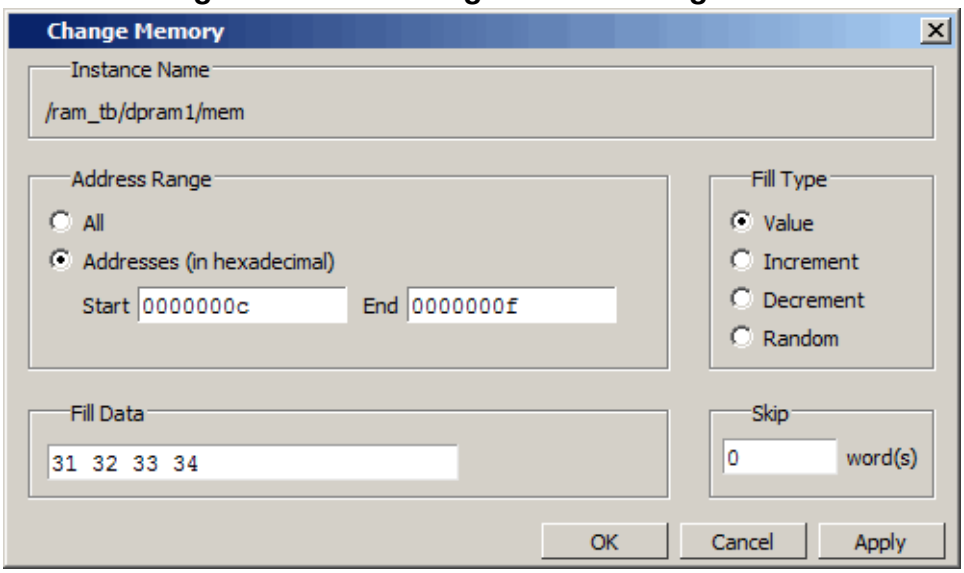
b. Right-click the highlighted data and select **Change**.

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

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

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

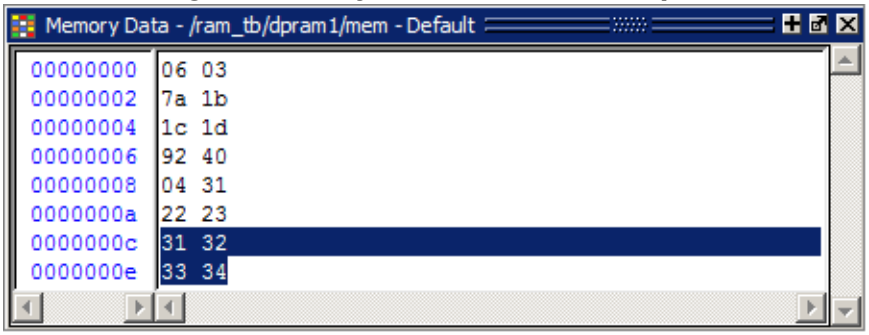
Figure 7-20. Entering Data to ChangeOK**



e. Click **OK**.

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

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



4. Edit data in place.

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

   a. Double click any value in the Data column.

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

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

Lesson Wrap-Up

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

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

Chapter 8
Automating Simulation

Aside from executing a couple of pre-existing DO files, the previous lessons focused on using ModelSim in interactive mode: executing single commands, one after another, 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*>/<*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.

Creating a Simple DO File

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

Refer to "Saving a Transcript File as a DO file."

Procedure

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

b. Enter the following commands into the Source window:

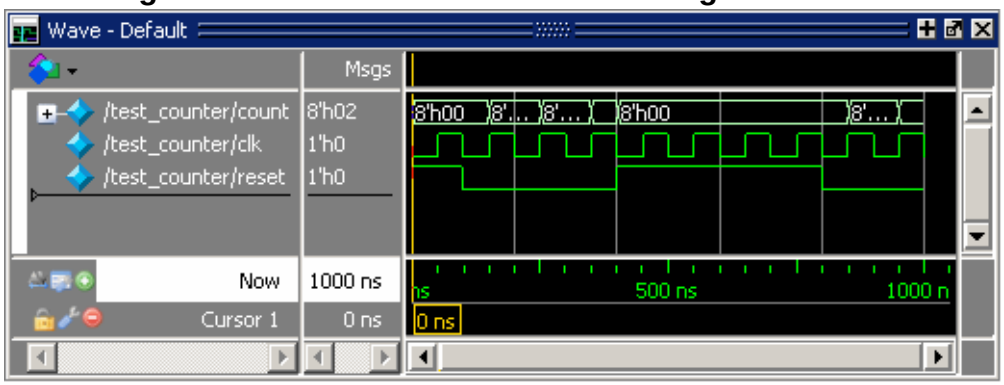
```
vsim test_counter

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

3. Save the file.
   a. Select **File > Save As**.
   b. Type **sim.do** in the File name: field and save it to the current directory.
4. Execute the DO file.
   a. Enter **do sim.do** at the VSIM> prompt.

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

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



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

Running in Command-Line Mode

We use the term “command-line mode” to refer to simulations that are run from a DOS/ UNIX prompt without invoking the GUI. Several ModelSim commands (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.

Procedure

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

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

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

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

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

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

   a. Type **vlib work** at the DOS/ UNIX prompt.

   b. For Verilog, type **vlog counter.v** at the DOS/ UNIX prompt. For VHDL, type **vcom counter.vhd**.

3. Create a DO file.

   a. Open a text editor.

   b. Type the following lines into a new file:

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

#change radix to symbolic
radix -symbolic

# read in stimulus
do stim.do

# output results
write list counter.lst

# quit the simulation
quit -f
```

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

4. Run the command line mode simulation.

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

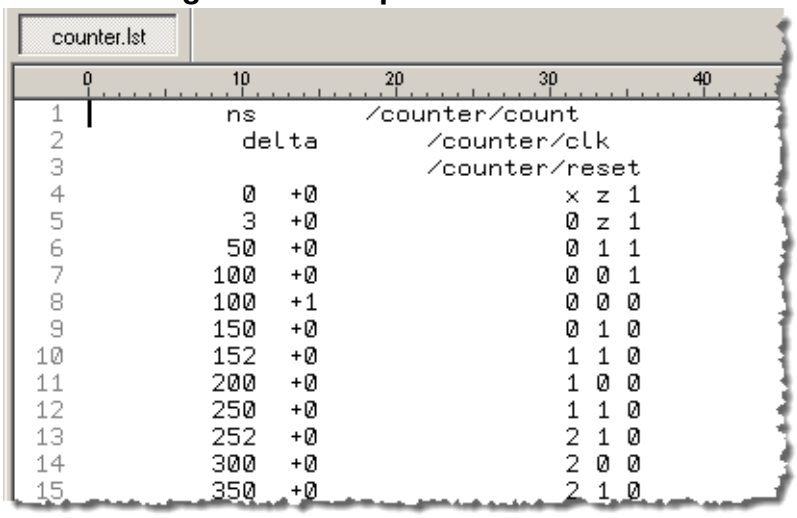
      vsim -c -do sim.do counter -wlf counter.wlf

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

5. View the list output.

   a. Open *counter.lst* and view the simulation results. Output produced by the Verilog version of the design should look like Figure 8-2:

Figure 8-2. Output of the Counter



   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.

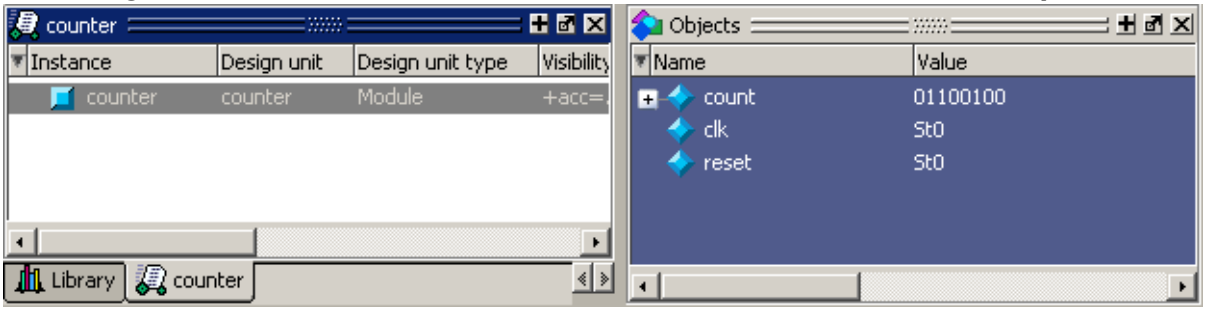
Note

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 prompt.

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

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



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

The waveforms display in the Wave window.

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

Using Tcl with the Simulator

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

Procedure

1. Create the script.

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

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

These commands do the following:

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

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

```
add wave -r /*
when {clk'event and clk="1"} {
   echo "Count is [exa count]"
   if {[examine count]== "8'h27"} {
      add_wave_zoom $now 1
   } elseif {[examine count]== "8'h47"} {
      add_wave_zoom $now 2
   }
}
```

These commands do the following:

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

c. Save the script with the name "*add_bkmrk.do*" into the directory you created in the Basic Simulation lesson.

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

a. Start ModelSim.

3. Execute the DO file and run the design.

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

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

The simulation runs and the DO file creates two bookmarks.

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

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

Lesson Wrap-Up

This concludes this lesson.

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

Related Topics

User's Manual Chapter: Tcl and DO Files.

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

Index

— Z —

End-User License Agreement

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

IMPORTANT INFORMATION

USE OF ALL SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE PRODUCTS. USE OF SOFTWARE INDICATES CUSTOMER'S 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 (as defined in Section 2) and hardware (collectively "Products") between the company acquiring the Products ("Customer"), and the Mentor Graphics entity that issued the corresponding quotation or, if no quotation was issued, the applicable local Mentor Graphics entity ("Mentor Graphics"). Except for license agreements related to the subject matter of this license agreement which are physically signed by Customer 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 Customer does not agree to these terms and conditions, promptly return or, in the case of Software 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. ORDERS, FEES AND PAYMENT.

1.1. To the extent Customer (or if agreed by Mentor Graphics, Customer's appointed third party buying agent) places and Mentor Graphics accepts purchase orders pursuant to this Agreement (each an "Order"), each Order will constitute a contract between Customer and Mentor Graphics, which shall be governed solely and exclusively by the terms and conditions of this Agreement, any applicable addenda and the applicable quotation, whether or not those documents are referenced on the Order. Any additional or conflicting terms and conditions appearing on an Order or presented in any electronic portal or automated order management system, whether or not required to be electronically accepted, will not be effective unless agreed in writing and physically signed by an authorized representative of Customer and Mentor Graphics.

1.2. Amounts invoiced will be paid, 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. Prices do not include freight, insurance, customs duties, taxes or other similar charges, which Mentor Graphics will state separately in the applicable invoice. Unless timely provided with a valid certificate of exemption or other evidence that items are not taxable, Mentor Graphics will invoice Customer for all applicable taxes including, but not limited to, VAT, GST, sales tax, consumption tax and service tax. Customer will make all payments free and clear of, and without reduction for, any withholding or other taxes; any such taxes imposed on payments by Customer hereunder will be Customer's sole responsibility. If Customer appoints a third party to place purchase orders and/or make payments on Customer's behalf, Customer shall be liable for payment under Orders placed by such third party in the event of default.

1.3. All Products are delivered FCA factory (Incoterms 2010), freight prepaid and invoiced to Customer, except Software delivered electronically, which shall be deemed delivered when made available to Customer for download. Mentor Graphics retains a security interest in all Products delivered under this Agreement, to secure payment of the purchase price of such Products, and Customer agrees to sign any documents that Mentor Graphics determines to be necessary or convenient for use in filing or perfecting such security interest. Mentor Graphics' delivery of Software by electronic means is subject to Customer's provision of both a primary and an alternate e-mail address.

2. **GRANT OF LICENSE.** The software installed, downloaded, or otherwise acquired by Customer under this Agreement, including any updates, modifications, revisions, copies, documentation, setup files 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. Except for Software that is embeddable ("Embedded Software"), which is licensed pursuant to separate embedded software terms or an embedded software supplement, Mentor Graphics grants to Customer, subject to payment of applicable license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form (except as provided in Subsection 4.2); (b) for Customer's internal business purposes; (c) for the term of the license; 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. Customer may have Software temporarily used by an employee for telecommuting purposes from locations other than a Customer office, such as the employee's residence, an airport or hotel, provided that such employee's primary place of employment is the site where the Software is authorized for use. 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. For the avoidance of doubt, if Customer provides any feedback or requests any change or enhancement to Products, whether in the course of receiving support or consulting services, evaluating Products, performing beta testing or otherwise, any inventions, product improvements, modifications or developments made by Mentor Graphics (at Mentor Graphics' sole discretion) will be the exclusive property of Mentor Graphics.

3. **BETA CODE.**

3.1. Portions or all of certain Software may contain code for experimental testing and evaluation (which may be either alpha or beta, collectively "Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to Customer 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. Mentor Graphics may choose, at its sole discretion, not to release Beta Code commercially in any form.

3.2. If Mentor Graphics authorizes Customer to use the Beta Code, Customer agrees to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. Customer will contact Mentor Graphics periodically during Customer's use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of Customer's evaluation and testing, Customer will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements.

3.3. Customer agrees to maintain Beta Code in confidence and shall restrict access to the Beta Code, including the methods and concepts utilized therein, solely to those employees and Customer location(s) authorized by Mentor Graphics to perform beta testing. Customer agrees 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 Customer's 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 Subsection 3.3 shall survive termination of this Agreement.

4. **RESTRICTIONS ON USE.**

4.1. Customer 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. Except for Embedded Software that has been embedded in executable code form in Customer's product(s), Customer 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. Customer shall not make Products available in any form to any person other than Customer's employees and on-site contractors, excluding Mentor Graphics competitors, whose job performance requires access and who are under obligations of confidentiality. Customer shall take appropriate action to protect the confidentiality of Products and ensure that any person permitted access does not disclose or use Products except as permitted by this Agreement. Customer shall give Mentor Graphics written notice of any unauthorized disclosure or use of the Products as soon as Customer becomes aware of such unauthorized disclosure or use. Customer acknowledges that Software provided hereunder may contain source code which is proprietary and its confidentiality is of the highest importance and value to Mentor Graphics. Customer acknowledges that Mentor Graphics may be seriously harmed if such source code is disclosed in violation of this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, Customer shall not reverse-assemble, disassemble, reverse-compile, or reverse-engineer any Product, or in any way derive any source code from Software that is not provided to Customer in source code form. Log files, data files, rule files and script files generated by or for the Software (collectively "Files"), including without limitation files containing Standard Verification Rule Format ("SVRF") and Tcl Verification Format ("TVF") which are Mentor Graphics' trade secret and proprietary syntaxes for expressing process rules, constitute or include confidential information of Mentor Graphics. Customer may share Files with third parties, excluding Mentor Graphics competitors, provided that the confidentiality of such Files is protected by written agreement at least as well as Customer protects other information of a similar nature or importance, but in any case with at least reasonable care. Customer may use Files containing SVRF or TVF only with Mentor Graphics products. Under no circumstances shall Customer use Products or Files or allow their use for the purpose of developing, enhancing or marketing any product that is in any way competitive with Products, or disclose to any third party the results of, or information pertaining to, any benchmark.

4.2. If any Software or portions thereof are provided in source code form, Customer will use the source code only to correct software errors and enhance or modify the Software for the authorized use, or as permitted for Embedded Software under separate embedded software terms or an embedded software supplement. Customer shall not disclose or permit disclosure of source code, in whole or in part, including any of its methods or concepts, to anyone except Customer's employees or on-site contractors, excluding Mentor Graphics competitors, with a need to know. Customer shall not copy or compile source code in any manner except to support this authorized use.

4.3. Customer agrees that it will not subject any Product to any open source software ("OSS") license that conflicts with this Agreement or that does not otherwise apply to such Product.

4.4. Customer may not assign this Agreement or the rights and duties under it, or relocate, sublicense, or otherwise transfer the Products, whether by operation of law or otherwise ("Attempted Transfer"), without Mentor Graphics' prior written consent and payment of Mentor Graphics' then-current applicable relocation and/or transfer fees. 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/or the licenses granted under this Agreement. The terms of this Agreement, including without limitation the licensing and assignment provisions, shall be binding upon Customer's permitted successors in interest and assigns.

4.5. The provisions of this Section 4 shall survive the termination of this Agreement.

5. **SUPPORT SERVICES.** To the extent Customer purchases support services, Mentor Graphics will provide Customer with updates and technical support for the Products, at the Customer site(s) for which support is purchased, in accordance with Mentor Graphics' then current End-User Support Terms located at http://supportnet.mentor.com/supportterms.

6. **OPEN SOURCE SOFTWARE.** Products may contain OSS or code distributed under a proprietary third party license agreement, to which additional rights or obligations ("Third Party Terms") may apply. Please see the applicable Product documentation (including license files, header files, read-me files or source code) for details. In the event of conflict between the terms of this Agreement

(including any addenda) and the Third Party Terms, the Third Party Terms will control solely with respect to the OSS or third party code. The provisions of this Section 6 shall survive the termination of this Agreement.

7. **LIMITED WARRANTY.**

7.1. Mentor Graphics warrants that during the warranty period its standard, generally supported Products, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Products will meet Customer's requirements or that operation of Products 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. Customer must notify Mentor Graphics in writing of any nonconformity within the warranty period. For the avoidance of doubt, this warranty applies only to the initial shipment of Software under an Order and does not renew or reset, for example, with the delivery of (a) Software updates or (b) authorization codes or alternate Software under a transaction involving Software re-mix. This warranty shall not be valid if Products have been subject to misuse, unauthorized modification, improper installation or Customer is not in compliance with this Agreement. MENTOR GRAPHICS' ENTIRE LIABILITY AND CUSTOMER'S EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF THE PRODUCTS TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF THE PRODUCTS THAT DO NOT MEET THIS LIMITED WARRANTY. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) PRODUCTS PROVIDED AT NO CHARGE; OR (C) BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

7.2. THE WARRANTIES SET FORTH IN THIS SECTION 7 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO PRODUCTS PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

8. **LIMITATION OF LIABILITY.** TO THE EXTENT PERMITTED 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 RECEIVED FROM CUSTOMER FOR THE HARDWARE, SOFTWARE LICENSE 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 8 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT.

9. **THIRD PARTY CLAIMS.**

9.1. Customer acknowledges that Mentor Graphics has no control over the testing of Customer's products, or the specific applications and use of Products. Mentor Graphics and its licensors shall not be liable for any claim or demand made against Customer by any third party, except to the extent such claim is covered under Section 10.

9.2. In the event that a third party makes a claim against Mentor Graphics arising out of the use of Customer's products, Mentor Graphics will give Customer prompt notice of such claim. At Customer's option and expense, Customer may take sole control of the defense and any settlement of such claim. Customer WILL reimburse and hold harmless Mentor Graphics for any LIABILITY, damages, settlement amounts, costs and expenses, including reasonable attorney's fees, incurred by or awarded against Mentor Graphics or its licensors in connection with such claims.

9.3. The provisions of this Section 9 shall survive any expiration or termination of this Agreement.

10. **INFRINGEMENT.**

10.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against Customer in the United States, Canada, Japan, or member state of the European Union which alleges that any standard, generally supported Product acquired by Customer hereunder infringes a patent or copyright or misappropriates a trade secret in such jurisdiction. Mentor Graphics will pay costs and damages finally awarded against Customer that are attributable to such action. Customer understands and agrees that as conditions to Mentor Graphics' obligations under this section Customer must: (a) notify Mentor Graphics promptly in writing of the action; (b) provide Mentor Graphics all reasonable information and assistance to settle or defend the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

10.2. If a claim is made under Subsection 10.1 Mentor Graphics may, at its option and expense: (a) replace or modify the Product so that it becomes noninfringing; (b) procure for Customer the right to continue using the Product; or (c) require the return of the Product and refund to Customer any purchase price or license fee paid, less a reasonable allowance for use.

10.3. Mentor Graphics has no liability to Customer if the action is based upon: (a) the combination of Software or hardware with any product not furnished by Mentor Graphics; (b) the modification of the Product other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of the Product as part of an infringing process; (e) a product that Customer makes, uses, or sells; (f) any Beta Code or Product provided at no charge; (g) any software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; (h) OSS, except to the extent that the infringement is directly caused by Mentor Graphics' modifications to such OSS; or (i) infringement by Customer that is deemed willful. In the case of (i), Customer shall reimburse Mentor Graphics for its reasonable attorney fees and other costs related to the action.

10.4. THIS SECTION 10 IS SUBJECT TO SECTION 8 ABOVE AND STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS, AND CUSTOMER'S SOLE AND EXCLUSIVE REMEDY, FOR DEFENSE,

SETTLEMENT AND DAMAGES, WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY PRODUCT PROVIDED UNDER THIS AGREEMENT.

11. **TERMINATION AND EFFECT OF TERMINATION.**

    11.1. If a Software license was provided for limited term use, such license will automatically terminate at the end of the authorized term. Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement immediately upon written notice if Customer: (a) exceeds the scope of the license or otherwise fails to comply with the licensing or confidentiality provisions of this Agreement, or (b) becomes insolvent, files a bankruptcy petition, institutes proceedings for liquidation or winding up or enters into an agreement to assign its assets for the benefit of creditors. For any other material breach of any provision of this Agreement, Mentor Graphics may terminate this Agreement and/or any license granted under this Agreement upon 30 days written notice if Customer fails to cure the breach within the 30 day notice period. Termination of this Agreement or any license granted hereunder will not affect Customer's obligation to pay for Products shipped or licenses granted prior to the termination, which amounts shall be payable immediately upon the date of termination.

    11.2. Upon termination of this Agreement, the rights and obligations of the parties shall cease except as expressly set forth in this Agreement. Upon termination of this Agreement and/or any license granted under this Agreement, Customer shall ensure that all use of the affected Products ceases, and shall return hardware and either return to Mentor Graphics or destroy Software in Customer's possession, including all copies and documentation, and certify in writing to Mentor Graphics within ten business days of the termination date that Customer no longer possesses any of the affected Products or copies of Software in any form.

12. **EXPORT.** The Products provided hereunder are subject to regulation by local laws and European Union ("E.U.") and United States ("U.S.") government agencies, which prohibit export, re-export or diversion of certain products, information about the products, and direct or indirect products thereof, to certain countries and certain persons. Customer agrees that it will not export or re-export Products in any manner without first obtaining all necessary approval from appropriate local, E.U. and U.S. government agencies. If Customer wishes to disclose any information to Mentor Graphics that is subject to any E.U., U.S. or other applicable export restrictions, including without limitation the U.S. International Traffic in Arms Regulations (ITAR) or special controls under the Export Administration Regulations (EAR), Customer will notify Mentor Graphics personnel, in advance of each instance of disclosure, that such information is subject to such export restrictions.

13. **U.S. GOVERNMENT LICENSE RIGHTS.** Software was developed entirely at private expense. The parties agree that all Software is commercial computer software within the meaning of the applicable acquisition regulations. Accordingly, pursuant to U.S. FAR 48 CFR 12.212 and DFAR 48 CFR 227.7202, use, duplication and disclosure of the Software by or for the U.S. government or a U.S. government subcontractor is subject solely to the terms and conditions set forth in this Agreement, which shall supersede any conflicting terms or conditions in any government order document, except for provisions which are contrary to applicable mandatory federal laws.

14. **THIRD PARTY BENEFICIARY.** Mentor Graphics Corporation, Mentor Graphics (Ireland) Limited, Microsoft Corporation and other licensors may be third party beneficiaries of this Agreement with the right to enforce the obligations set forth herein.

15. **REVIEW OF LICENSE USAGE.** Customer will monitor the access to and use of Software. With prior written notice and during Customer's normal business hours, Mentor Graphics may engage an internationally recognized accounting firm to review Customer's software monitoring system and records deemed relevant by the internationally recognized accounting firm to confirm Customer's compliance with the terms of this Agreement or U.S. or other local export laws. Such review may include FlexNet (or successor product) report log files that Customer shall capture and provide at Mentor Graphics' request. Customer shall make records available in electronic format and shall fully cooperate with data gathering to support the license review. Mentor Graphics shall bear the expense of any such review unless a material non-compliance is revealed. Mentor Graphics shall treat as confidential information all 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. The provisions of this Section 15 shall survive the termination of this Agreement.

16. **CONTROLLING LAW, JURISDICTION AND DISPUTE RESOLUTION.** The owners of certain Mentor Graphics intellectual property licensed under this Agreement are located in Ireland and the U.S. To promote consistency around the world, disputes shall be resolved as follows: excluding conflict of laws rules, this Agreement shall be governed by and construed under the laws of the State of Oregon, U.S., if Customer is located in North or South America, and the laws of Ireland if Customer is located outside of North or South America or Japan, and the laws of Japan if Customer is located in Japan. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of the courts of Portland, Oregon when the laws of Oregon apply, or Dublin, Ireland when the laws of Ireland apply, or the Tokyo District Court when the laws of Japan apply. Notwithstanding the foregoing, all disputes in Asia (excluding 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. Nothing in this section shall restrict Mentor Graphics' right to bring an action (including for example a motion for injunctive relief) against Customer in the jurisdiction where Customer's place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

17. **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.

18. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements. Any translation of this Agreement is provided to comply with local legal requirements only. In the event of a dispute between the English and any non-English versions, the English version of this Agreement shall govern to the extent not prohibited by local law in the applicable jurisdiction. This Agreement may only be modified in writing, signed by an authorized representative of each party. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse.