ModelSim®

Advanced Verification and Debugging

Tutorial

Version 6.0b

Published: November 15, 2004

Contractor/manufacturer is:

Mentor Graphics Corporation

8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

Contacting ModelSim Support

Telephone: 503.685.0820

Toll-Free Telephone: 877-744-6699

Website: www.model.com

Support: www.model.com/support

Table of Contents

Introduction

Topics

The following topics are covered in this chapter:

Assumptions

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

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

Where to find our documentation

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

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

Technical support and updates

Support

Model Technology online and email technical support options, maintenance renewal, and links to international support contacts:
www.model.com/support/default.asp

Mentor Graphics support:
www.mentor.com/supportnet

Updates

Access to the most current version of ModelSim:
www.model.com/downloads/default.asp

Latest version email

Place your name on our list for email notification of news and updates:
www.model.com/products/informant.asp

Before you begin

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

Example designs

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

Lesson 1 - ModelSim conceptual overview

Topics

The following topics are covered in this chapter:

Introduction

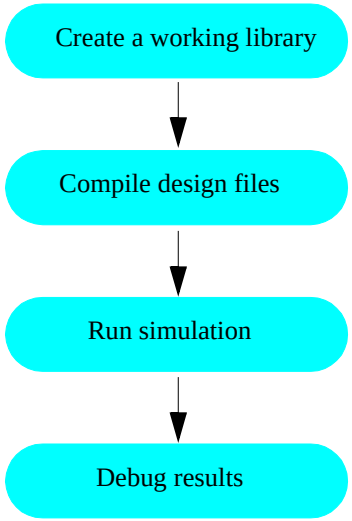
ModelSim is a simulation and debugging tool for VHDL, Verilog, SystemC, and mixed-language designs.

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

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

Basic simulation flow

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



Creating the working library

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

Compiling your design

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

Running the simulation

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

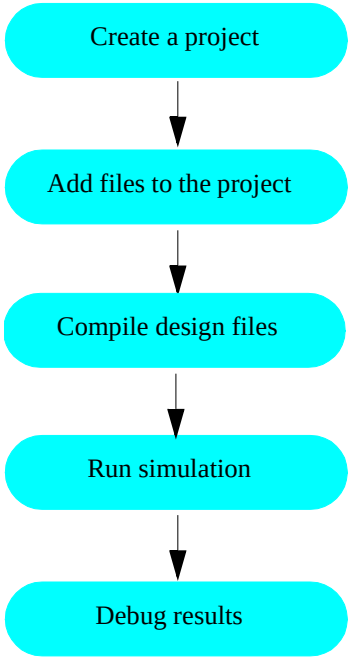
Debugging your results

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

Project flow

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

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



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

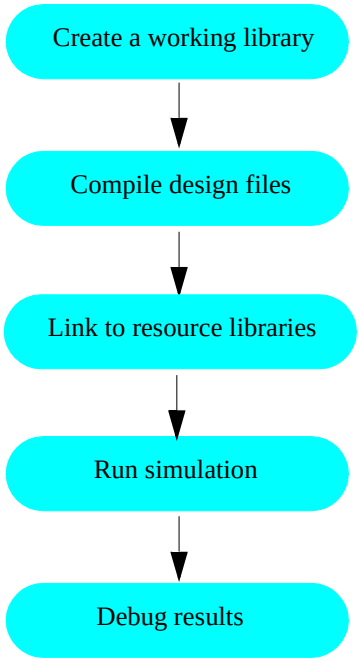
- You do not have to create a working library in the project flow; it is done for you automatically.
- Projects are persistent. In other words, they will open every time you invoke ModelSim unless you specifically close them.

Multiple library flow

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

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

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



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

Debugging tools

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

- Setting breakpoints and stepping through the source code
- Viewing waveforms and measuring time
- Exploring the "physical" connectivity of your design
- Viewing and initializing memories
- Testing code coverage
- Comparing waveforms

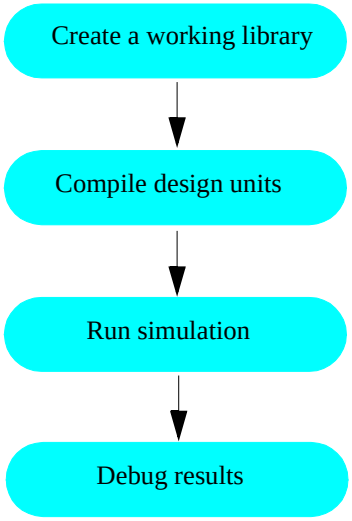
Lesson 2 - Basic simulation

Topics

The following topics are covered in this lesson:

Introduction

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



Design files for this lesson

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

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

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

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

Related reading

ModelSim User's Manual – Chapter 3 - Design libraries (UM-51), *Chapter 5 - Verilog simulation* (UM-97), *Chapter 4 - VHDL simulation* (UM-65)

ModelSim Command Reference (**vlib** (CR-281), **vmap** (CR-293), **vlog** (CR-282), **vcom** (CR-237), **and view** (CR-257) commands)

Creating the working design library

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

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

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

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

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

2 Start ModelSim if necessary.

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

Upon opening ModelSim for the first time, you will see the Welcome to ModelSim dialog (Figure 1). Click **Close**.

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

3 Create the working library.

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

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

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

Figure 1: The Welcome to ModelSim dialog

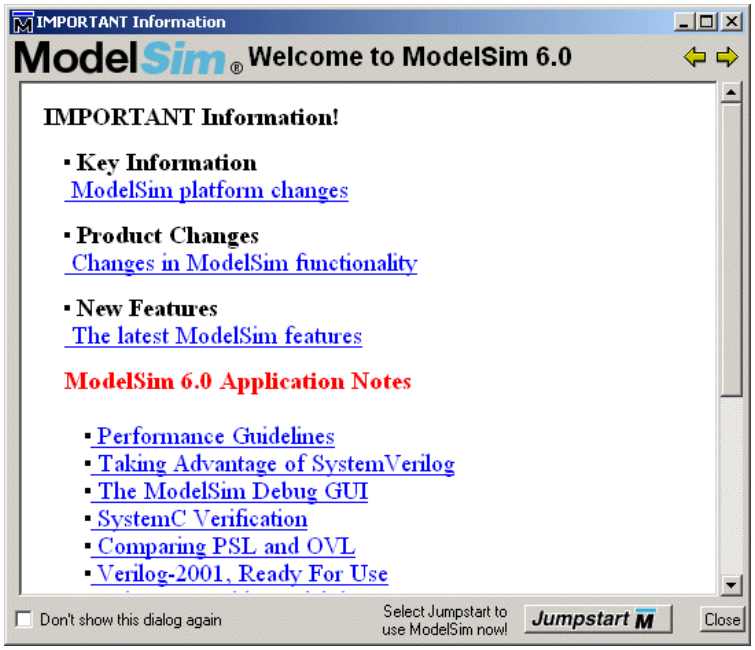
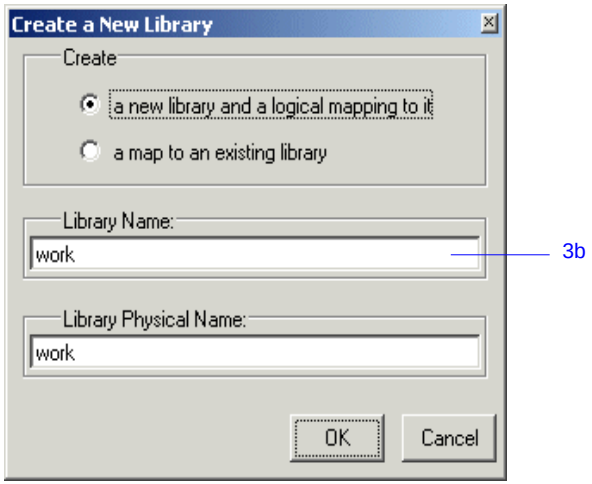


Figure 2: The Create a New Library dialog



c Click **OK**.

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

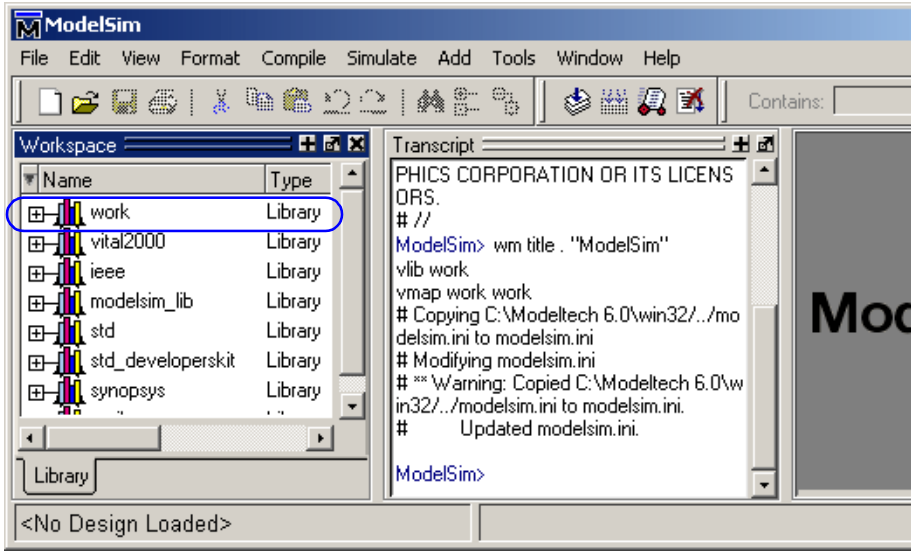
ModelSim also adds the library to the list in the Workspace (Figure 3) and records the library mapping for future reference in the ModelSim initialization file (*modelsim.ini*).

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

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

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

Figure 3: The newly created work library



Compiling the design

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

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

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

a Select **Compile > Compile**.

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

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

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

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

The files are compiled into the *work* library.

d Click **Done**.

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

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

2 View the compiled design units.

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

Figure 4: The Compile HDL Source Files dialog

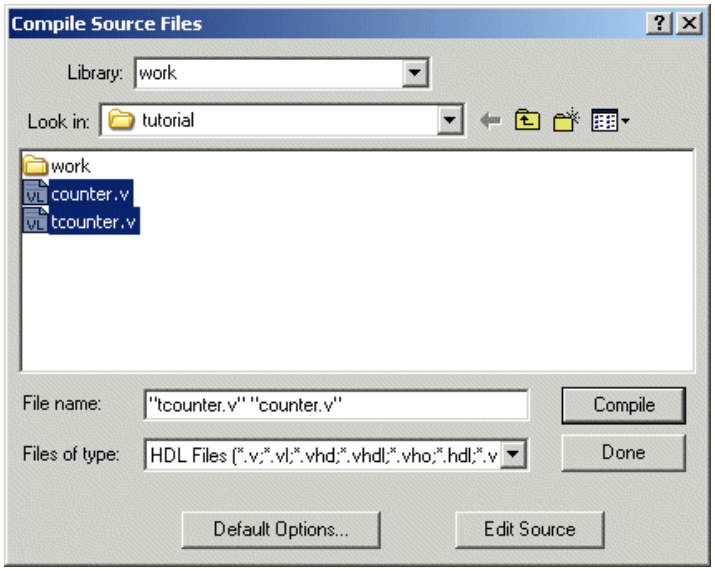
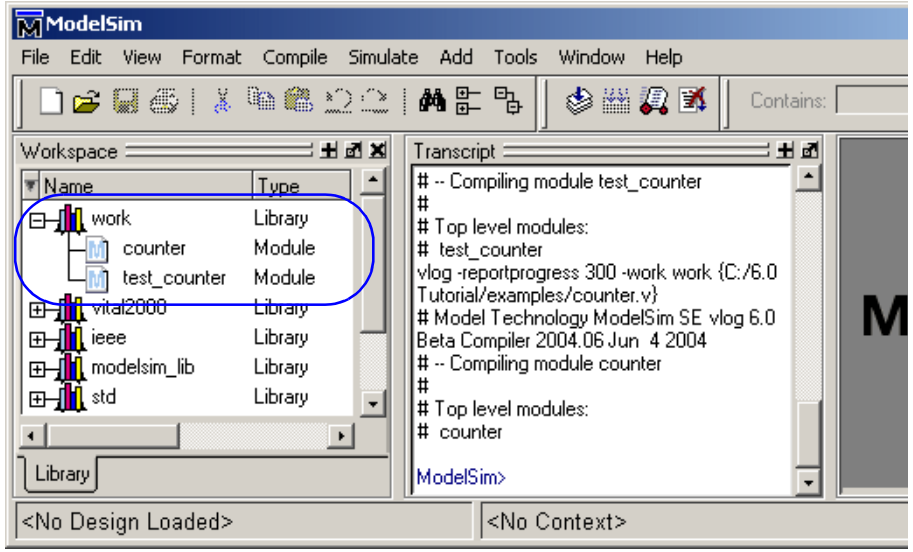


Figure 5: Verilog modules compiled into the work library



Loading the design into the simulator

1 Load the *test_counter* module into the simulator.

a Double-click *test_counter* in the Main window Workspace to load the design.

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

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

Figure 6: Loading the design with the Start Simulation dialog

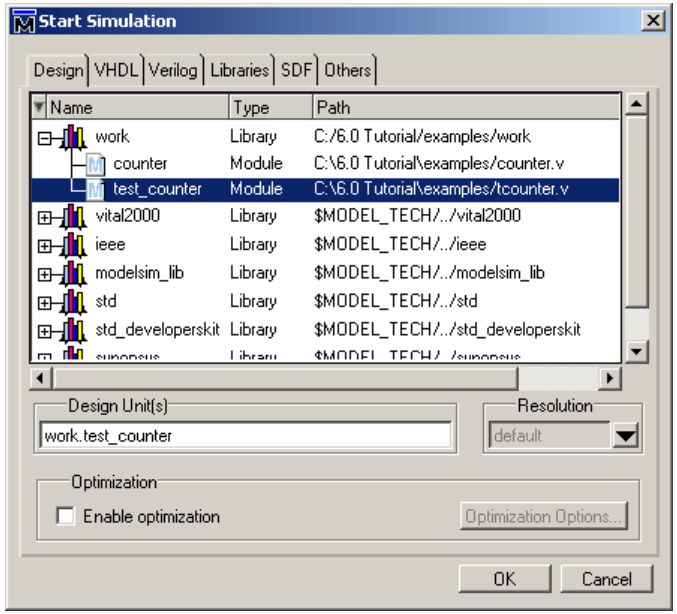
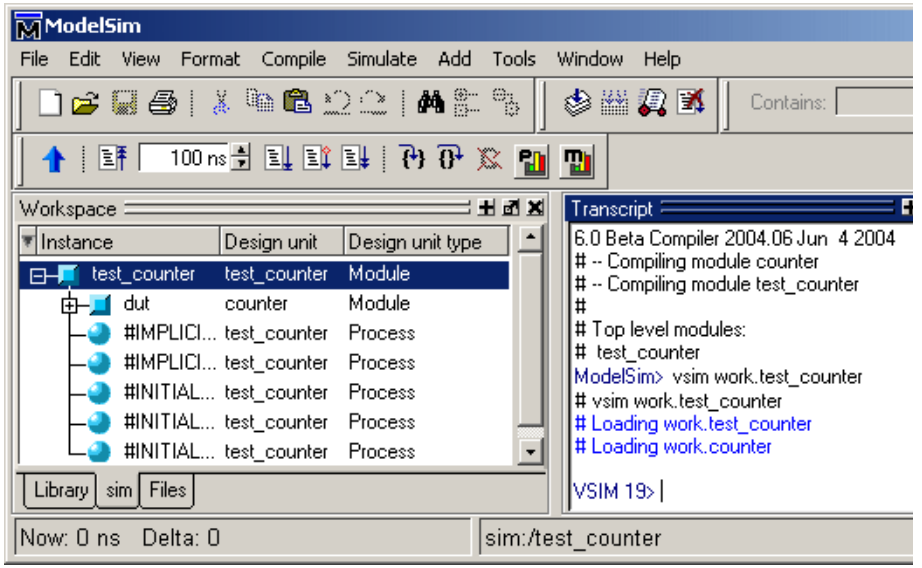


Figure 7: Workspace tab showing a Verilog design



Running the simulation

Now you will run the simulation.

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

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

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

2 Add signals to the Wave window.

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

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

c Select **Add > Add to Wave** (Figure 8).

Three signals are added to the Wave window.

3 Run the simulation.

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

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

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

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

Figure 8: Adding signals to the Wave window

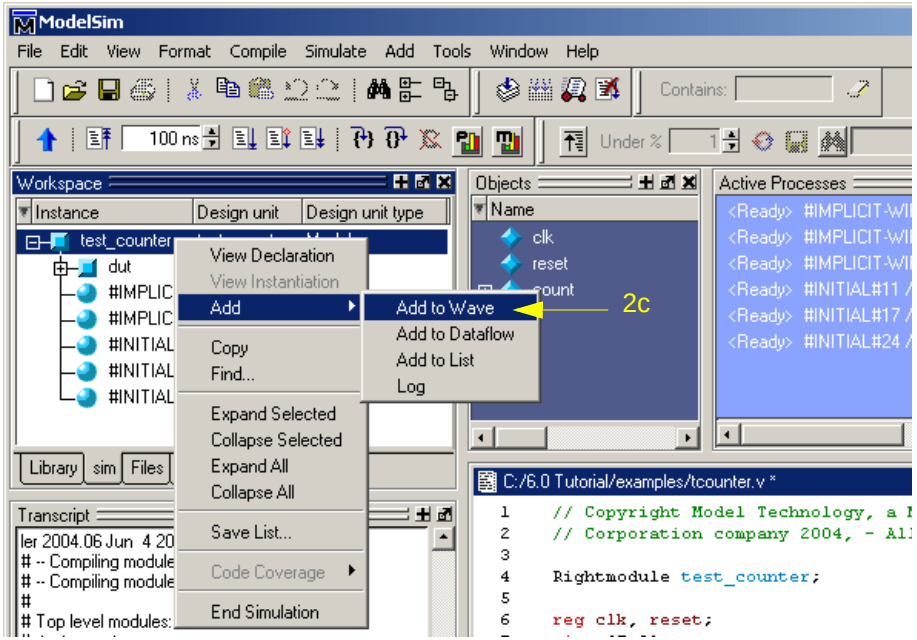
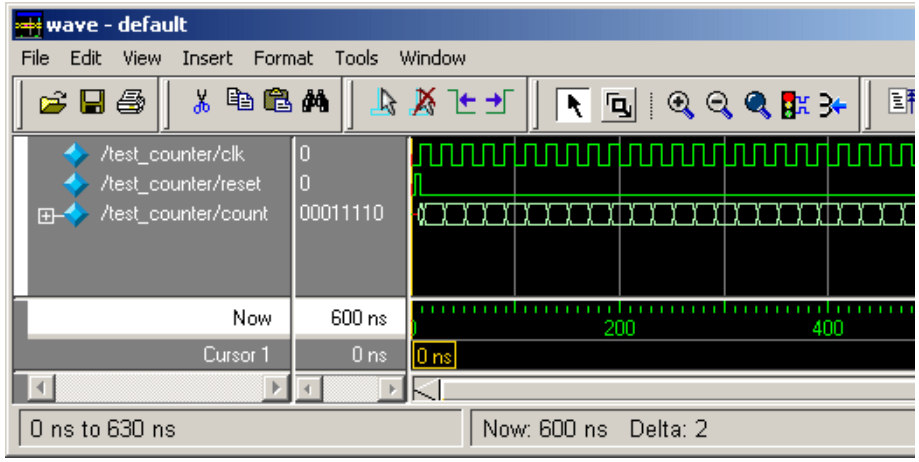


Figure 9: Waves being drawn in the Wave window



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

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

d Click the Break icon.

The simulation stops running.

Setting breakpoints and stepping in the Source window

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

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

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

   b Double-click *counter.v* to add it to the Source window.

2 Set a breakpoint on line 31 of *counter.v* (if you are simulating the VHDL files, use line 30 instead).

   a Scroll to line 31 and click on the line number.

   A red ball appears next to the line (Figure 10) indicating that a breakpoint has been set.

3 Disable, enable, and delete the breakpoint.

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

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

   c Click the red ball with your right mouse button and select **Remove Breakpoint 31**.

   d Click on line number 31 again to re-create the breakpoint.

4 Restart the simulation.

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

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

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

Figure 10: A breakpoint in the Source window

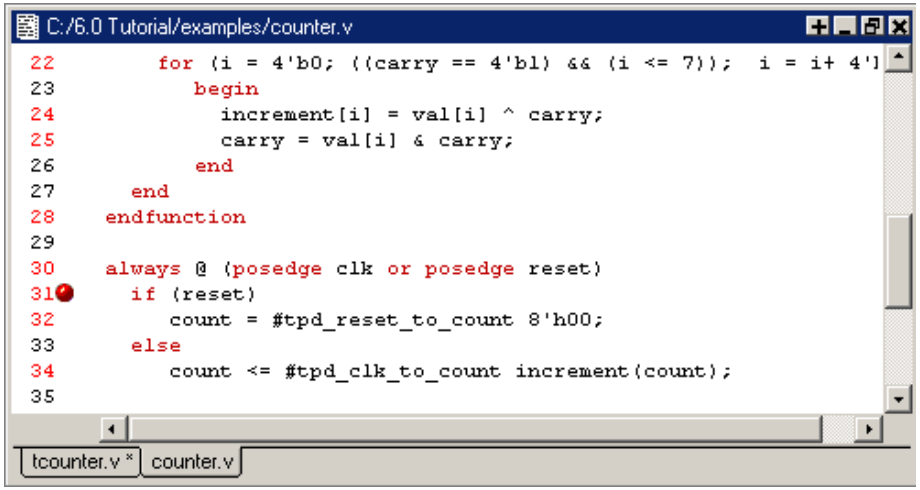
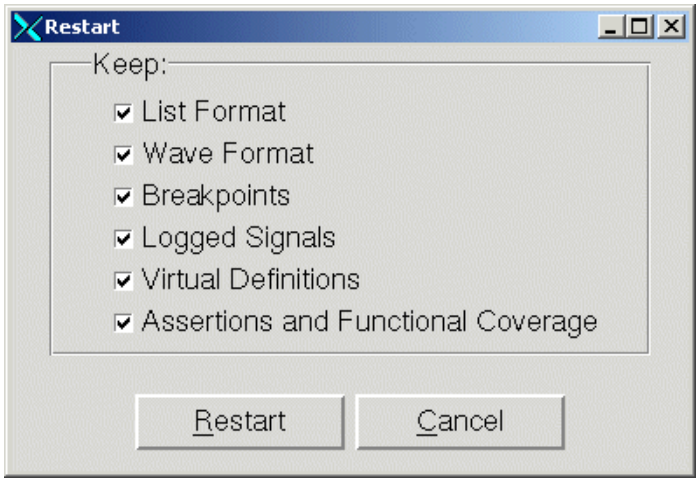


Figure 11: The Restart dialog



c Click the Run -All icon.

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

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

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

5 Try out the step commands.

a Click the Step icon on the Main window toolbar.

This single-steps the debugger.

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

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

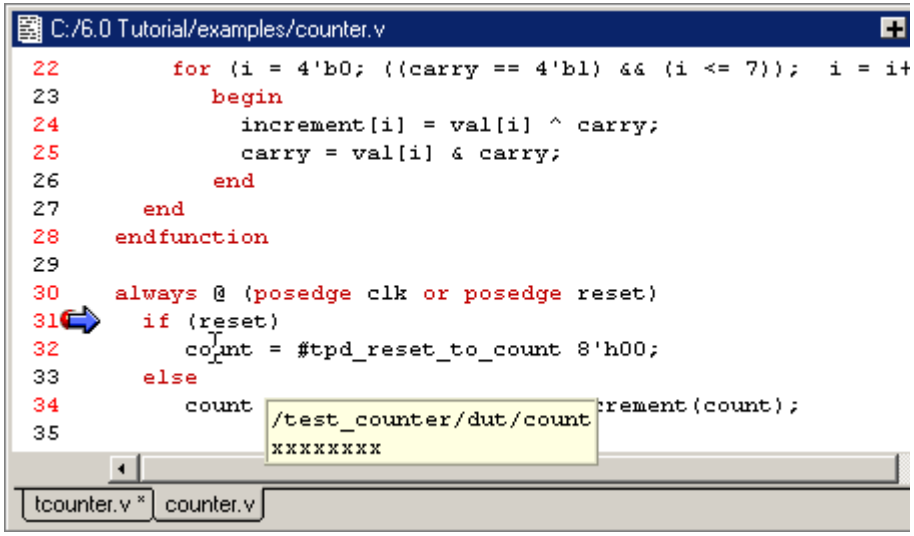
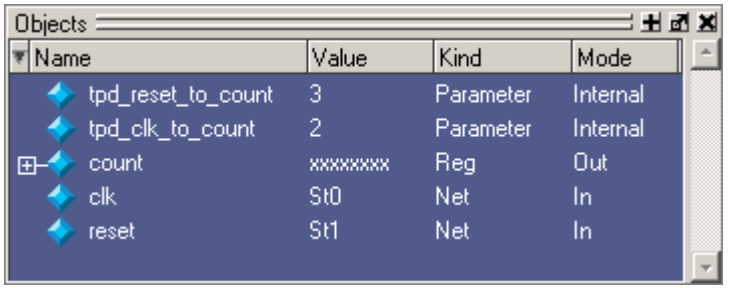


Figure 13: Values shown in the Objects window



Lesson wrap-up

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

1 Select **Simulate > End Simulation**.

2 Click **Yes** when prompted to confirm that you wish to quit simulating.

Lesson 3 - ModelSim projects

Topics

The following topics are covered in this lesson:

Introduction

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

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

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

Related reading

ModelSim User's Manual, Chapter 2 - Projects (UM-31)

Creating a new project

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

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

2 Create a new project.

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

   This opens a dialog where you enter a Project Name, Project Location (i.e., directory), and Default Library Name (Figure 14). The default library is where compiled design units will reside.

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

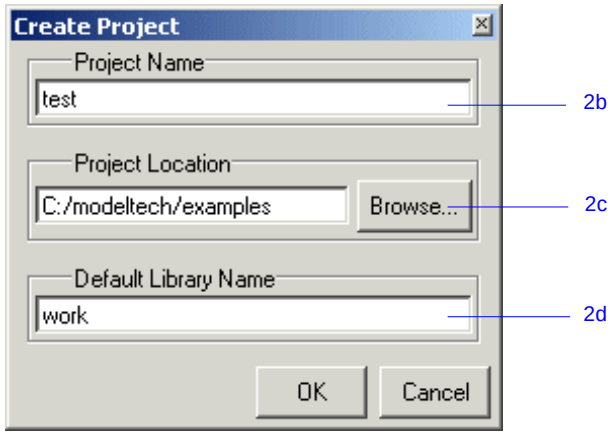
   c Click **Browse** to select a directory where the project file will be stored.

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

   e Click **OK**.

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

Figure 14: The Create Project dialog



Adding objects to the project

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

1 Add two existing files.

a Click **Add Existing File**.

This opens the Add file to Project dialog (Figure 16). This dialog lets you browse to find files, specify the file type, specify which folder to add the file to, and identify whether to leave the file in its current location or to copy it to the project directory.

b Click **Browse**.

c Open the *examples* directory in your ModelSim installation tree.

d **Verilog:** Select *counter.v*, hold the <Ctrl> key down, and then select *tcounter.v*.
VHDL: Select *counter.vhd*, hold the <Ctrl> key down, and then select *tcounter.vhd*.

e Click **Open** and then **OK**.

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

Figure 15: Adding new items to a project

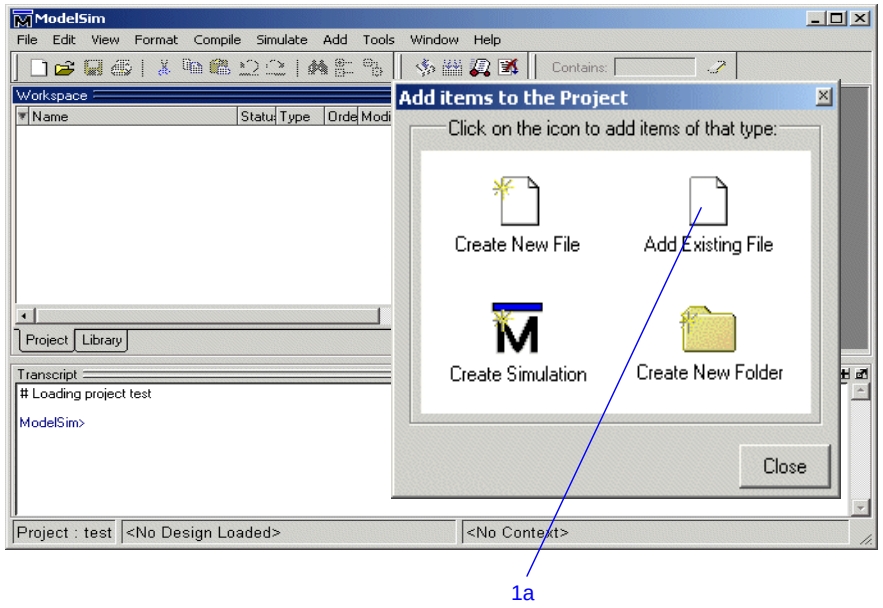
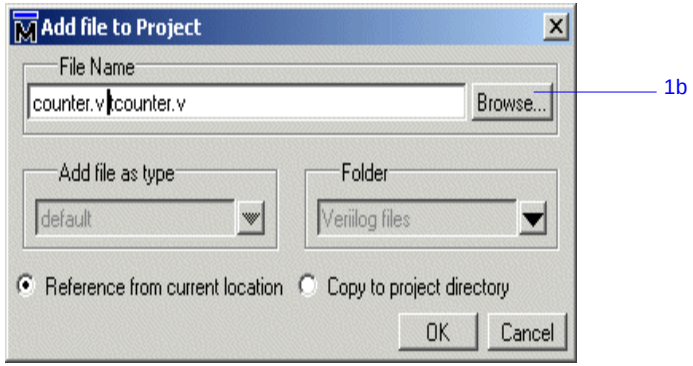


Figure 16: The Add file to Project dialog



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

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

Changing compile order (VHDL)

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

1 Change the compile order.

a Select **Compile > Compile Order**.

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

b Click the **Auto Generate** button.

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

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

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

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

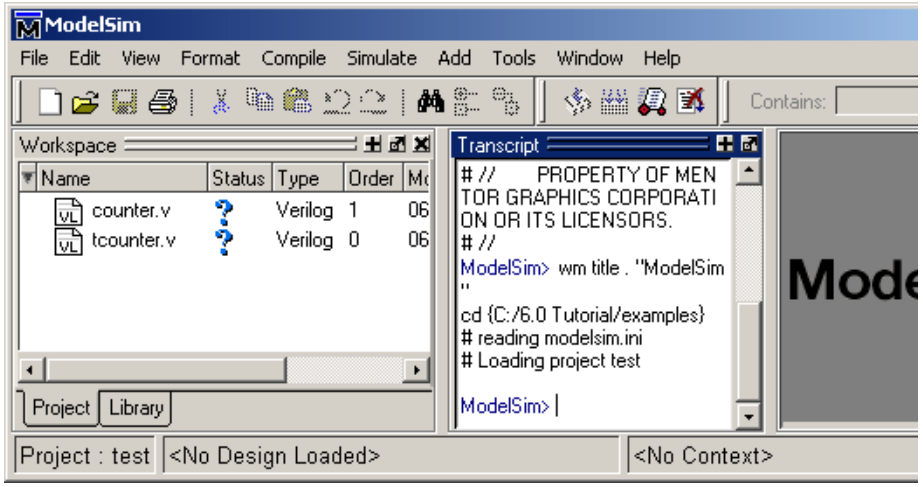
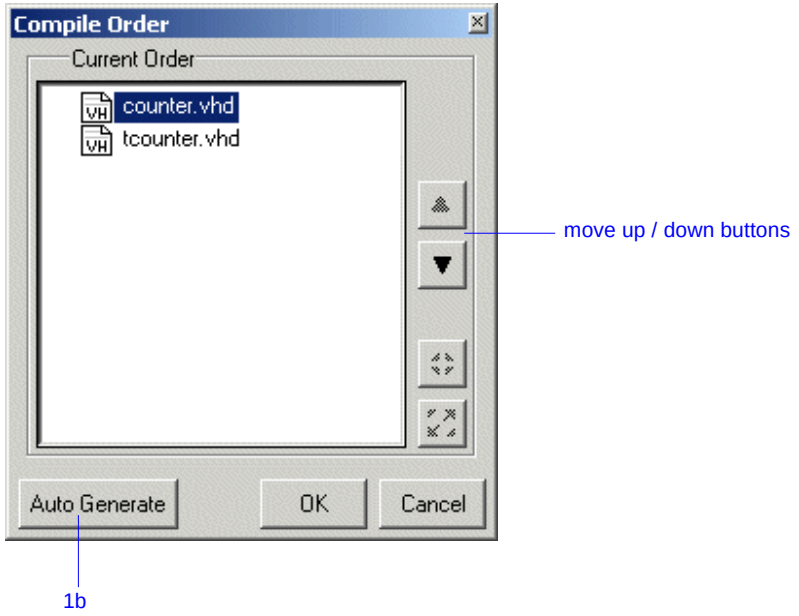


Figure 18: The Compile Order dialog box



Compiling and loading a design

1 Compile the files.

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

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

2 View the design units.

   a Click the **Library** tab in the workspace.

   b Click the "+" icon next to the *work* library.

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

3 Load the *test_counter* design unit.

   a Double-click the *test_counter* design unit.

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

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

4 End the simulation.

   a Select **Simulate > End Simulation**.

   b Click **Yes**.

Figure 19: The Library tab with an expanded library

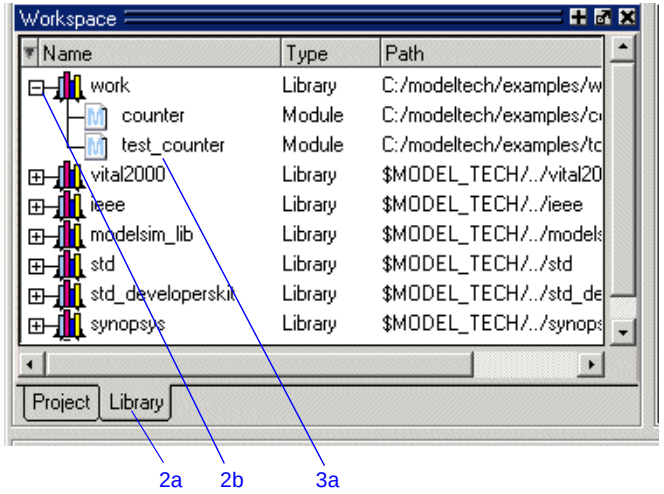
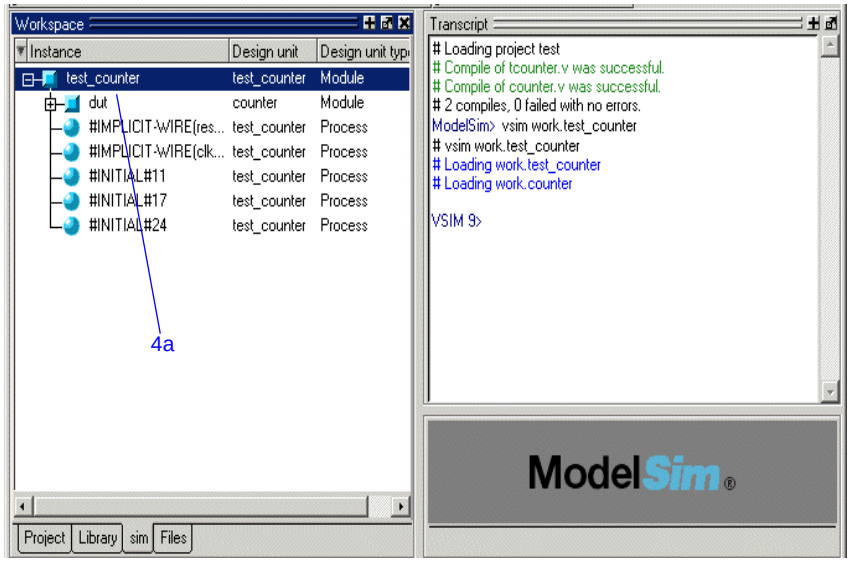


Figure 20: The structure tab for the *counter* design unit



Organizing projects with folders

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

Adding folders

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

1 Add a new folder.

a Select **File > Add to Project > Folder**.

b Type **Design Files** in the **Folder Name** field (Figure 21).

c Click **OK**.

You'll now see a folder in the Project tab (Figure 22).

2 Add a sub-folder.

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

b Type **HDL** in the **Folder Name** field (Figure 23).

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

d Click OK.

Figure 21: Adding a new folder to the project

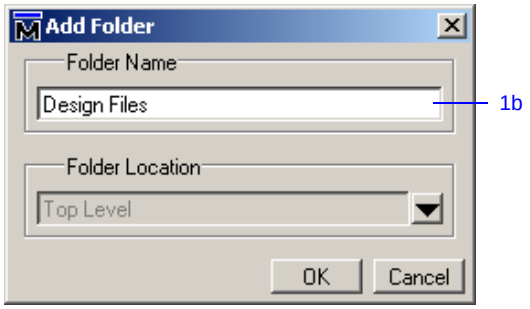


Figure 22: A folder in a project

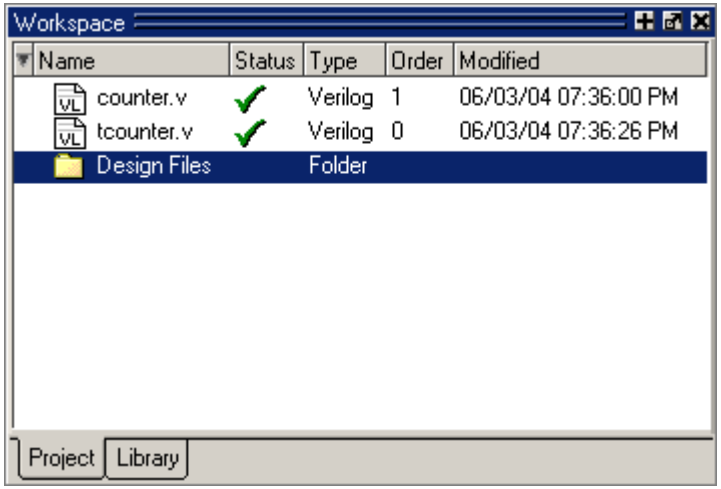
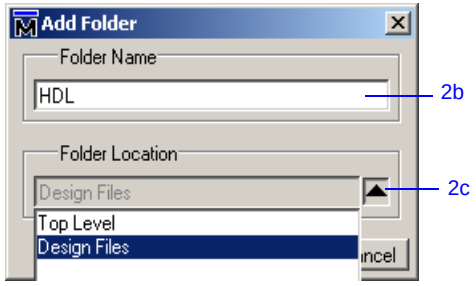


Figure 23: Creating a subfolder



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

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

Moving files to folders

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

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

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

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

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

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

d Click OK.

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

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

Figure 24: A folder with a sub-folder

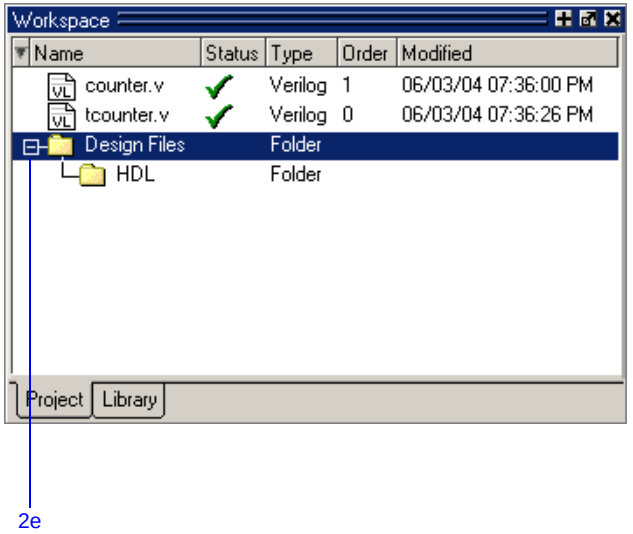
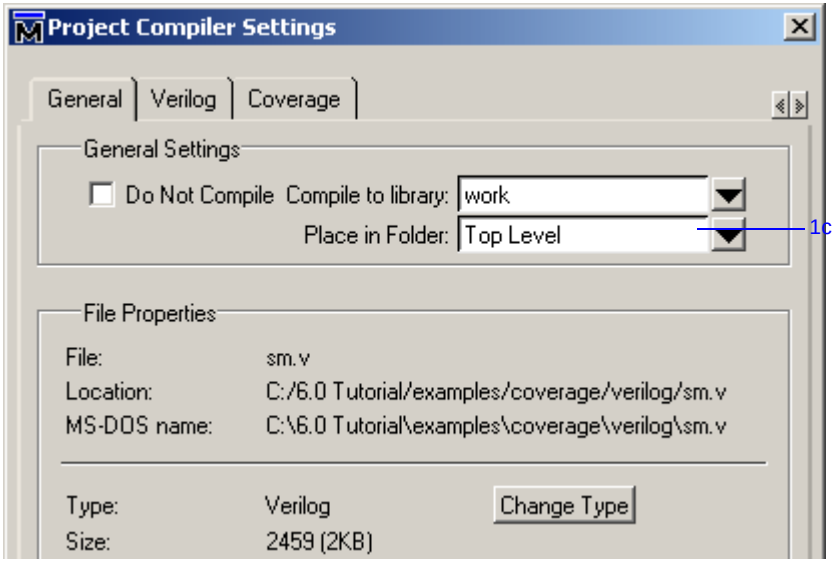


Figure 25: Changing file location via the project settings dialog



Simulation Configurations

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

1 Create a new Simulation Configuration.

a Select **File > Add to Project > Simulation Configuration**.

This opens the Simulate dialog (Figure 26). The tabs in this dialog present a myriad of simulation options. You may want to explore the tabs to see what's available. You can consult the ModelSim User's Manual to get a description of each option.

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

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

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

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

f For Verilog, click the Verilog tab and check **Enable Hazard Checking**.

g Click **OK**.

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

2 Load the Simulation Configuration.

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

In the Transcript pane of the Main window, the **vsim** (the ModelSim simulator) invocation shows the **-hazards** and **-t ps** switches (Figure 28). These are the command-line equivalents of the options you specified in the Simulate dialog.

Figure 26: The Simulation Configuration dialog

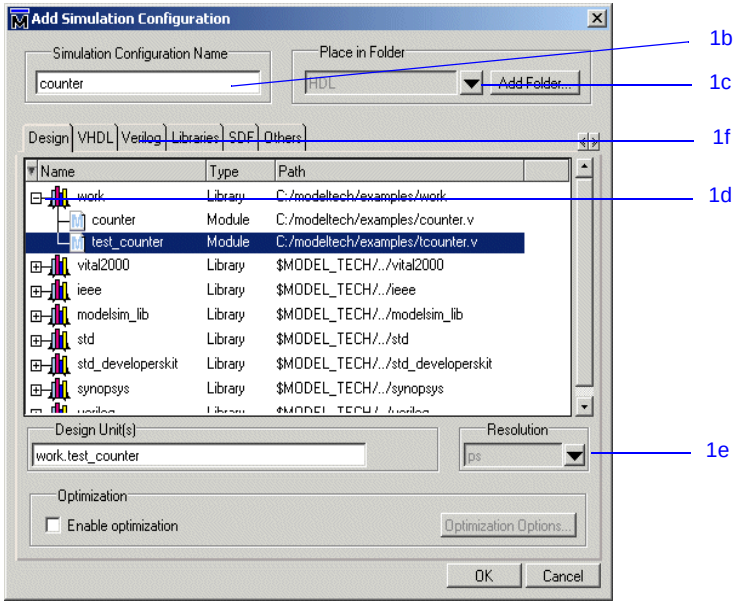
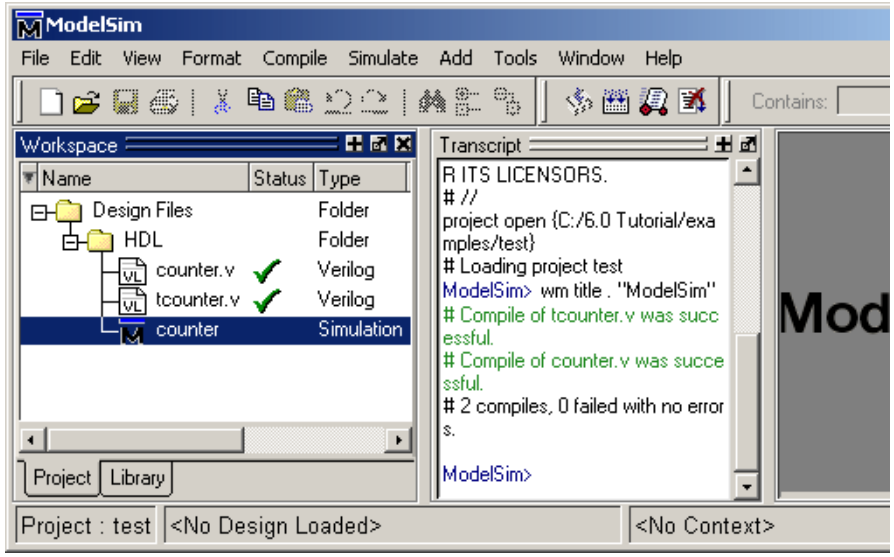


Figure 27: A Simulation Configuration in the Project tab



Lesson wrap-up

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

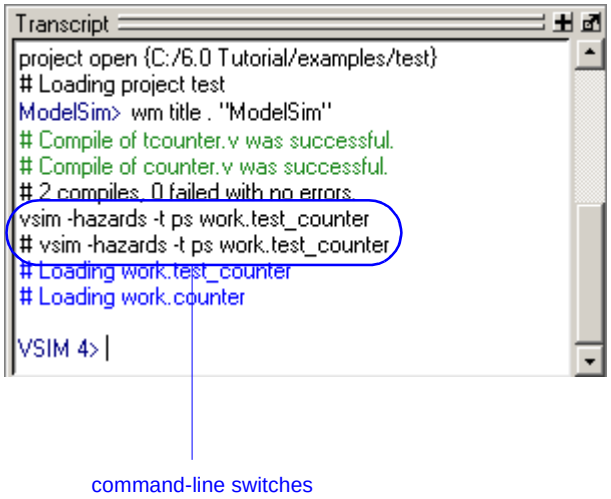
1 Select **Simulate > End Simulation**. Click Yes.

2 Select the Project tab in the Main window Workspace.

3 Right-click the *test* project to open a context popup menu and select **Close Project**.

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

Figure 28: Transcript shows options used for Simulation Configuration



Lesson 4 - Working with multiple libraries

Topics

The following topics are covered in this lesson:

Introduction

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

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

Design files for this lesson

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

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

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

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

Related reading

ModelSim User's Manual, *3 - Design libraries* (UM-51)

Creating the resource library

1 Create a directory for the resource library.

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

2 Create a directory for the testbench.

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

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

3 Start ModelSim and change to the exercise directory.

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

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

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

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

4 Create the resource library.

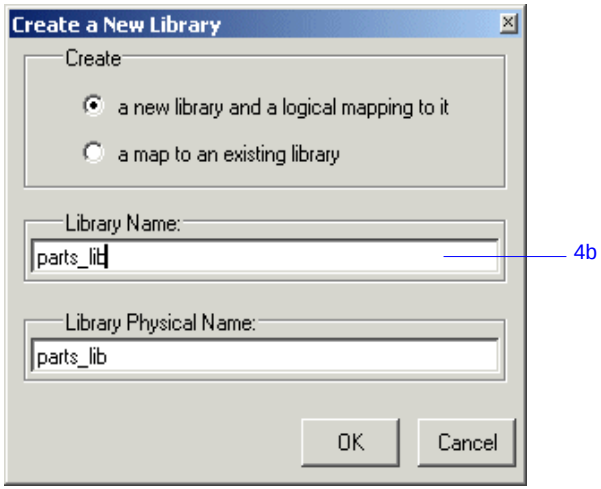
a Select **File > New > Library**.

b Type **parts_lib** in the Library Name field (Figure 29).

The Library Physical Name field is filled out automatically.

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

Figure 29: Creating the new resource library



5 Compile the counter into the resource library.

a Click the Compile icon on the Main window toolbar.

b Select the *parts_lib* library from the Library list (Figure 30).

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

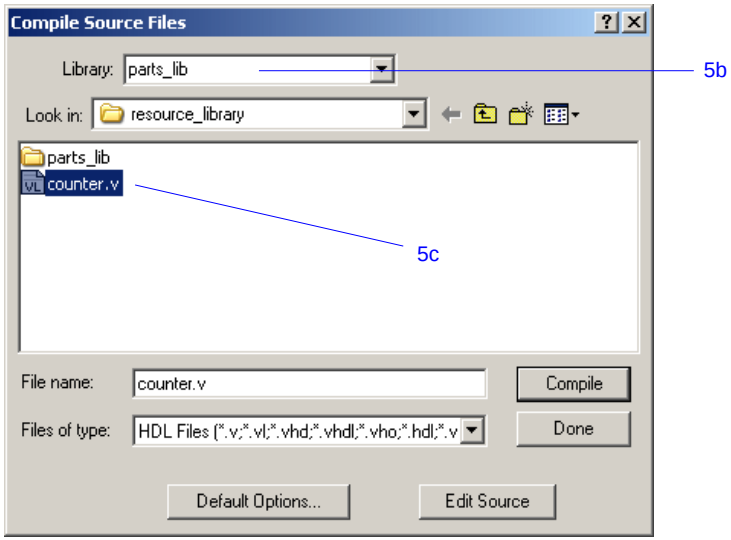
d Click Done.

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

6 Change to the *testbench* directory.

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

Figure 30: Compiling into the resource library



Creating the project

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

1 Create the project.
   a Select **File > New > Project**.
   b Type **counter** in the Project Name field.
   c Click **OK**.
   d If a dialog appears asking about which *modelsim.ini* file to use, click **Use Default Ini**.

2 Add the testbench to the project.
   a Click **Add Existing File** in the Add items to the Project dialog.
   b Click the Browse button and select *tcounter.v*.
   c Click Open and then OK.
   d Click Close to dismiss the Add items to the Project dialog.

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

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

Linking to the resource library

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

ModelSim responds differently for Verilog and VHDL in this situation.

Verilog

1 Simulate a Verilog design with a missing resource library.

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

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

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

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

VHDL

1 Simulate a VHDL design with a missing resource library.

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

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

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

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

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

Figure 31: Verilog simulation error reported in the Main window

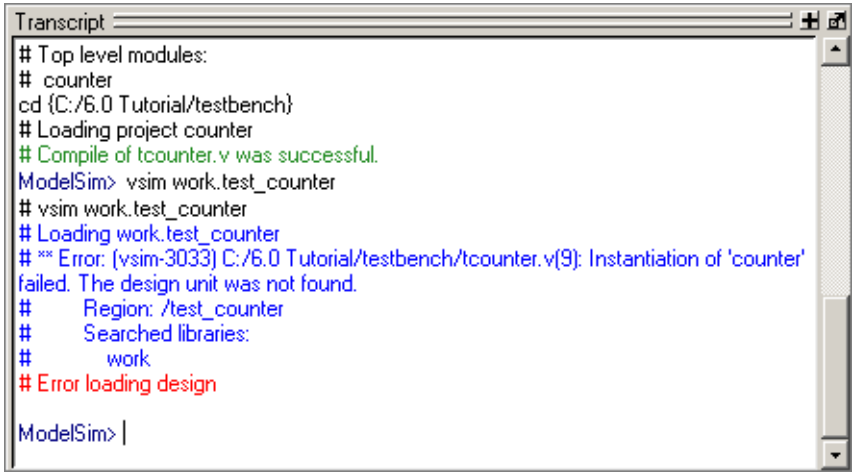
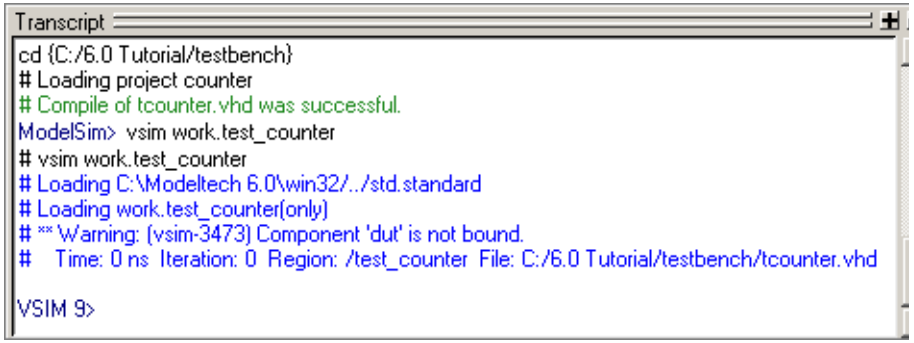


Figure 32: VHDL simulation warning reported in Main window



The process for linking to a resource library differs between Verilog and VHDL. If you are using Verilog, follow the steps in "Linking in Verilog" (T-47). If you are using VHDL, follow the steps in "Linking in VHDL" (T-48) one page later.

Linking in Verilog

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

1 Specify a search library during simulation.

a Click the Simulate icon on the Main window toolbar.

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

c Click the Libraries tab.

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

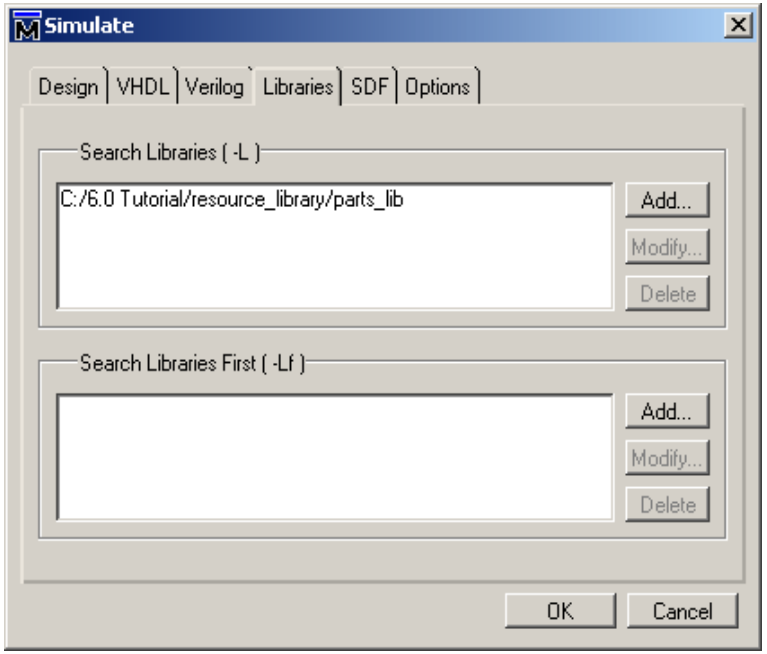
e Click OK.

The dialog should have *parts_lib* listed in the Search Libraries field (Figure 33).

f Click OK.

The design loads without errors.

Figure 33: Specifying a search library in the Simulate dialog



Linking in VHDL

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

1 Create a logical mapping to *parts_lib*.

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

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

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

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

e The Create a New Library dialog should look similar to the one shown in Figure 34. Click OK to close the dialog.

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

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

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

This opens the file in the Source window.

c Add these two lines to the top of the file:

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

The testbench source code should now look similar to that shown in Figure 33.

d Select **File > Save**.

3 Recompile and simulate.

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

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

The design loads without errors.

Figure 34: Mapping to the parts_lib library

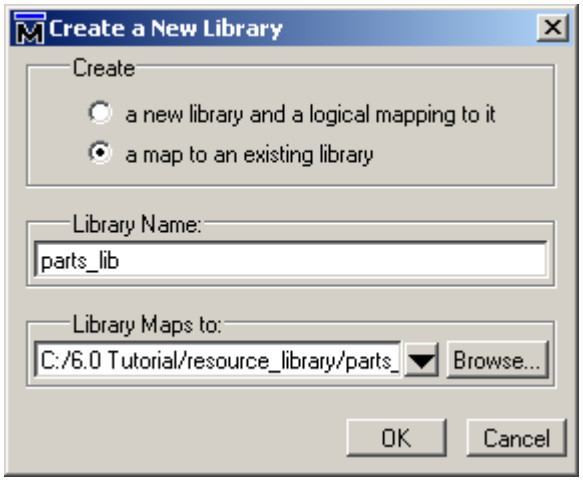
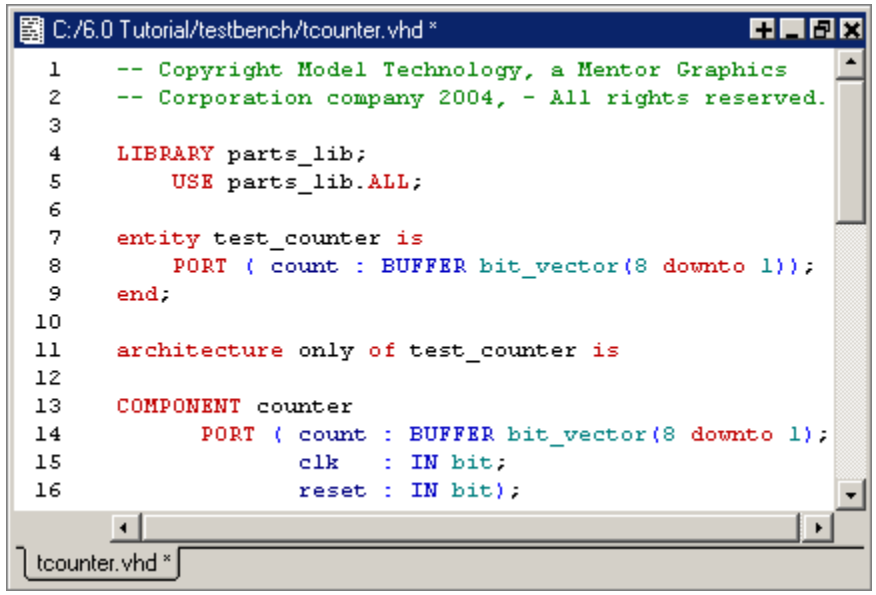


Figure 35: Adding LIBRARY and USE statements to the testbench



Permanently mapping resource libraries

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

1 Locate the *modelsim.ini* file in the ModelSim installation directory (<*install_dir*>*/modeltech/modelsim.ini*).

2 IMPORTANT - Make a backup copy of the file.

3 Change the file attributes of modelsim.ini so it is no longer "read-only."

4 Open the file and enter your library mappings in the [Library] section. For example:

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

5 Save the file.

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

Lesson wrap-up

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

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

2 Select the Project tab of the Main window Workspace.

3 Select **File > Close**. Click OK.

Lesson 5 - Viewing simulations in the Wave window

Topics

The following topics are covered in this lesson:

Introduction

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

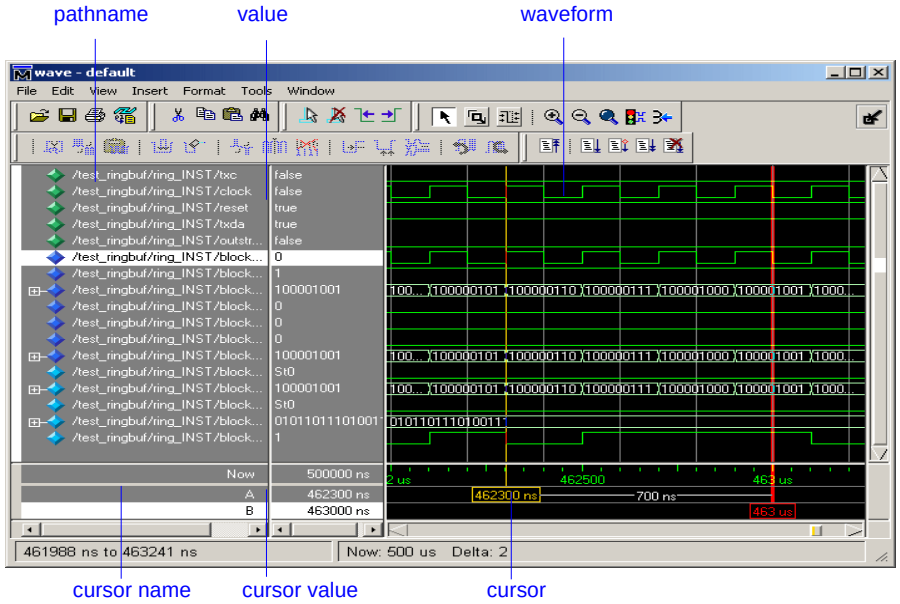
The Wave window is divided into a number of window panes (Figure 36). All window panes in the Wave window can be resized by clicking and dragging the bar between any two panes.

Related reading

ModelSim GUI Reference – "Wave window" (GR-182)

ModelSim User's Manual – Chapter 8 - WLF files (datasets) and virtuals (UM-195)

Figure 36: The Wave window and its many panes



Loading a design

For the examples in this lesson, we have used the design simulated in *Lesson 2 - Basic simulation*.

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

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

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

2 Load the design.

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

      The *work* library should already exist.

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

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

Adding objects to the Wave window

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

1 Add objects from the Objects pane.

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

   ModelSim adds several signals to the Wave window.

2 Undock the Wave window.

   By default ModelSim opens Wave windows as a tab in the MDI frame of the Main window. You can change the default via the Preferences dialog (Tools > Edit Preferences). See "ModelSim GUI preferences" (GR-236) in the *ModelSim GUI & Interface Reference* for more information.

   a Click the undock icon on the Wave pane (Figure 37).

   The Wave pane becomes a standalone, un-docked window.

3 Add objects using drag-and-drop.

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

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

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

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

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

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

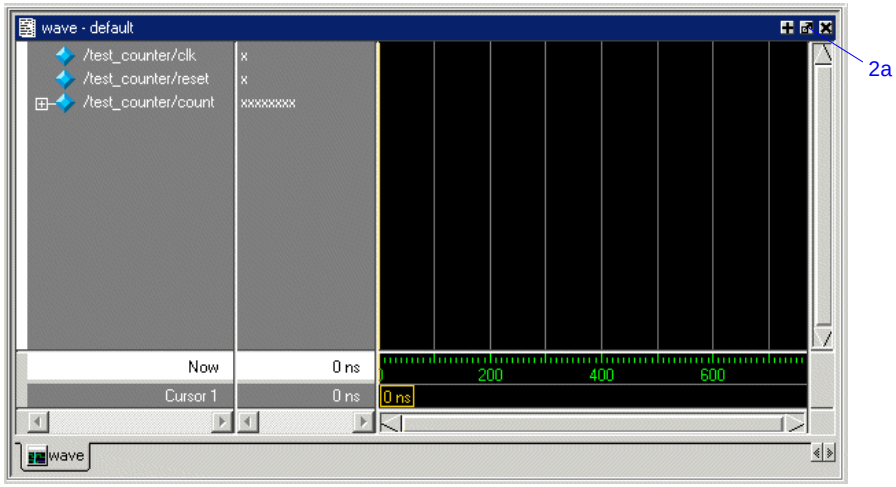
4 Add objects using a command.

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

   ModelSim adds all objects from the current region.

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

Figure 37: A Wave window docked in the Main window



Zooming the waveform display

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

1 Zoom the display using various techniques.

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

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

You should see blue vertical lines and numbers defining an area to zoom in (Figure 38).

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

The waveform pane returns to the previous display range.

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

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

You should see a blue line and numbers defining an area to zoom out (Figure 39).

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

Figure 38: Zooming in with the mouse pointer

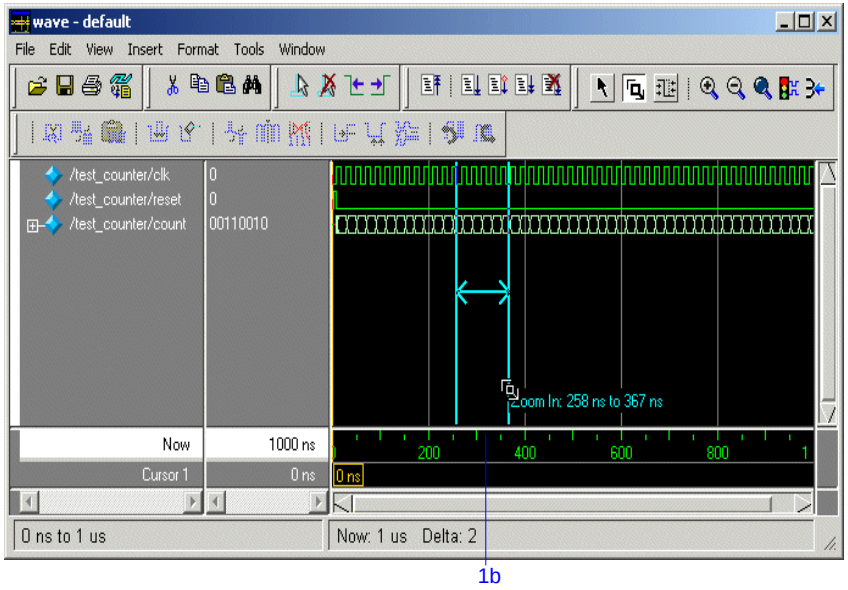
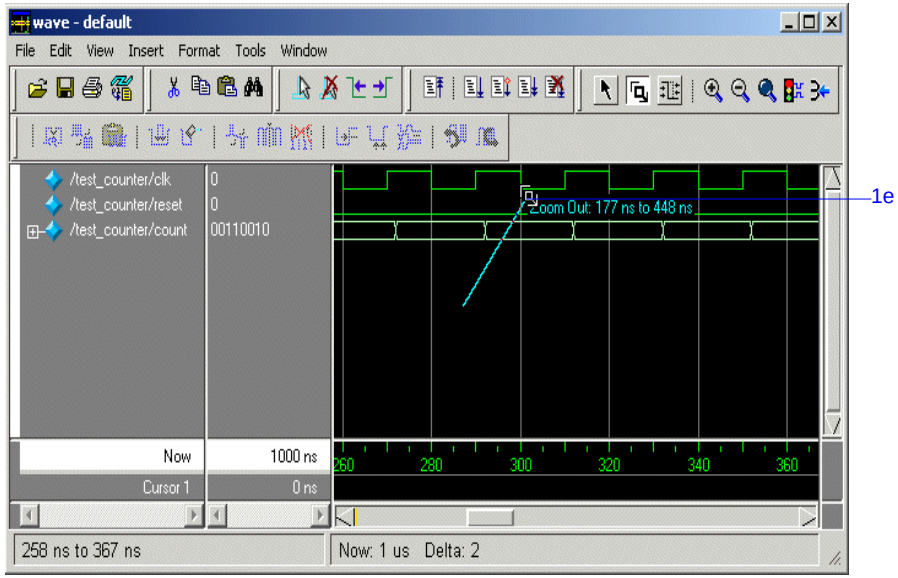


Figure 39: Zooming out with the mouse pointer



Using cursors in the Wave window

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

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

Working with a single cursor

1 Position the cursor by clicking and dragging.

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

b Click anywhere in the waveform pane.

A cursor is inserted at the time where you clicked (Figure 40).

c Drag the cursor and observe the value pane.

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

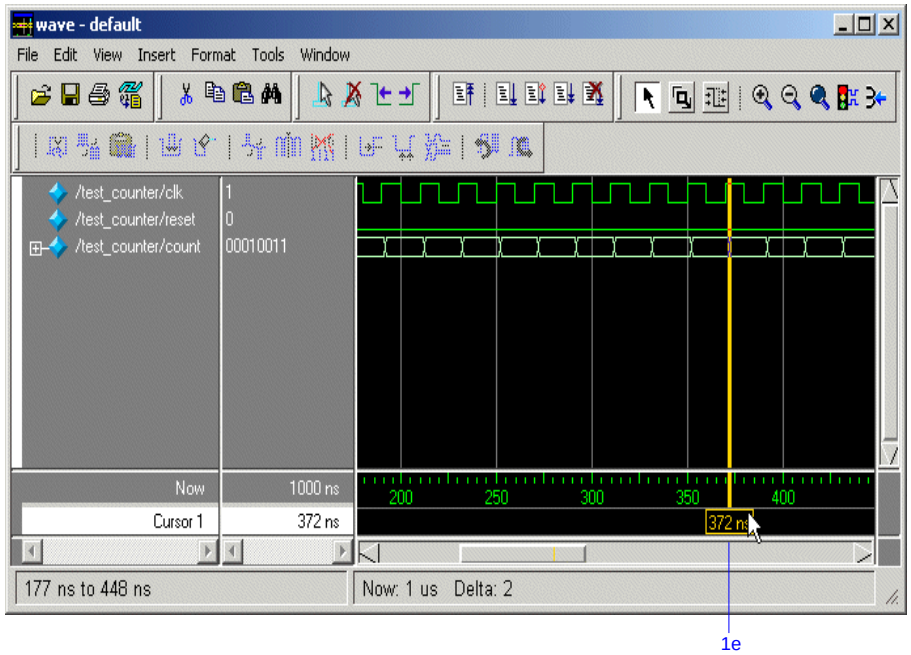
d In the waveform pane, drag the cursor to the right of a transition with the mouse positioned over a waveform.

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

e In the cursor pane, drag the cursor to the right of a transition (Figure 40).

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

Figure 40: Working with a single cursor in the Wave window



2 Rename the cursor.

a Right-click "Cursor 1" in the cursor name pane, and select and delete the text (Figure 41).

b Type **A** and press Enter.

The cursor name changes to "A".

3 Jump the cursor to the next or previous transition.

a Click signal *count* in the pathname pane.

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

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

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

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

Working with multiple cursors

1 Add a second cursor.

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

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

c Type **B** and press Enter.

d Drag cursor *B* and watch the interval measurement change dynamically (Figure 42).

Figure 41: Renaming a cursor

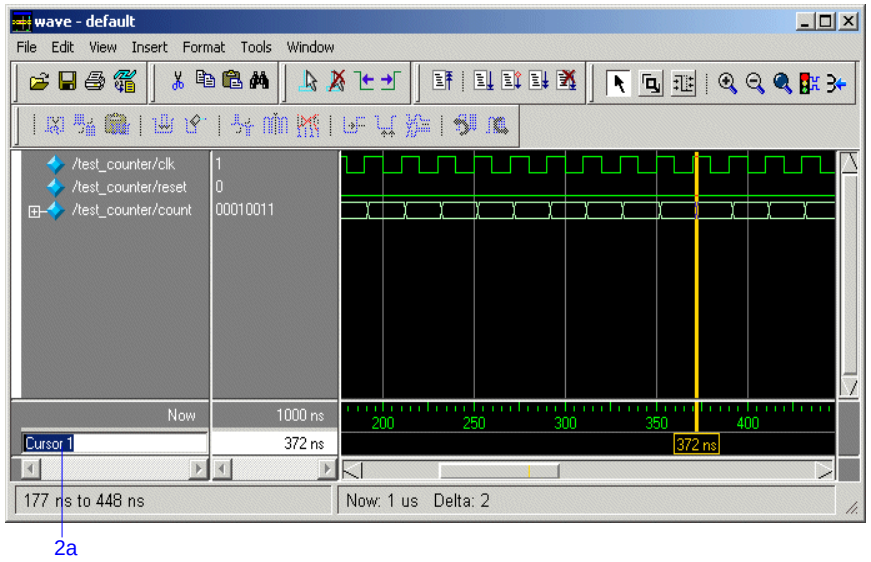
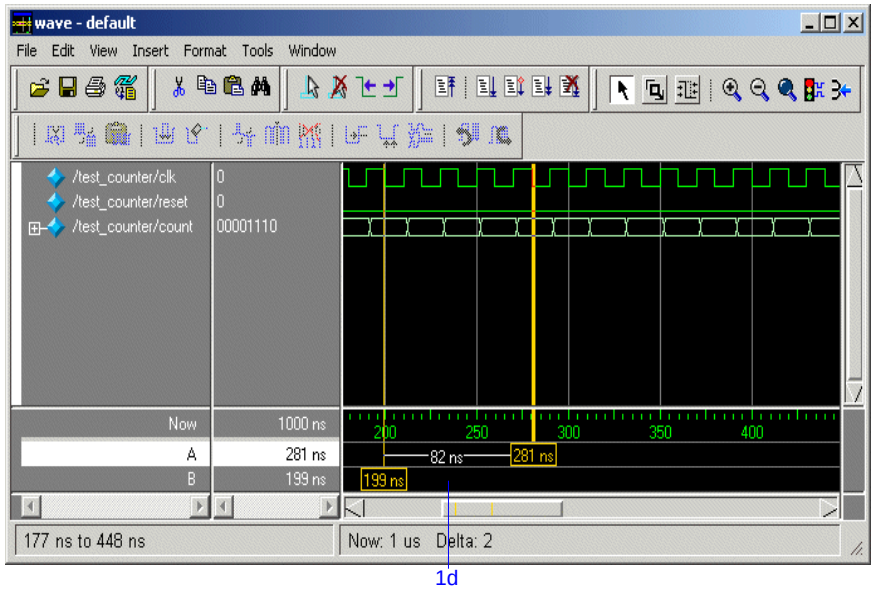


Figure 42: Interval measurement between two cursors



2 Lock cursor *B*.

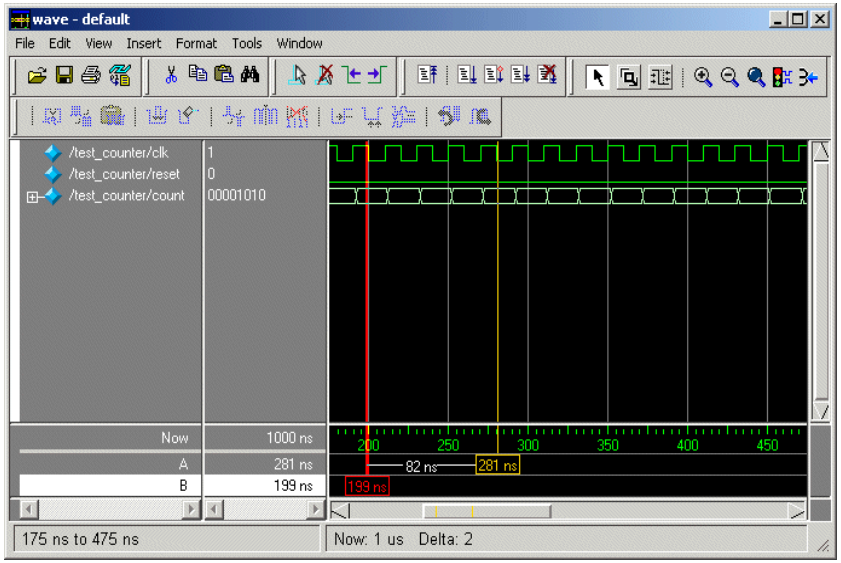
a Right-click cursor *B* in the cursor pane and select **Lock B**.

The cursor color changes to red and you can no longer drag the cursor (Figure 43).

3 Delete cursor *B*.

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

Figure 43: A locked cursor in the Wave window



Saving the window format

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

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

1 Save a format file.
   a Select **File > Save > Format**.
   b Leave the file name set to *wave.do* and click **Save**.
   c Close the Wave window.

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

Lesson wrap-up

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

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

Lesson 6 - Debugging with the Dataflow window

Topics

The following topics are covered in this lesson:

Note: The functionality described in this tutorial requires a dataflow license feature in your ModelSim license file. Please contact your Mentor Graphics sales representative if you currently do not have such a feature.

Introduction

The Dataflow window allows you to explore the "physical" connectivity of your design; to trace events that propagate through the design; and to identify the cause of unexpected outputs. The window displays processes; signals, nets, and registers; and interconnect.

Design files for this lesson

The sample design for this lesson is a testbench that verifies a cache module and how it works with primary memory. A processor design unit provides read and write requests.

The pathnames to the files are as follows:

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

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

This lesson uses the Verilog version in the examples. If you have a VHDL license, use the VHDL version instead. When necessary, we distinguish between the Verilog and VHDL versions of the design.

Related reading

ModelSim User's Manual –"Tracing signals with the Dataflow window" (UM-255)

ModelSim GUI Reference – "Dataflow window" (GR-111)

Compiling and loading the design

In this exercise you will use a DO file to compile and load the design.

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

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

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

2 Start ModelSim and change to the exercise directory.

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

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

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

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

3 Execute the lesson DO file.

a Type **do run.do** at the ModelSim> prompt.

The DO file does the following:

- Creates the working library
- Compiles the design files
- Opens the Dataflow and Wave windows
- Adds signals to the Wave window
- Logs all signals in the design
- Runs the simulation

Feel free to open the DO file and look at its contents.

Exploring connectivity

A primary use of the Dataflow window is exploring the "physical" connectivity of your design. You do this by expanding the view from process to process. This allows you to see the drivers/receivers of a particular signal, net, or register.

1 Add a signal to the Dataflow window.

a Make sure instance *p* is selected in the sim tab of the Workspace pane.

b Drag signal *strb* from the Objects pane to the Dataflow window (Figure 44).

2 Explore the design.

a Double-click the net highlighted in red.

The view expands to display the processes that are connected to *strb* (Figure 45).

b Select signal *test* on process *#NAND#44* (labeled *line_62* in the VHDL version) and click the **Expand net to all drivers** icon.

Notice that after the display expands, the signal line for *strb* is highlighted in green. This highlighting indicates the path you have traversed in the design.

c Select signal *oen* on process *#ALWAYS#149*(labeled *line_75* in the VHDL version), and click the **Expand net to all readers** icon.

Continue exploring if you wish. When you are done, click the **Erase All** icon.

Figure 44: A signal in the Dataflow window

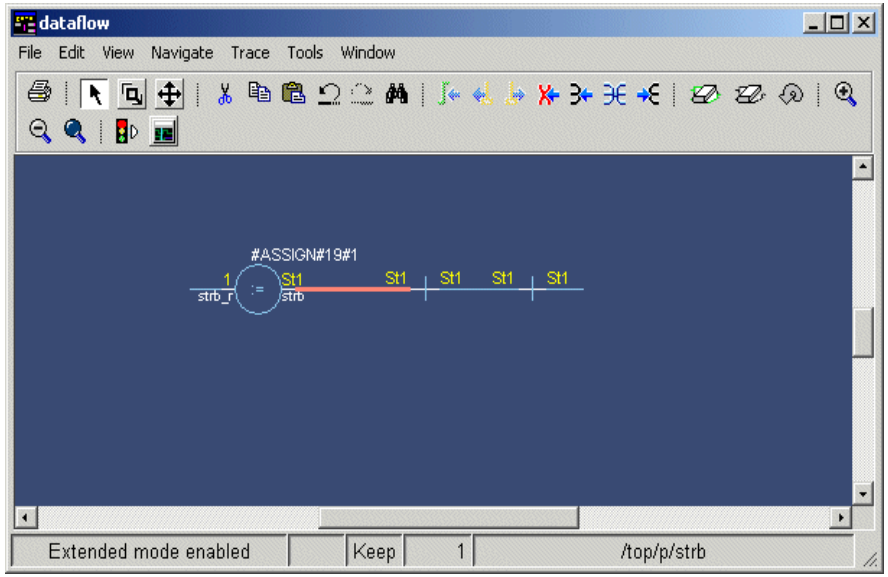
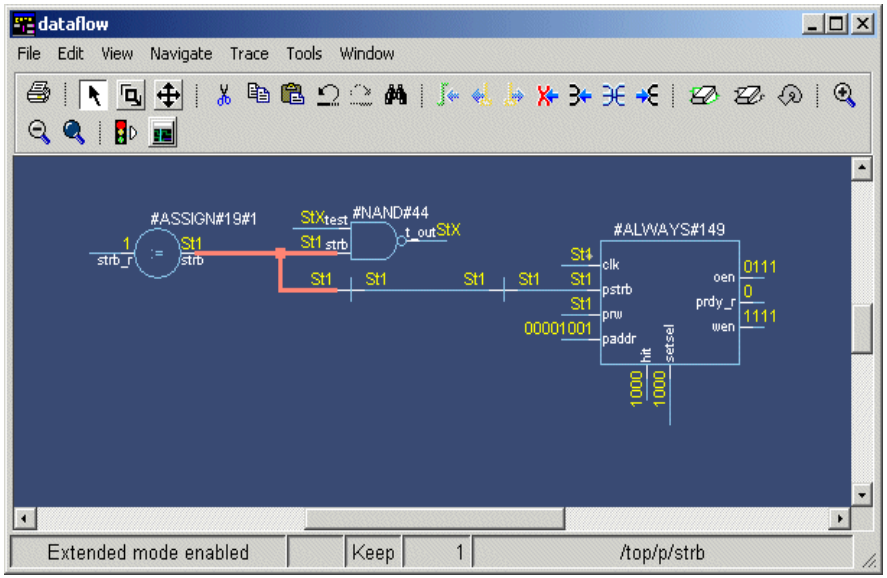


Figure 45: Expanding the view to display connected processes



Tracing events

Another useful debugging feature is tracing events that contribute to an unexpected output value. Using the Dataflow window's embedded wave viewer, you can trace backward from a transition to see which process or signal caused the unexpected output.

1 Add an object to the Dataflow window.

   a Make sure instance *p* is selected in the sim tab of the Main window.

   b Drag signal *t_out* from the Objects pane into the Dataflow window.

   c Select **View > Show Wave** in the Dataflow window to open the wave viewer (Figure 46). You may need to increase the size of the Dataflow window and scroll the panes to see everything.

2 Trace the inputs of the nand gate.

   a Select process *#NAND#44* (labeled *line_62* in the VHDL version) in the dataflow pane.

   All input and output signals of the process are displayed automatically in the wave viewer.

   b In the wave view, scroll to time 2785 ns (the last transition of signal *t_out*) .

   c Click on the last transition of signal *t_out* to set a cursor (Figure 47).

Figure 46: The embedded wave viewer pane

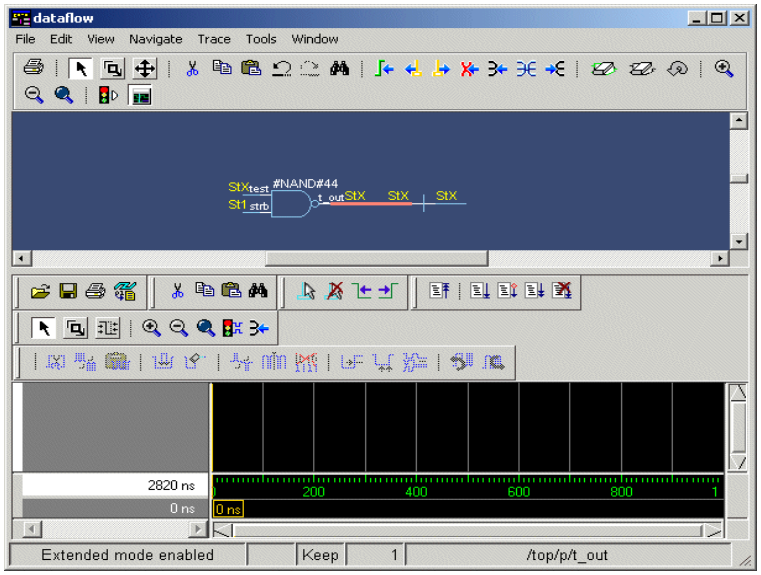
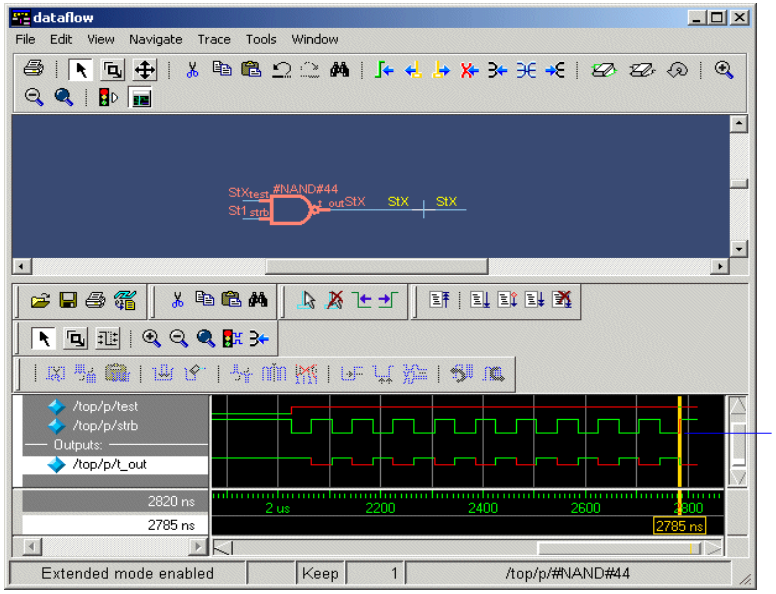


Figure 47: Signals added to the wave viewer automatically



d Select **Trace > Trace next event** to trace the first contributing event.

ModelSim adds a cursor marking the last event, the transition of the strobe to 0 at 2745 ns, which caused the output of 1 on *t_out* (Figure 48).

e Select **Trace > Trace next event** two more times.

f Select **Trace > Trace event set**.

The dataflow pane sprouts to the preceding process and shows the input driver of signal *strb* (Figure 49). Notice too that the wave viewer now shows the input and output signals of the newly selected process.

You can continue tracing events through the design in this manner: select **Trace next event** until you get to a transition of interest in the wave viewer, and then select **Trace event set** to update the dataflow pane.

3 Select **File > Close** to close the Dataflow window.

Figure 48: Cursor in wave viewer marking last event

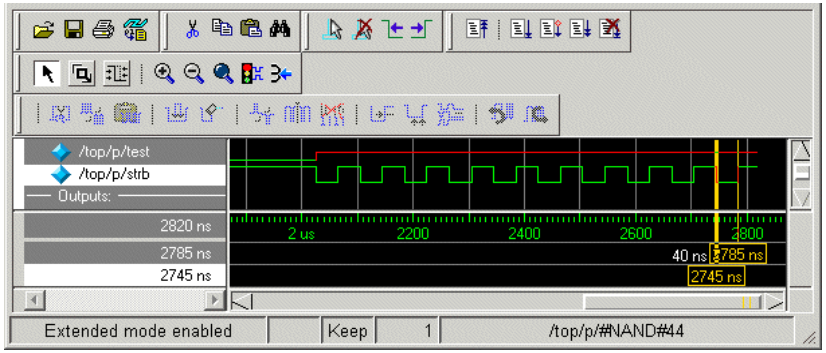
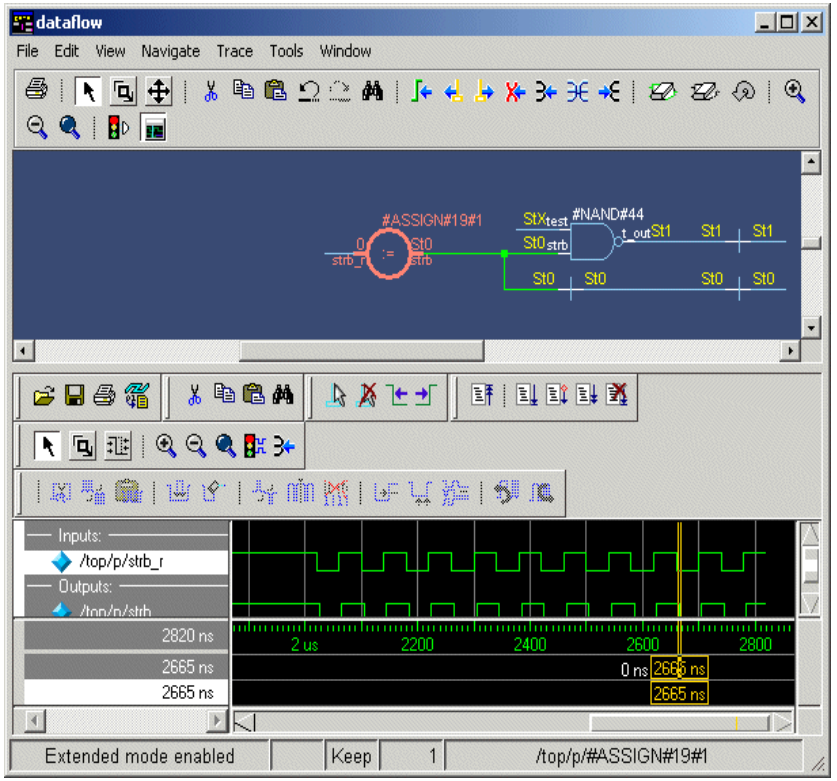


Figure 49: Tracing the event set



Tracing an 'X' (unknown)

The Dataflow window lets you easily track an unknown value (X) as it propagates through the design. The Dataflow window is linked to the stand-alone Wave window, so you can view signals in the Wave window and then use the Dataflow window to track the source of a problem. As you traverse your design in the Dataflow window, appropriate signals are added automatically to the Wave window.

1 View *t_out* in the Wave and Dataflow windows.

a Scroll in the Wave window until you can see */top/p/t_out*.

t_out goes to an unknown state at 2065 ns and continues transitioning between 1 and unknown for the rest of the run (Figure 50). The red color of the waveform indicates an unknown value.

b Double-click the last transition of signal *t_out* at 2785 ns.

This automatically opens the Dataflow window and displays *t_out*, its associated process, and its waveform. You may need to increase the size of the Dataflow window and scroll the panes to see everything.

c Move the cursor in the Wave window.

As previously mentioned the Wave and Dataflow windows are designed to work together. As you move the cursor in the Wave, the value of *t_out* changes in the Dataflow window.

d Move the cursor to a time when *t_out* is unknown (e.g., 2724 ns).

2 Trace the unknown.

a In the Dataflow window, make sure *t_out* is selected and then select **Trace > ChaseX**.

The design expands to show the source of the unknown (Figure 51). In this case there is a HiZ (U in the VHDL version) on input signal *test_in* and a 0 on input signal_*rw* (*bar_rw* in the VHDL version), so output signal *test2* resolves to an unknown.

Scroll to the bottom of the Wave window, and you will see that all of the signals contributing to the unknown value have been added.

3 Clear the Dataflow window before continuing.

Figure 50: A signal with unknown values

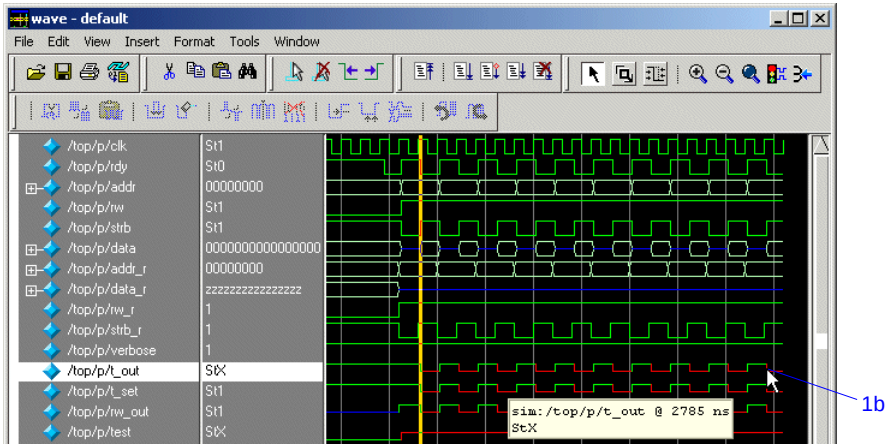
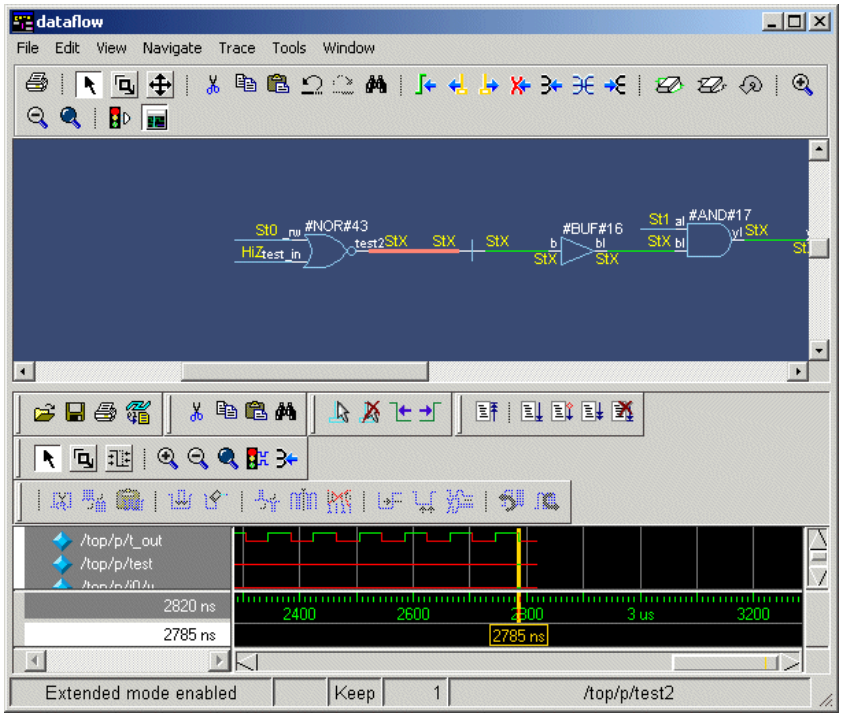


Figure 51: ChaseX identifies the cause of the unknown on *t_out*



Displaying hierarchy in the Dataflow window

You can display connectivity in the Dataflow window using hierarchical instances. You enable this by modifying the options prior to adding objects to the window.

1 Change options to display hierarchy.

   a Select **Tools > Options** from the Dataflow window menu bar.

   b Check **Show Hierarchy** and then click **OK** (Figure 52).

2 Add signal *t_out* to the Dataflow window.

   a Type **add dataflow /top/p/t_out** at the VSIM> prompt.

   The Dataflow window will display *t_out* and all hierarchical instances (Figure 53).

Figure 52: The Dataflow options dialog

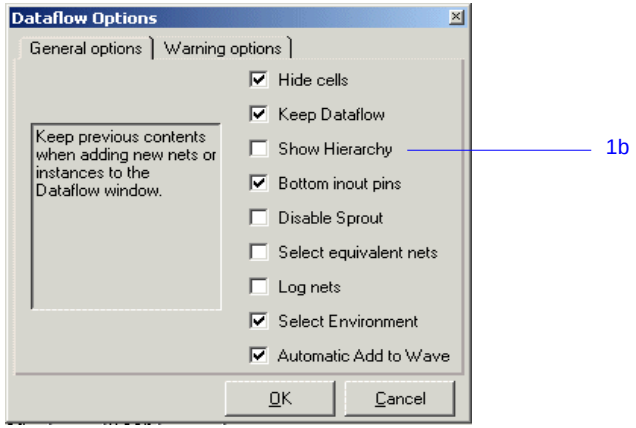
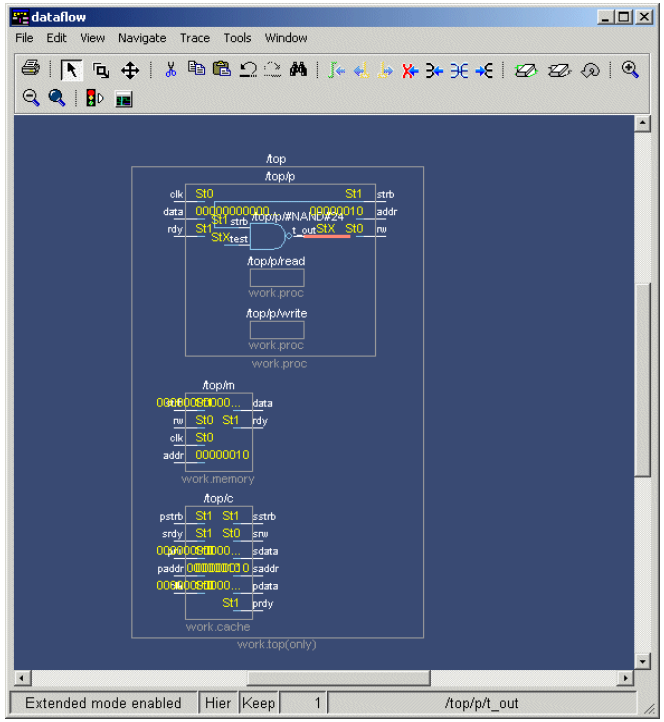


Figure 53: Dataflow window displaying with hierarchy



Lesson Wrap-up

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

1 Type **quit -sim** at the VSIM> prompt.

Lesson 7 - Viewing and initializing memories

Topics

The following topics are covered in this lesson:

Introduction

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

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

Design files for this lesson

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

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

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

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

Related reading

ModelSim GUI Reference – "Memory windows" (GR-140)

ModelSim Command Reference – **mem display** (CR-145), **mem load** (CR-148), **mem save** (CR-151), **radix** (CR-182) commands

Compiling and loading the design

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

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

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

2 Start ModelSim and change to the exercise directory.

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

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

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

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

3 Create the working library and compile the design.

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

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

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

4 Load the design.

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

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

Viewing a memory

Memories can be viewed via the ModelSim GUI.

1 Open a Memory instance.

a Select **View > Debug Windows > Memory**.

The Memories tab opens in the Workspace pane (Figure 54) and lists the memories in the current design context (*ram_tb*) with the range, depth, and width of each memory.

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

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

A **mem** tab is created in the MDI frame to display the memory contents. The data are all **X** (**0** in VHDL) since you have not yet simulated the design. The first column (blue hex characters) lists the addresses (Figure 55), and the remaining columns show the data values.

d Double-click instance */ram_tb/spram2/mem* in the Memories tab of the Workspace,

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

Figure 54: Viewing the memories tab in the Main window workspace

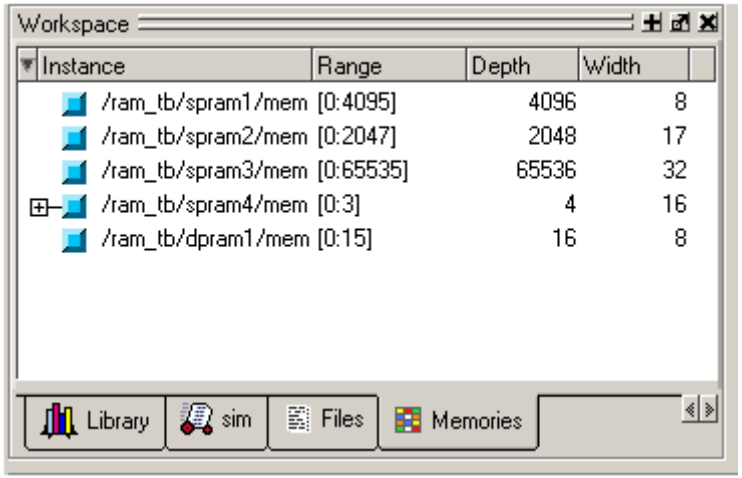
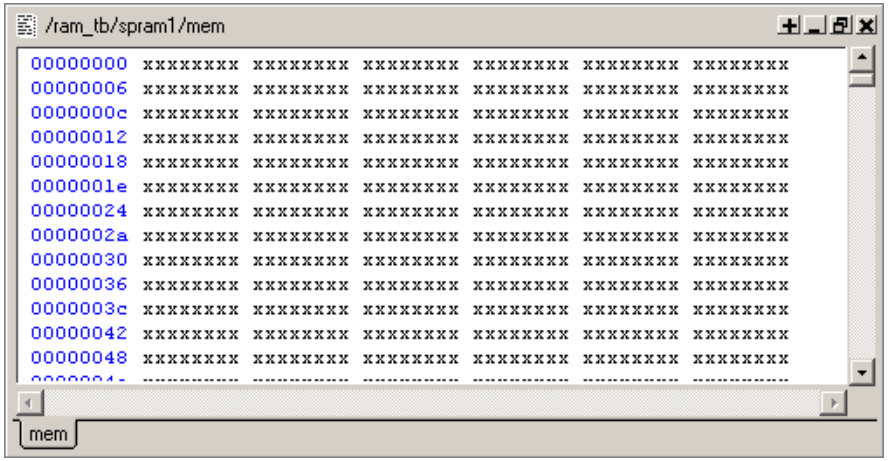


Figure 55: The mem tab in the MDI pane shows instance /ram_tb/spram1/mem



2 Simulate the design.

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

b Click the **mem** tab of the MDI frame to bring the */ram_tb/spram1/mem* instance to the foreground (Figure 56).

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

3 Let's change the address radix and the number of words per line for instance */ram_tb/spram1/mem*.

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

The Properties dialog box opens (Figure 57).

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

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

d Click OK.

You can see the results of the settings in Figure 58. 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 56: Memory display updates with simulation

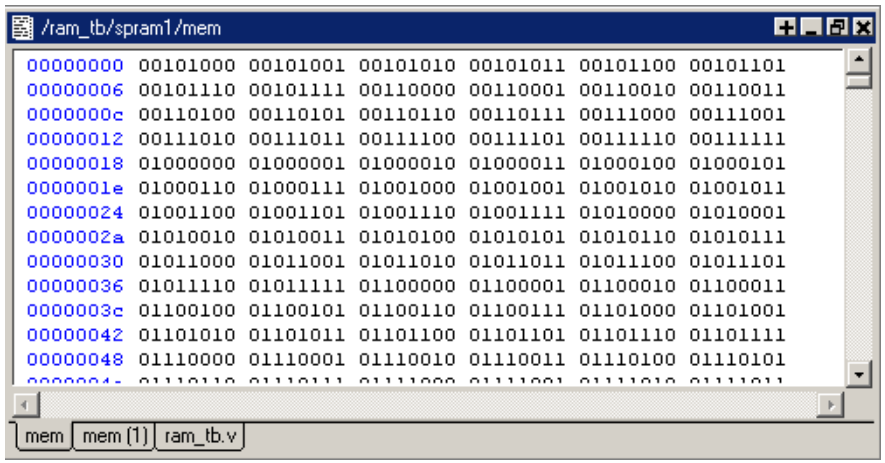
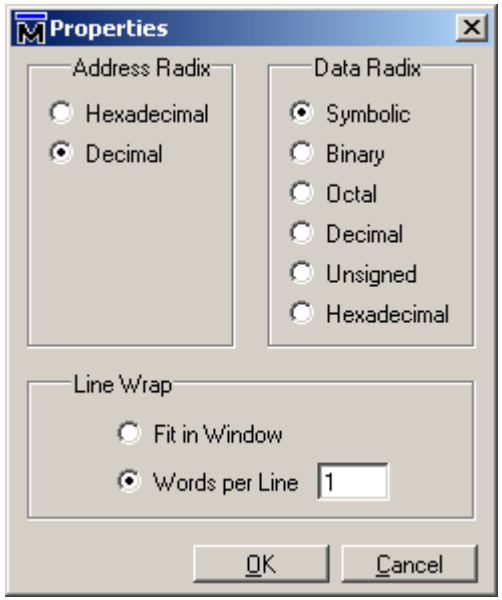


Figure 57: Changing the address radix



Navigating within the memory

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

1 Use Goto to find a specific address.

a Right-click anywhere in address column and select **Goto** (Figure 59).

The Goto dialog box opens in the data pane.

b Type **30** in the dialog box.

c Click OK.

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

Figure 58: Memory window: new address radix and line length

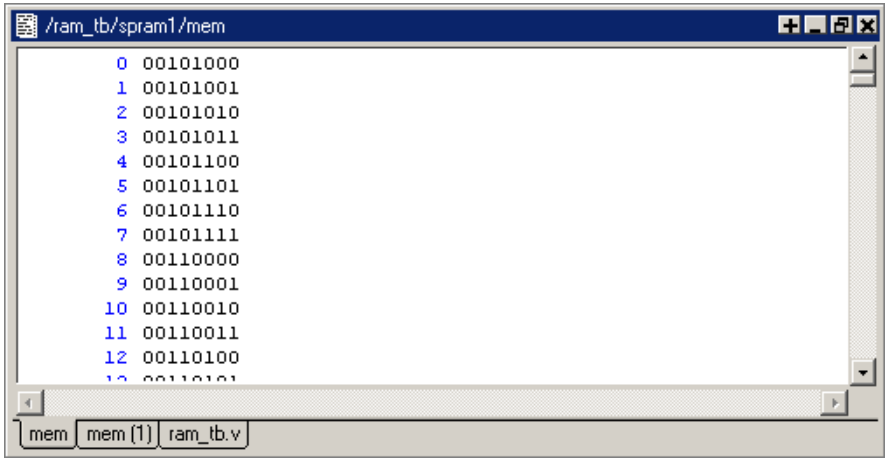
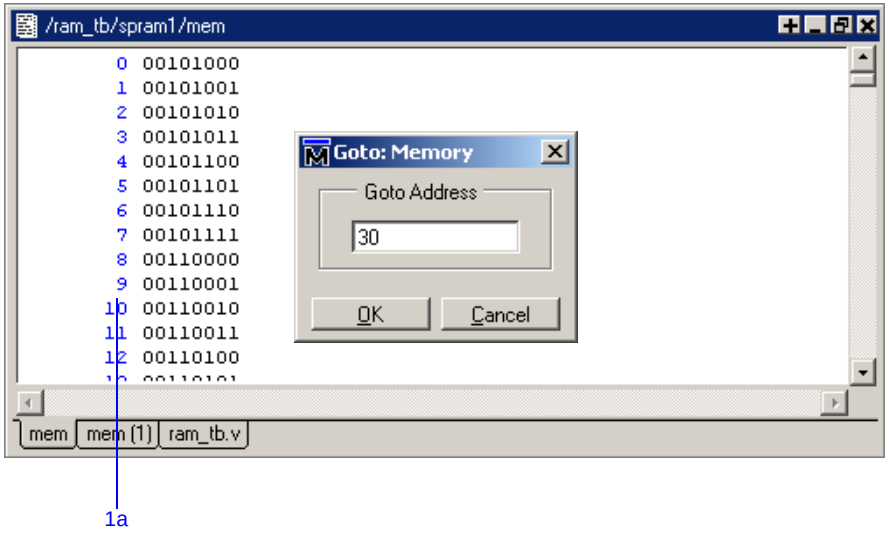


Figure 59: The Goto dialog box



2 Edit the address location directly.

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

a Double click any address in the address column.

b Enter any desired address. (Figure 60)

c Press <Enter> on your keyboard.

The pane scrolls to that address.

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

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

The Find in dialog box opens (Figure 61).

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

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

c Click Close to close the dialog box.

Figure 60: Edit the address directly

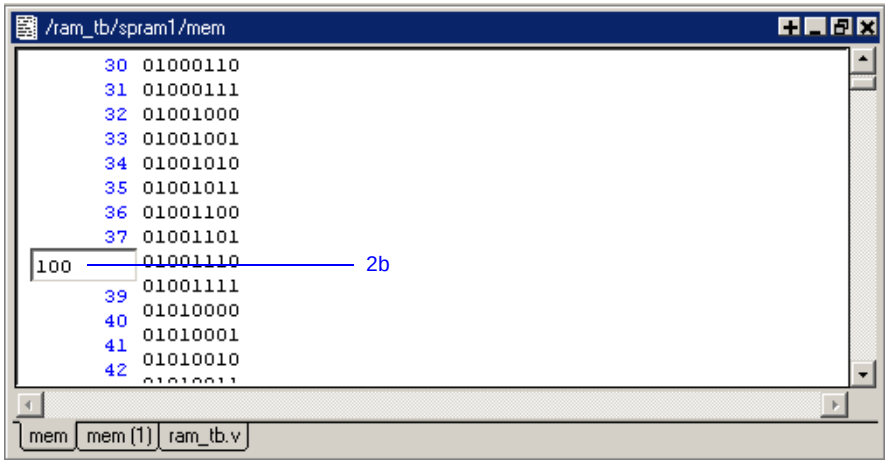
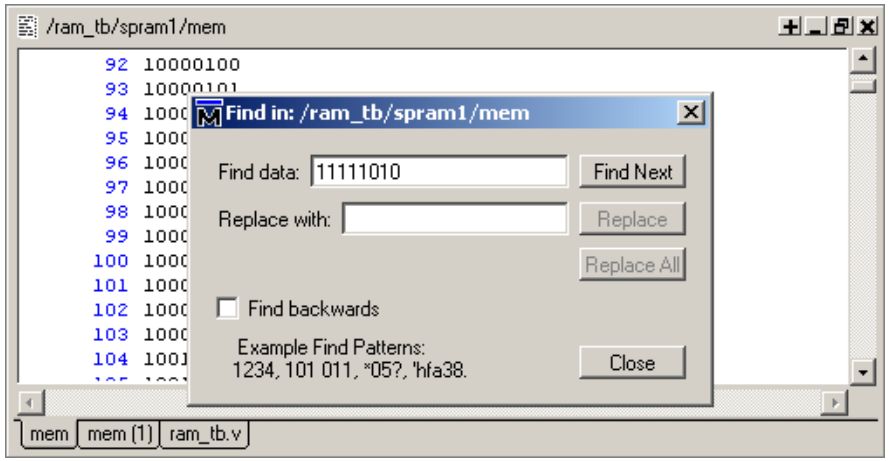


Figure 61: Find in: searching for data value



Saving memory contents to a file

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

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

a Make sure */ram_tb/spram1/mem* is open and selected in the MDI frame.

b Select **File > Save** to bring up the Save Memory dialog box (Figure 62).

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

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

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

f Click OK.

You can view the saved file in any editor.

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

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

a Select the **mem(1)** tab in the MDI pane to see the data for the */ram_tb/spram2/mem* instance.

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

c In the Properties dialog, set the Address Radix to Decimal and the Data Radix to Binary. Click OK to accept the changes and close the dialog.

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

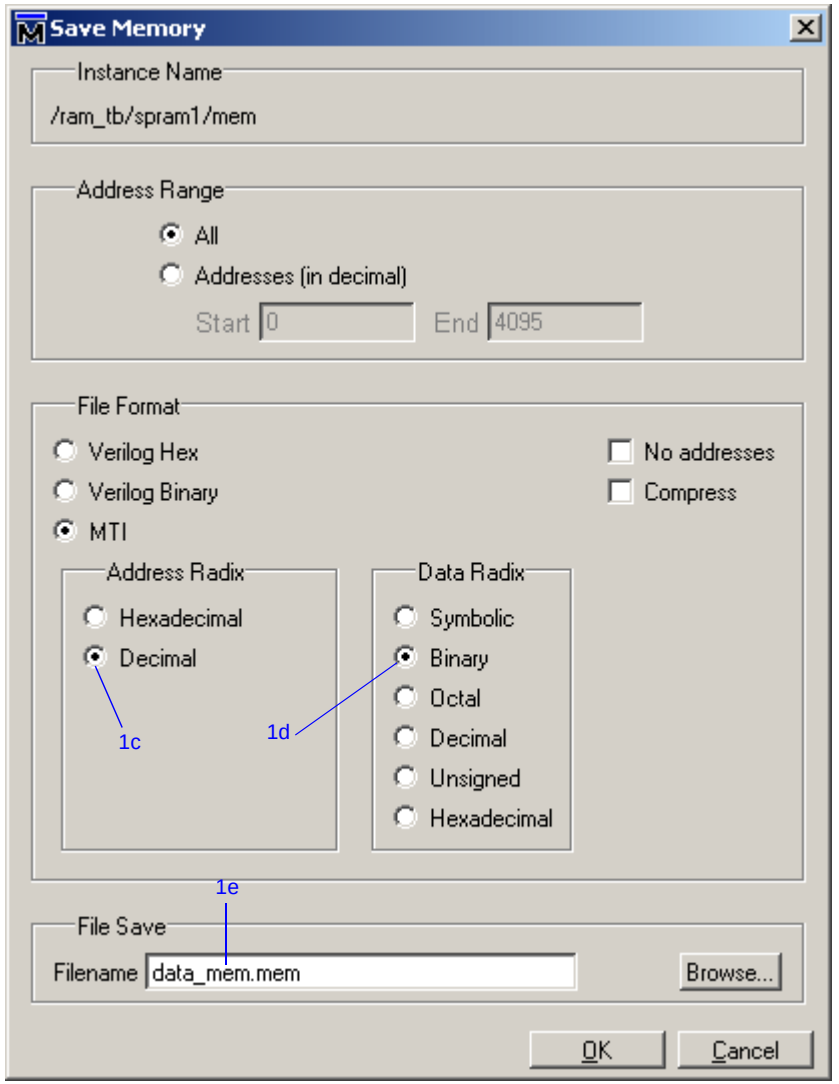
e Specify a Start address of **0** and End address of **250.**

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

g Click **No addresses** to create a memory pattern that you can use to relocate somewhere else in the memory, or in another memory.

h Enter the file name as **reloc.mem**, then click OK to save the memory contents and close the dialog.

Figure 62: Save Memory dialog box



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

Initializing a memory

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

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

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

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

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

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

   c Change the Address Radix to **Decimal** and Data Radix to **Binary** and click OK.

2 Initialize *spram3* from a file.

   a Right-click anywhere in the data column and select **Load** to bring up the Load Memory dialog box (Figure 63).

   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 64).

Figure 63: Load Memory dialog box

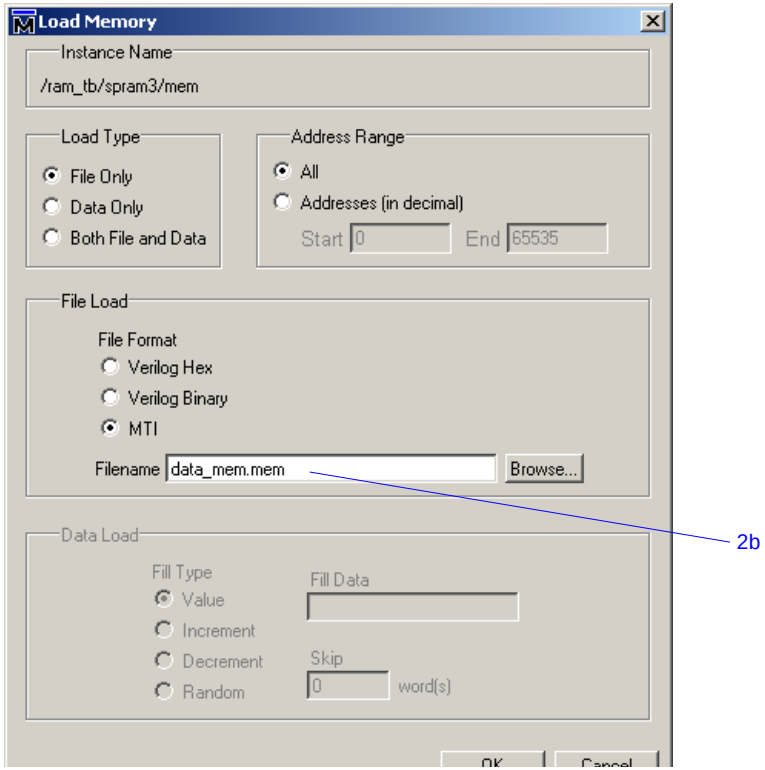
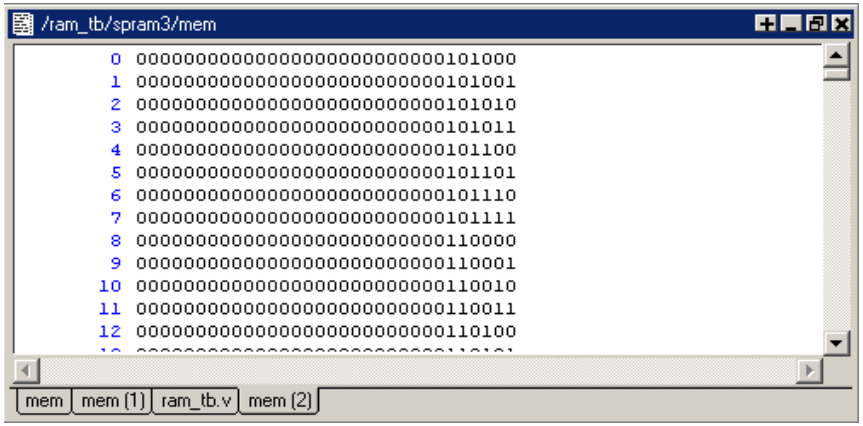


Figure 64: Initialized memory from file and fill pattern



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

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

   a Right-click in the data column of the **mem(2)** tab and select **Load** to bring up the Load Memory dialog box (Figure 65).

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

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

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

   d For File Load, enter **reloc.mem** in the Filename field.

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

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

   g Click OK.

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

You can see the specified range of addresses overwritten with the new data. Also, you can see the incrementing data beginning at address 251 (Figure 66).

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

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

Figure 65: Loading a relocatable memory file

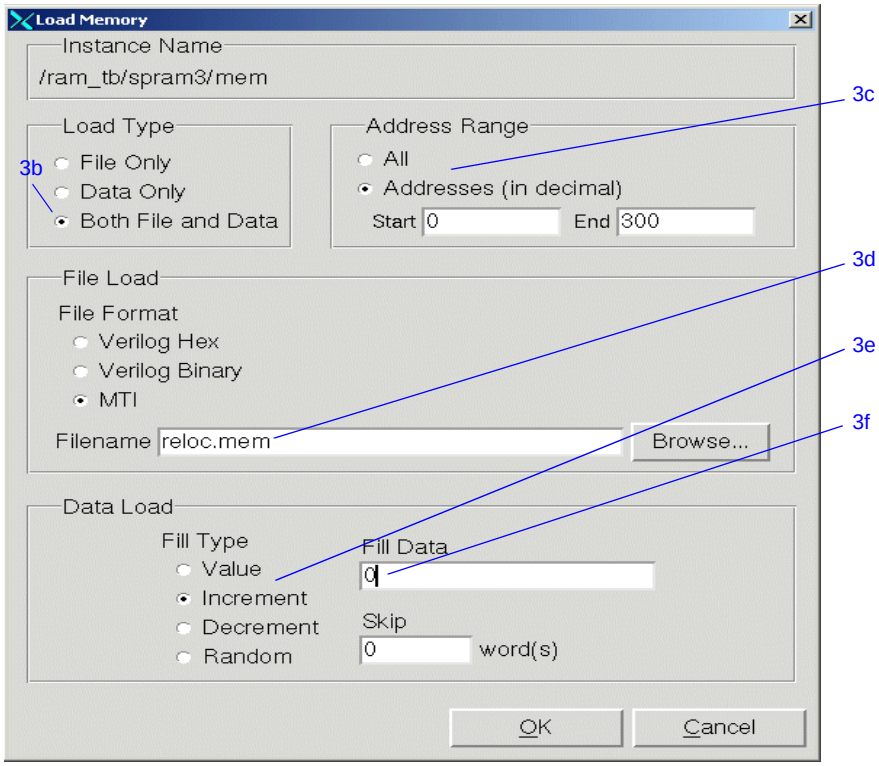
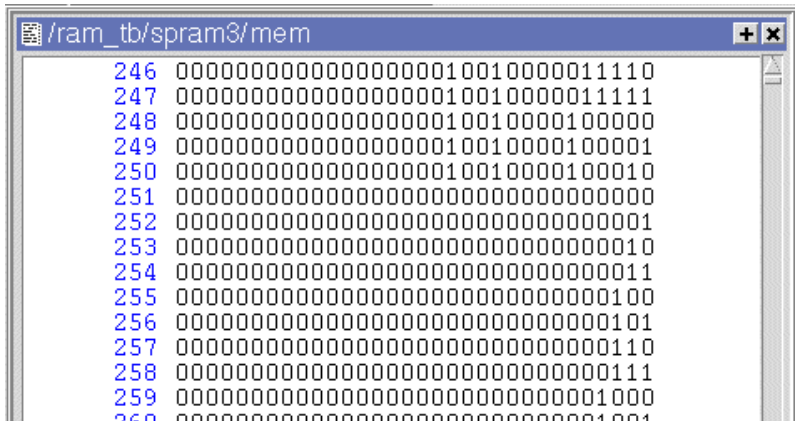


Figure 66: Overwritten values in memory instance



Interactive debugging commands

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

1 Open a memory instance and change its display characteristics.

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

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

   c Change the Data Radix to **Hexadecimal**.

   d Select **Words per line** and enter **2**.

   e Click OK.

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

   a Right-click in the data column of */ram_tb/dpram1/mem* contents pane and select **Change** to open the Change Memory dialog (Figure 68).

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

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

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

   e Click OK.

   The data in the specified range are replaced with a generated random fill pattern (Figure 69).

Figure 67: Original memory contents

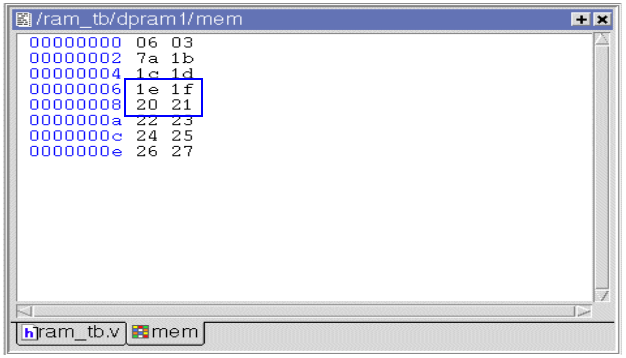


Figure 68: Changing memory contents for a range of addresses

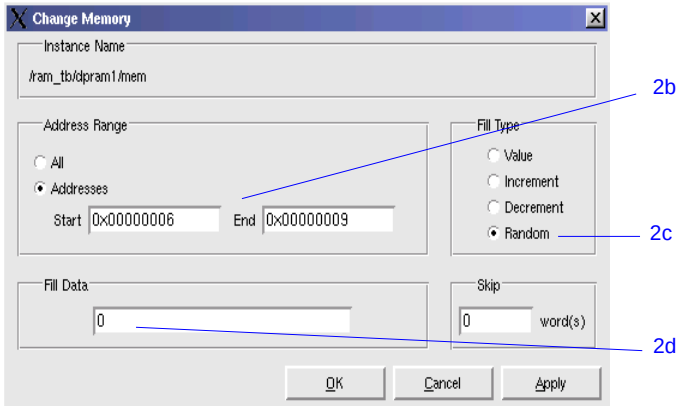
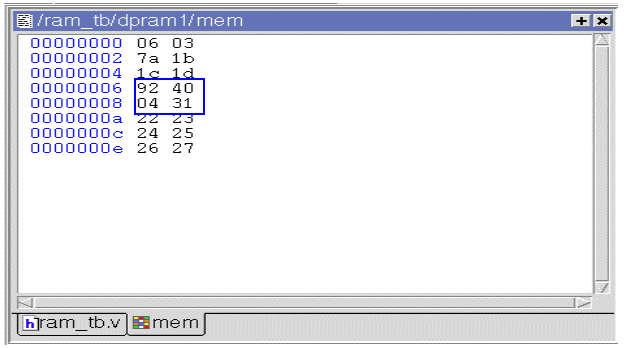


Figure 69: Random contents of a range of addresses



3 Change contents by highlighting.

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

a Highlight the data for the addresses **0x0000000c:0x0000000e**, as shown in Figure 70.

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

This brings up the Change dialog box (Figure 71). Note that the Addresses field is already populated with the range you highlighted.

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

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

e Click OK.

The data in the address locations change to the values you entered (Figure 72).

4 Edit data in place.

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

a Double click any value in the Data column.

b Enter the desired value and press <Enter>.

c When you are finished editing all values, press the <Enter> key on your keyboard to exit the editing mode.

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

Figure 70: Changing contents by highlighting

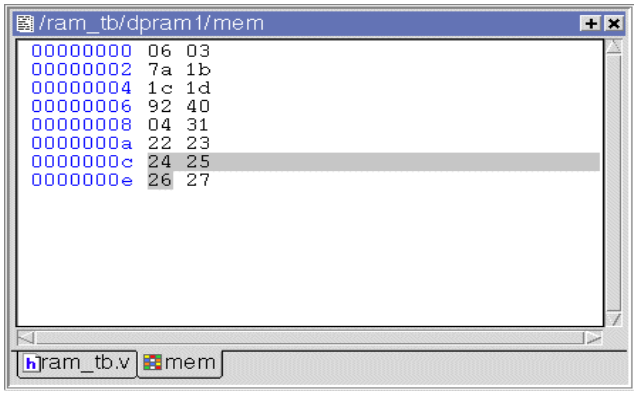


Figure 71: Entering data to change

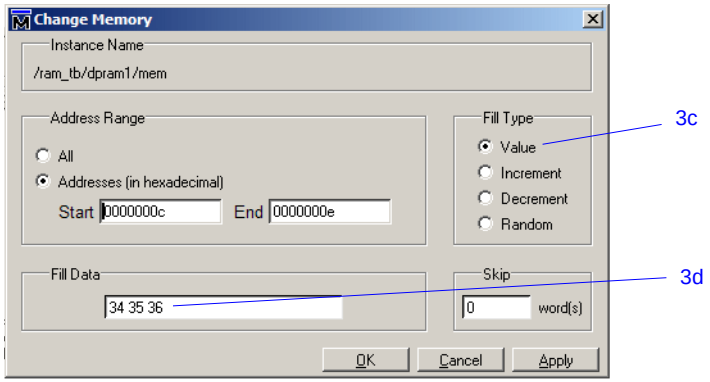
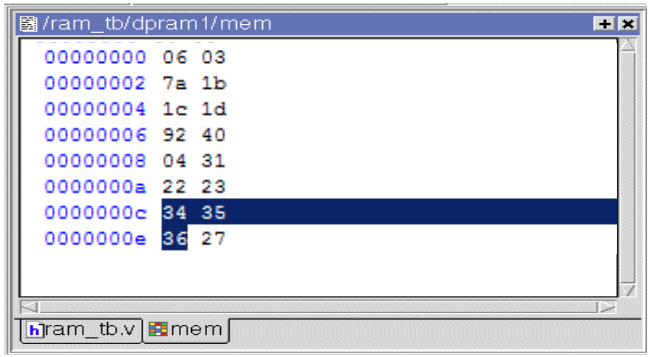


Figure 72: Changed contents for specified addresses



Lesson Wrap-up

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

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

Lesson 8 - Simulating with Code Coverage

Topics

The following topics are covered in this lesson:

Note: The functionality described in this tutorial requires a coverage license feature in your ModelSim license file. Please contact your Mentor Graphics sales representative if you currently do not have such a feature.

Introduction

ModelSim Code Coverage gives you graphical and report file feedback on which executable statements, branches, conditions, and expressions in your source code have been executed. It also measures bits of logic that have been toggled during execution.

Design files for this lesson

The sample design for this lesson consists of a finite state machine which controls a behavioral memory. The testbench *test_sm* provides stimulus.

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

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

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

This lesson uses the Verilog version in the examples. If you have a VHDL license, use the VHDL version instead. When necessary, we distinguish between the Verilog and VHDL versions of the design.

Related reading

ModelSim User's Manual – Chapter 12 - Measuring code coverage (UM-289)

Compiling the design

Enabling Code Coverage is a two step process–first, you compile the files and identify which coverage statistics you want; second, you load the design and tell ModelSim to produce those statistics.

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>/modeltech/examples/coverage/verilog* to the new directory.

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

2 Start ModelSim and change to the exercise directory.

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

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

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

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

3 Create the working library.

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

4 Compile the design files.

a For Verilog – Type **vlog -cover bct sm.v sm_seq.v beh_sram.v test_sm.v** at the ModelSim> prompt.

For VHDL – Type **vcom -cover bct sm.vhd sm_seq.vhd sm_sram.vhd test_sm.vhd** at the ModelSim> prompt.

The **-cover bct** argument instructs ModelSim that you want branch, condition, and toggle coverage statistics (statement coverage is included by default). See "Enabling code coverage" (UM-293) for more information on the available coverage types.

Loading and running the design

1 Load the design.

a Type **vsim -coverage test_sm** at the ModelSim> prompt.

2 Run the simulation

b Type **run 1 ms** at the VSIM> prompt.

When you load a design with Code Coverage enabled, ModelSim adds several columns to the Files and sim tabs in the Workspace (Figure 73). ModelSim also displays three Code Coverage panes in the Main window (Figure 74):

- **Missed Coverage**

  Displays the selected file's un-executed statements, branches, conditions, and expressions and signals that have not toggled.

- **Instance Coverage**

  Displays statement, branch, condition, expression and toggle coverage statistics for each instance in a flat, non-hierarchical view.

- **Details**

  Shows details of missed coverage such as truth tables or toggle details.

Another coverage-related pane is the Current Exclusions pane. Select **View > Code Coverage > Current Exclusions** to display that pane.

- **Current Exclusions**

  Lists all files and lines that are excluded from coverage statistics (see "Excluding lines and files from coverage statistics" (T-94) for more information).

These panes can be re-sized, rearranged, and "undocked" to make the data more easily viewable. To resize a pane, click-and-drag on the top or bottom border. To move a pane, click-and-drag on the double-line to the right of the pane name. To undock a pane you can select it then drag it out of the Main window, or you can click the Dock/Undock Pane button in the header bar (top right). To redock the pane, click the Dock/Undock Pane button again.

We will look at these panes more closely in the next exercise. For complete details on each pane, see "Code coverage panes" (GR-99).

Figure 73: Coverage columns in the Main window Workspace

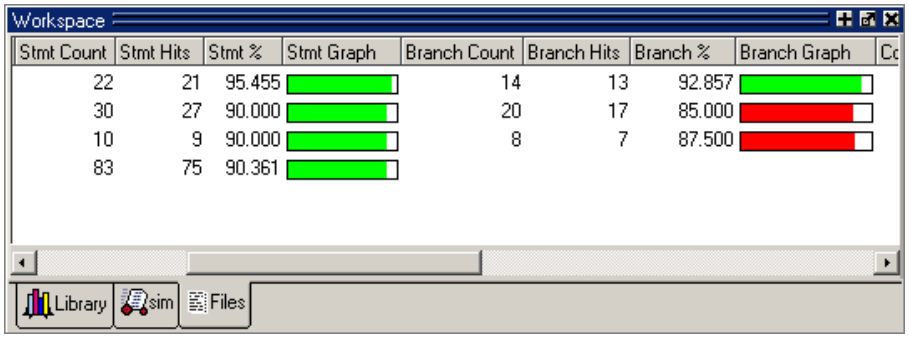
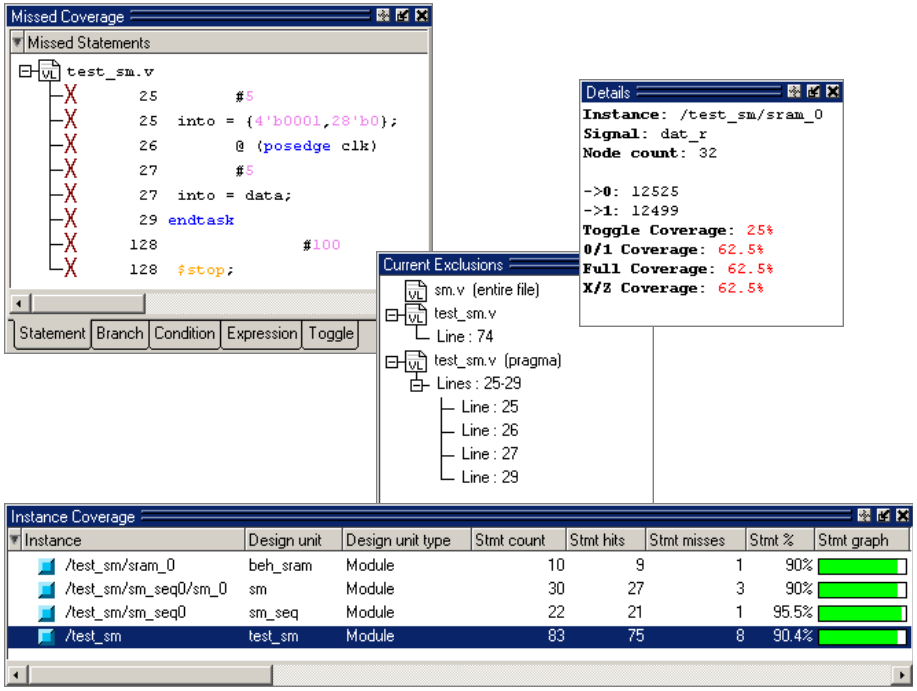


Figure 74: Coverage panes



Viewing statistics in the Main window

Let's take a look at the data in these various panes.

1 View statistics in the Workspace pane.

a Select the sim tab in the Workspace and scroll to the right.

Coverage statistics are shown for each object in the design.

b Select the Files tab in the Workspace and scroll to the right.

Each file in the design shows summary statistics for statements, branches, conditions, and expressions.

c Click the right-mouse button on any column name and select an object from the list (Figure 75).

Whichever column you selected is hidden. To redisplay the column, right-click again and select that column name. The status of which columns are displayed or hidden is persistent between invocations of ModelSim.

2 View statistics in the Missed Coverage pane.

a Select different files from the Files tab of the Workspace.

The Missed Coverage pane updates to show statistics for the selected file (Figure 76).

b Select any entry in the Statement tab to display that line in the Source window.

Figure 75: Right click a column heading to hide or show columns

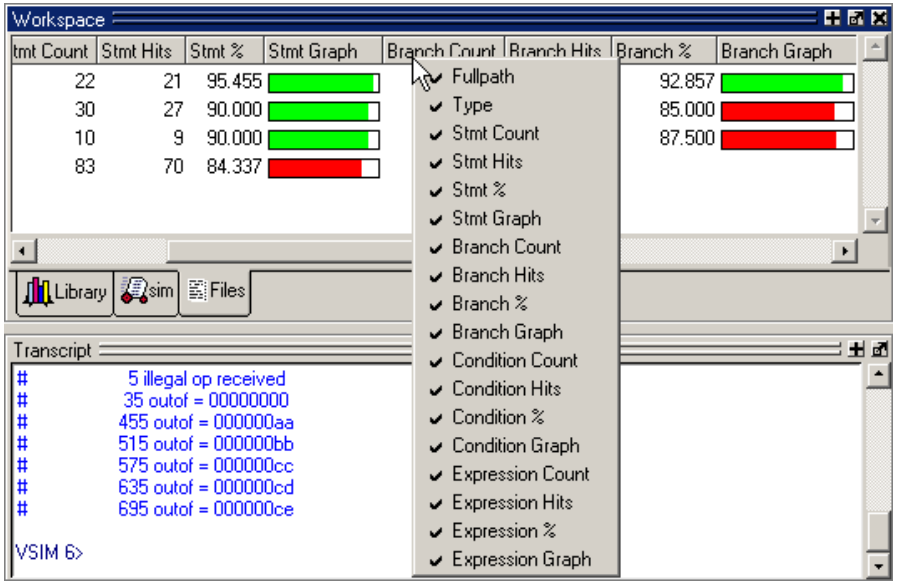
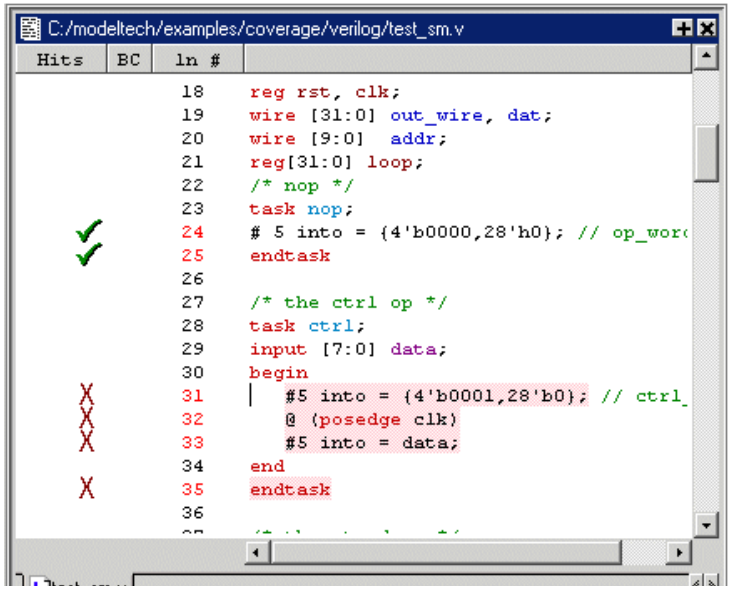


Figure 76: Statement statistics in the Missed Coverage pane



3 View statistics in the Details pane.

a Select the Toggle tab in the Missed Coverage pane.

If the Toggle tab isn't visible, you can do one of two things: 1) widen the pane by clicking-and-dragging on the pane border; 2) if your mouse has a middle button, click-and-drag the tabs with the middle mouse button.

b Select any object in the Toggle tab to see details in the Details pane (Figure 77).

4 View instance coverage statistics.

The Instance Coverage pane displays coverage statistics for each instance in a flat, non-hierarchical view (Figure 78). Select any instance in the Instance Coverage pane to see its source code displayed in the Source window.

Figure 77: Details pane showing toggle coverage statistics

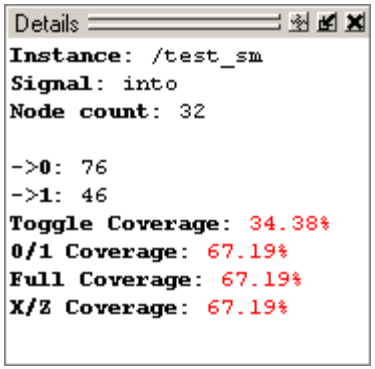
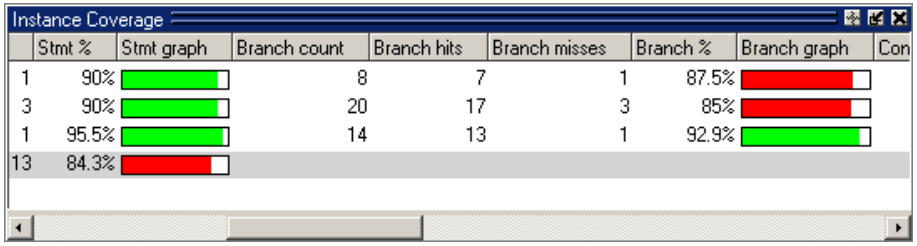


Figure 78: The Instance Coverage pane



Viewing statistics in the Source window

In the previous section you saw that the Source window and the Main window coverage panes are linked. You can select objects in the Main window panes to view the underlying source code in the Source window. Furthermore, the Source window contains statistics of its own.

1 View coverage statistics for *test_sm* in the Source window.

a Make sure *test_sm* is selected in the sim tab of the Workspace.

b In the Statement tab of the Missed Coverage pane, expand *test_sm.v* if necessary and select any line (Figure 79).

The Source window opens in the MDI frame with the line you selected highlighted (Figure 80).

c Switch to the Source window.

The table below describes the various icons.

Icon	Description
green checkmark	indicates a statement that has been executed
red X	indicates that a statement in that line has not been executed (zero hits)
green E	indicates a line that has been excluded from code coverage statistics
red X_T or X_F	indicates that a true or false branch (respectively) of a conditional statement has not been executed

Figure 79: Selecting a line in the Missed Coverage pane

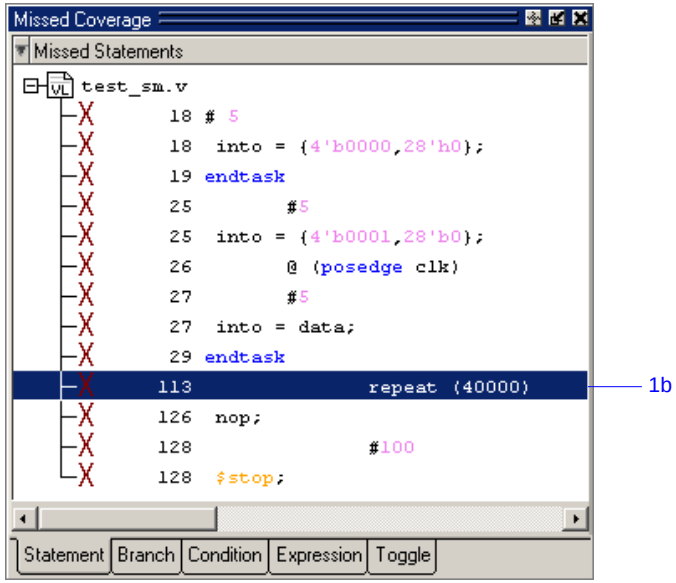
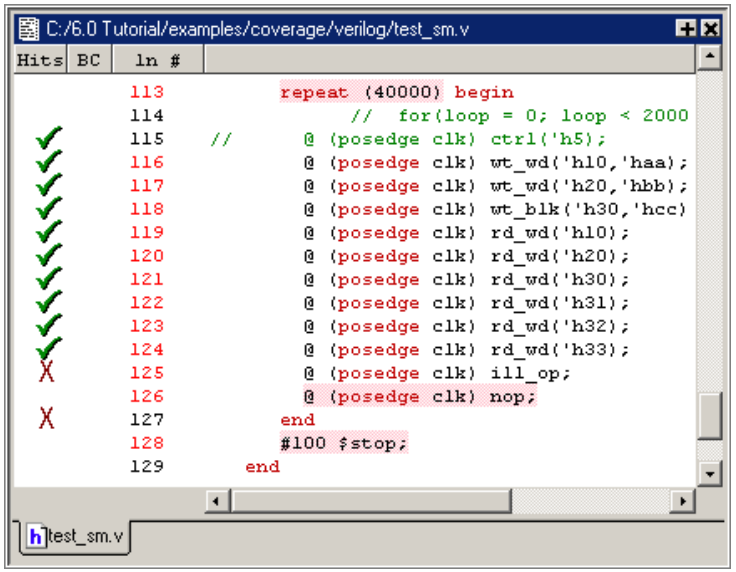


Figure 80: Coverage statistics in the Source window



d Hover your mouse pointer over a line of code with a green checkmark.

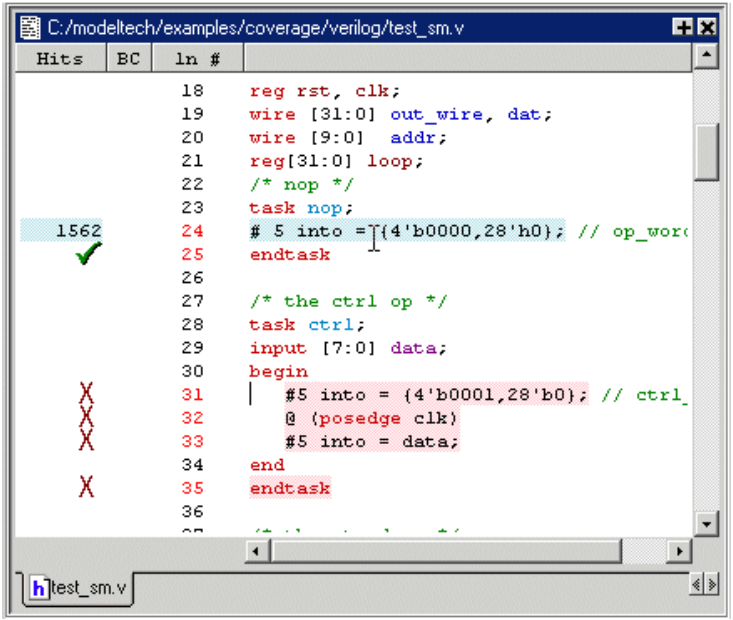
The icons change to numbers that indicate how many times the statements and branches in that line were executed (Figure 81). In this case line 24 was executed 1562 times.

e Select **Tools > Code Coverage > Show coverage numbers**.

The icons are replaced by execution counts on every line. An ellipsis (...) is displayed whenever there are multiple statements on the line. Hover the mouse pointer over a statement to see the count for that statement.

f Select **Tools > Code Coverage > Hide coverage numbers** to return to icon display.

Figure 81: Coverage numbers shown by hovering the mouse pointer



Viewing toggle statistics in the Objects pane

Toggle coverage counts each time a logic node transitions from one state to another. Earlier in the lesson you enabled two-state toggle coverage (0 -> 1 and 1-> 0) with the **-cover t** argument. Alternatively, you can enable six-state toggle coverage using the **-cover x** argument. See "Toggle coverage" (UM-299) for more information.

1 View toggle data in the Objects pane of the Main window.

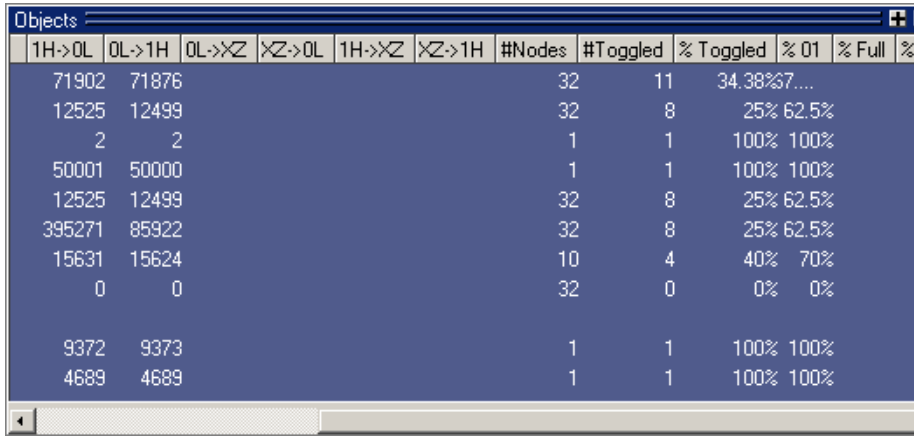
   a Select *test_sm* in the sim tab of the Main window.

   b If the Objects pane isn't open already, select **View > Debug Windows > Objects**.

   c Scroll to the right and you will see the various toggle coverage columns (Figure 82).

   The blank columns show data when you have extended toggle coverage enabled.

Figure 82: Toggle coverage columns in the Source window



Excluding lines and files from coverage statistics

ModelSim allows you to exclude lines and files from code coverage statistics. You can set exclusions with the GUI, with a text file called an "exclusion filter file", or with "pragmas" in your source code. Pragmas are statements that instruct ModelSim to not collect statistics for the bracketed code. See "Excluding objects from coverage" (UM-303) for more details on exclusion filter files and pragmas.

1 Exclude a line via the Missed Coverage pane.

   a Right click a line in the Missed Coverage pane and select **Exclude Selection**. (You can also exclude the selection for the current instance only by selecting Exclude Selection For Instance <inst_name>.)

2 Exclude an entire file.

   a In the Files tab of the Workspace, locate *sm.v* (or *sm.vhd* if you are using the VHDL example).

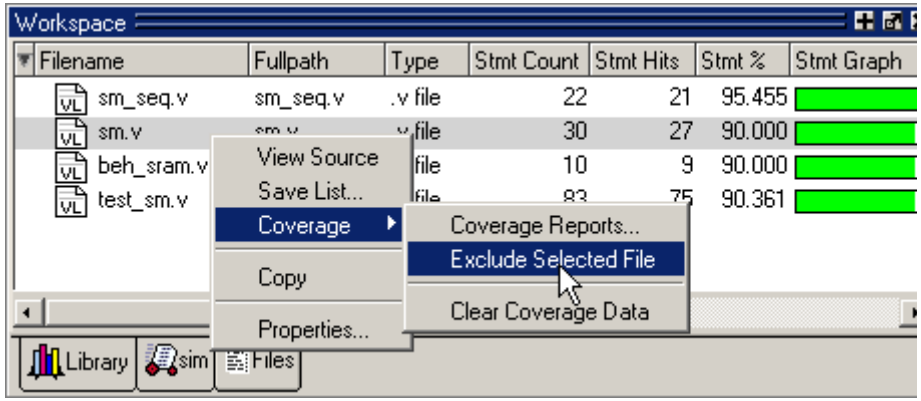
   b Right-click the file name and select **Coverage > Exclude Selected File** (Figure 83).

   The file is added to the Current Exclusions pane.

3 Cancel the exclusion of *sm.v*.

   a Right-click *sm.v* in the Current Exclusions pane and select **Cancel Selected Exclusions**.

Figure 83: Excluding an entire file via the GUI



Creating Code Coverage reports

You can create reports on the coverage statistics using either the menus or by entering commands in the Transcript pane. The reports are output to a text file regardless of which method you use.

To create coverage reports via the menus, do one of the following:

- select **Tools > Code Coverage > Reports** from the Main window menu
- right-click any object in the sim or Files tab of the Workspace and select **Code Coverage > Coverage Reports**
- right-click any object in the Instance Coverage pane and select **Code coverage reports** from the context menu

1 Create a report on all instances.

   a Select **Tools > Coverage > Reports** from the Main window toolbar.

   This opens the Coverage Report dialog (Figure 84).

   b Make sure **Report on all instances** and **No Filtering** are selected and then click OK.

   ModelSim creates a file *report.txt* in the current directory and displays the report in Notepad.

   c Close Notepad when you are done looking at the report.

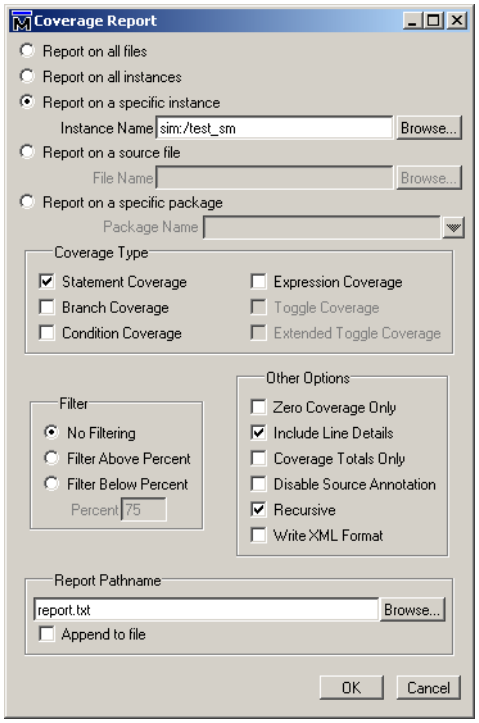
2 Create a summary report on all design files from the Transcript pane.

   a Type **coverage report -file cover.txt** at the VSIM> prompt.

   b Type **notepad cover.txt** at the VSIM> prompt to view the report.

   c Close Notepad when you are done reviewing the report.

Figure 84: The Coverage Report dialog



Lesson wrap-up

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

1 Type **quit -sim** at the VSIM> prompt.

Lesson 9 - Waveform Compare

Topics

The following topics are covered in this lesson:

Note: The functionality described in this tutorial requires a compare license feature in your ModelSim license file. Please contact your Mentor Graphics sales representative if you currently do not have such a feature.

Introduction

Waveform Compare computes timing differences between test signals and reference signals. The general procedure for comparing waveforms has four main steps:

1 Selecting the simulations or datasets to compare

2 Specifying the signals or regions to compare

3 Running the comparison

4 Viewing the comparison results

In this exercise you will run and save a simulation, edit one of the source files, run the simulation again, and finally compare the two runs.

Design files for this lesson

The sample design for this lesson consists of a finite state machine which controls a behavioral memory. The testbench *test_sm* provides stimulus.

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

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

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

This lesson uses the Verilog version in the examples. If you have a VHDL license, use the VHDL version instead. When necessary, we distinguish between the Verilog and VHDL versions of the design.

Related reading

"Waveform Compare" (UM-240), *Chapter 8 - WLF files (datasets) and virtuals* (UM-195)

Creating the reference dataset

The reference dataset is the *.wlf* file that the test dataset will be compared against. It can be a saved dataset, the current simulation dataset, or any part of the current simulation dataset.

In this exercise you will use a DO file to create the reference dataset.

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>/modeltech/examples/compare/verilog* to the new directory.

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

2 Start ModelSim and change to the exercise directory.

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

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

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

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

3 Execute the lesson DO file.

a Type **do gold_sim.do** at the ModelSim> prompt.

The DO file does the following:

- Creates and maps the work library
- Compiles the Verilog and VHDL files
- Runs the simulation and saves the results to a dataset named *gold.wlf*
- Quits the simulation

Feel free to open the DO file and look at its contents.

Creating the test dataset

The test dataset is the *.wlf* file that will be compared against the reference dataset. Like the reference dataset, the test dataset can be a saved dataset, the current simulation dataset, or any part of the current simulation dataset.

To simplify matters, you will create the test dataset from the simulation you just ran. However, you will edit the testbench to create differences between the two runs.

Verilog

1 Edit the testbench.

   a Select **File > Open** and open *test_sm.v*.

   b Scroll to line 122, which looks like this:

```
@ (posedge clk) wt_wd('h10,'haa);
```

   c Change the data pattern ’aa’ to ’ab’:

```
@ (posedge clk) wt_wd('h10,'hab);
```

   d Select **File > Save** to save the file.

2 Compile the revised file and rerun the simulation.

   a Type **do sec_sim.do** at the ModelSim> prompt.

   The DO file does the following:

   - Re-compiles the testbench
   - Adds waves to the Wave window
   - Runs the simulation

VHDL

1 Edit the testbench.

a Select **File > Open** and open *test_sm.vhd*.

b Scroll to line 151, which looks like this:

```
wt_wd ( 16#10#, 16#aa#, clk, into );
```

c Change the data pattern ’aa’ to ’ab’:

```
wt_wd ( 16#10#, 16#ab#, clk, into );
```

d Select **File > Save** to save the file.

2 Compile the revised file and rerun the simulation.

a Type **do sec_sim.do** at the ModelSim> prompt.

The DO file does the following:

- Re-compiles the testbench
- Adds waves to the Wave window
- Runs the simulation

Comparing the simulation runs

ModelSim includes a Comparison Wizard that walks you through the process. You can also configure the comparison manually with menu or command line commands.

1 Create a comparison using the Comparison Wizard.

a Select **Tools > Waveform Compare > Comparison Wizard**.

b Click the **Browse** button and select *gold.wlf* as the reference dataset (Figure 85).

Recall that *gold.wlf* is from the first simulation run.

c Leaving the test dataset set to **Use Current Simulation**, click **Next**.

d Select **Compare All Signals** in the second dialog and click **Next** (Figure 86).

e In the next three dialogs, click **Next**, **Compute Differences Now**, and **Finish**, respectively.

ModelSim performs the comparison and displays the compared signals in the Wave window.

Figure 85: First dialog of the Comparison Wizard

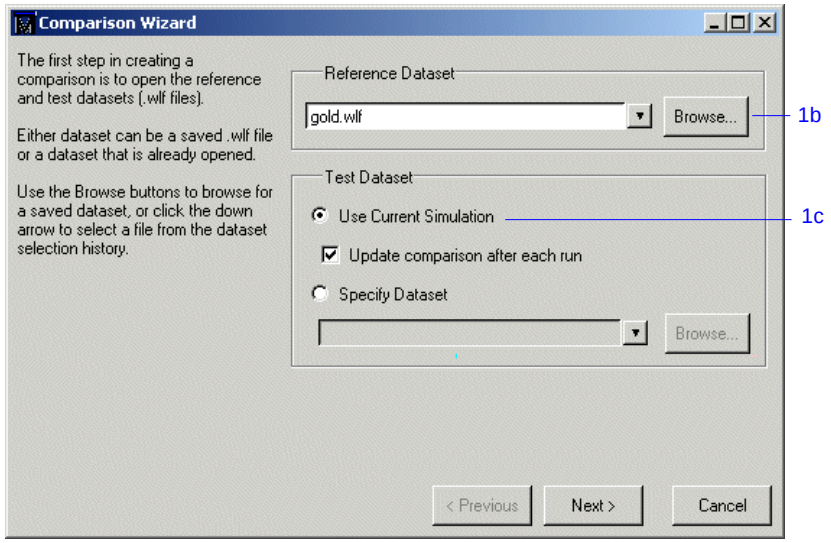
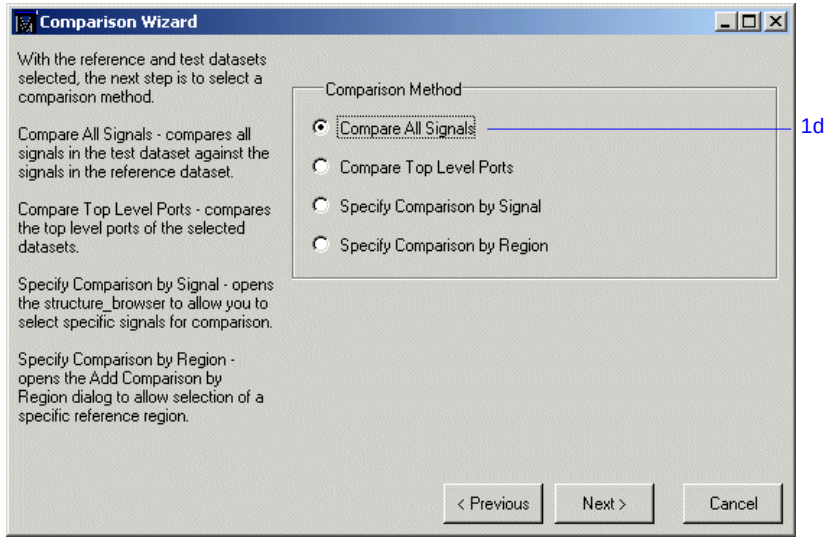


Figure 86: Second dialog of the Comparison Wizard



Viewing comparison data

Comparison data displays in three places within the ModelSim GUI: the Workspace pane of the Main window, the Wave window, and the List window.

Viewing comparison data in the Main window

Comparison information displays in three places in the Main window: the Compare tab in the Workspace pane shows the region that was compared; the Transcript shows the number of differences found between the reference and test datasets; and the Objects pane shows comparison differences if you select the object on the Compare tab (Figure 87).

Viewing comparison data in the Wave window

In the pathnames pane of the Wave window, a timing difference is denoted by a red X (Figure 88). Red areas in the waveform pane show the location of the timing differences, as do the red lines in the scrollbars. Annotated differences are highlighted in blue.

The Wave window includes six compare icons that let you quickly jump between differences (Figure 89). From left to right, the icons do the following: find first difference, find previous annotated difference, find previous difference, find next difference, find next annotated difference, find last difference. Use these icons to move the selected cursor.

The compare icons cycle through differences on all signals. To view differences for just the selected signal, use <tab> and <shift> - <tab>.

Figure 87: Comparison information in the Main window

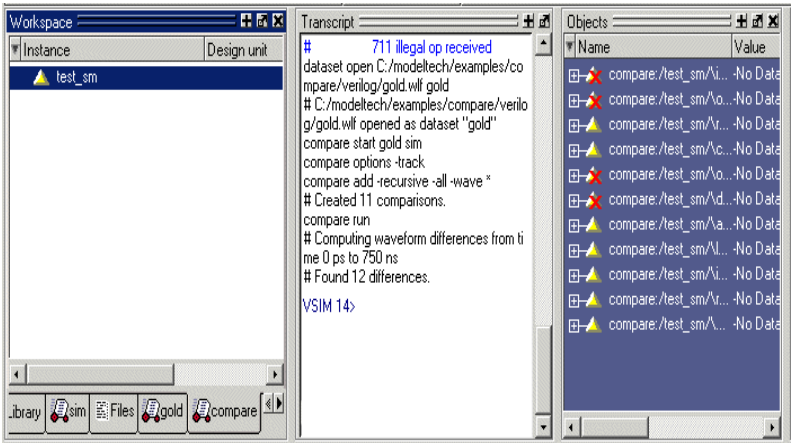


Figure 88: Comparison objects in the Wave window

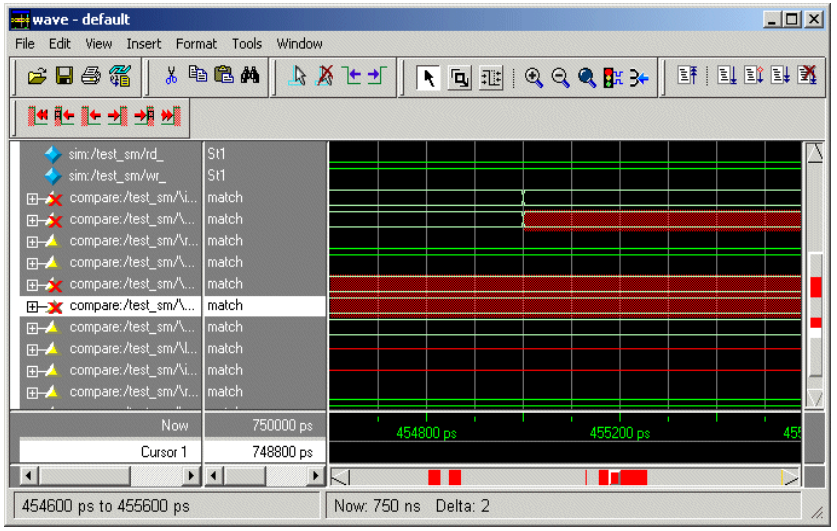
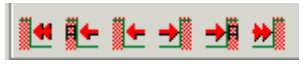


Figure 89: The compare icons



Viewing comparison data in the List window

You can also view the results of your waveform comparison in the List window.

1 Add comparison data to the List window.

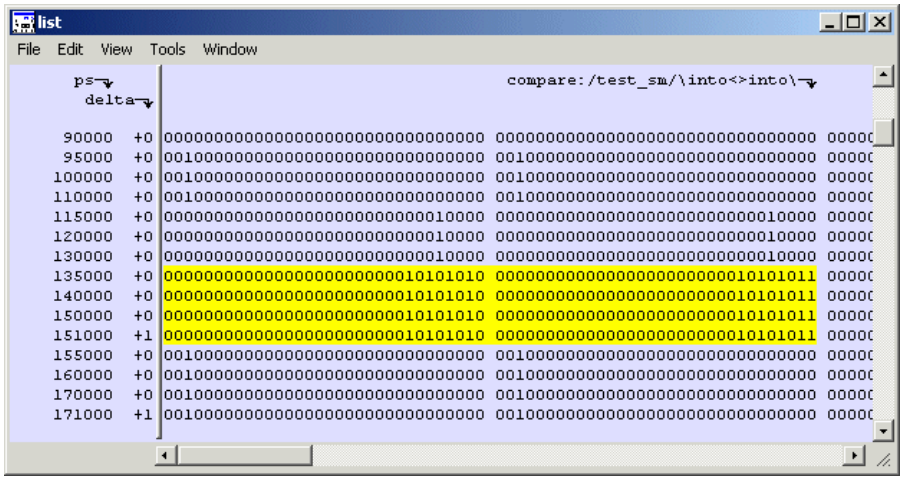
   a Select **View > Debug Windows > List** from the Main window menu bar.

   b Drag the *test_sm* comparison object from the compare tab of the Main window to the List window.

   c Scroll down the window.

   Differences are noted with yellow highlighting (Figure 90). Differences that have been annotated have red highlighting.

Figure 90: Compare differences in the List window



Saving and reloading comparison data

You can save comparison data for later viewing, either in a text file or in files that can be reloaded into ModelSim.

To save comparison data so it can be reloaded into ModelSim, you must save two files. First, you save the computed differences to one file; next, you save the comparison configuration rules to a separate file. When you reload the data, you must have the reference dataset open.

1 Save the comparison data to a text file.

   a Select **Tools > Waveform Compare > Differences > Write Report**.

   b Click **Save**.

   This saved *compare.txt* to the current directory (Figure 91).

   c Type **notepad compare.txt** at the VSIM> prompt.

   d Close Notepad when you are done reviewing the report.

2 Save the comparison data in files that can be reloaded into ModelSim.

   a Select **Tools > Waveform Compare > Differences > Save**.

   b Click **Save**.

   This saved *compare.dif* to the current directory.

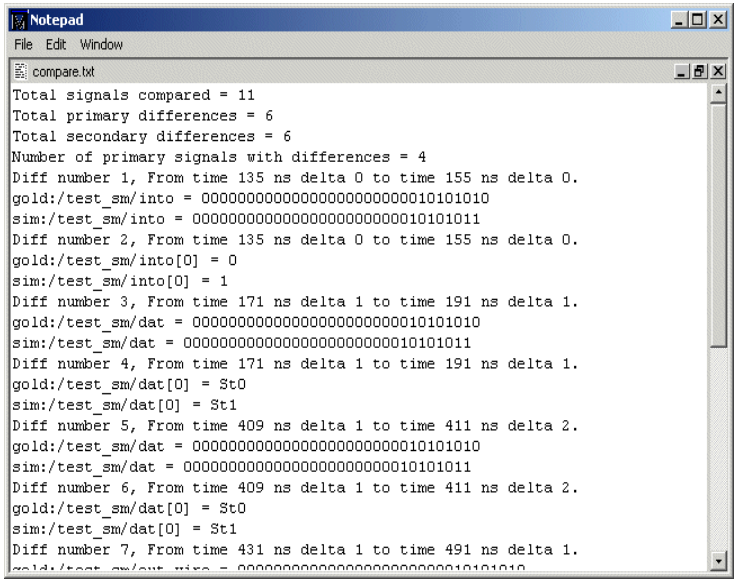
   c Select **Tools > Waveform Compare > Rules > Save**.

   d Click Save.

   This saved *compare.rul* to the current directory.

   e Select **Tools > Waveform Compare > End Comparison**.

Figure 91: Coverage data saved to a text file



3 Reload the comparison data.

a Select **File > Open** and open.

b Change the **Files of Type** to Log Files (*.wlf).

c Double-click *gold.wlf* to open the dataset.

d Select **Tools > Waveform Compare > Reload**.

Since you saved the data using default file names, the dialog should already have the correct files specified (Figure 93).

e Click **OK**.

The comparison reloads.

Figure 92: Displaying log files in the Open dialog

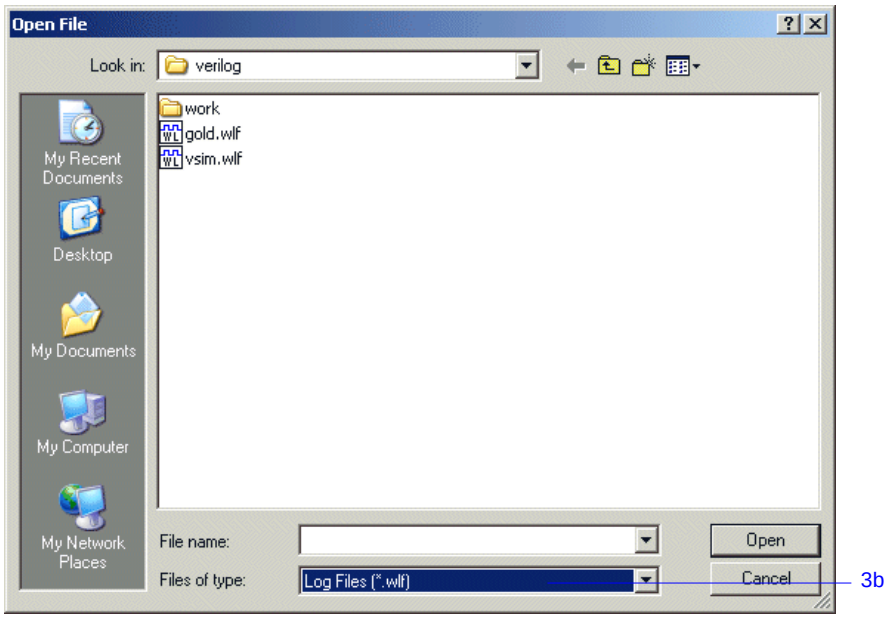
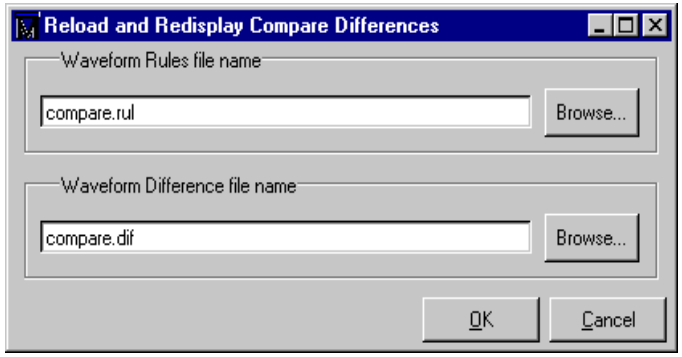


Figure 93: Reloading saved comparison data



Lesson wrap-up

This concludes this lesson. Before continuing we need to end the current simulation and close the *gold.wlf* dataset.

1 Type **quit -sim** at the VSIM> prompt.

2 Type **dataset close gold** at the ModelSim> prompt.

Lesson 10 - Automating ModelSim

Topics

The following topics are covered in this lesson:

Introduction

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

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

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

Related reading

ModelSim User's Manual – 17 - Tcl and macros (DO files) (UM-195)

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

Creating a simple DO file

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

1 Load the *test_counter* design unit.

   a If necessary, start ModelSim.

   b Change to the directory you created in *Lesson 2 - Basic simulation*.

   c In the Library tab of the Workspace pane, double-click the *test_counter* design unit to load it.

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

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

   a Enter the following commands into the source window:

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

3 Save the file.

   a Select **File > Save As**.

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

4 Load the simulation again and use the DO file.

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

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

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

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

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

Running ModelSim in command-line mode

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

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

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

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

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

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

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

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

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

3 Create a DO file.

a Open a text editor.

b Type the following lines into a new file:

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

# read in stimulus
do stim.do

# output results
write list counter.lst

# quit the simulation
quit -f
```

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

4 Run the batch-mode simulation.

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

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

5 View the list output.

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

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

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

6 View the results in the GUI.

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

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

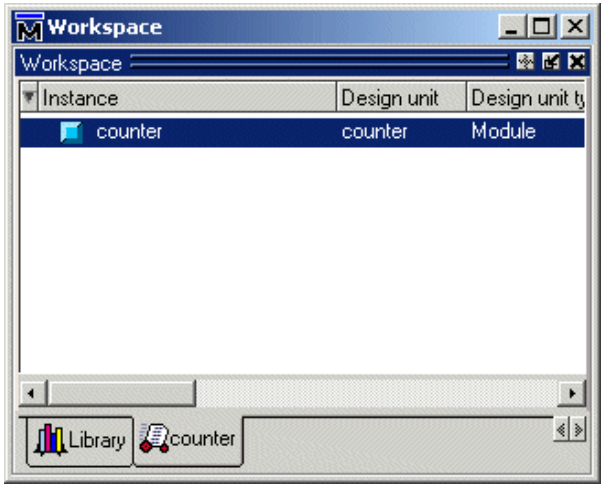
The GUI opens and a dataset tab named "counter" is displayed in the Workspace (Figure 94).

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

The waveforms display in the Wave window.

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

Figure 94: A dataset in the Main window Workspace



Using Tcl with ModelSim

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

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

1 Create the script.

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

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

These commands do the following:

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

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

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

These commands do the following:

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

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

Save it into the directory you created in *Lesson 2 - Basic simulation*.

2 Load the *test_counter* design unit.

a Start ModelSim.

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

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

3 Execute the DO file and run the design.

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

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

c The simulation runs and the DO file creates two bookmarks.Select **View > Bookmarks > bm1**.

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

Lesson Wrap-up

This concludes this lesson.

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

End-User License Agreement

IMPORTANT - USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS. CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE.

This license is a legal "Agreement" concerning the use of Software between you, the end user, either individually or as an authorized representative of the company acquiring the license, and Mentor Graphics Corporation and Mentor Graphics (Ireland) Limited acting directly or through their subsidiaries or authorized distributors (collectively "Mentor Graphics"). USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT. If you do not agree to these terms and conditions, promptly return, or, if received electronically, certify destruction of Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

END-USER LICENSE AGREEMENT

1. **GRANT OF LICENSE.** The software programs you are installing, downloading, or have acquired with this Agreement, including any updates, modifications, revisions, copies, documentation and design data ("Software") are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form; (b) for your internal business purposes; and (c) on the computer hardware or at the site for which an applicable license fee is paid, or as authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or service plan purchased, apply to the following and are subject to change: (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 communicated and technically implemented through the use of authorization codes or similar devices); (c) support services provided, including eligibility to receive telephone support, updates, modifications, and revisions. Current standard policies and programs are available upon request.

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

3. **BETA CODE.** Portions or all of certain Software may contain code for experimental testing and evaluation ("Beta Code"), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to you a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and your use

of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceived or made during or subsequent to this Agreement, including those based partly or wholly on your feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this subsection shall survive termination or expiration of this Agreement.

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

   The terms of this Agreement, including without limitation, the licensing and assignment provisions shall be binding upon your heirs, successors in interest and assigns. The provisions of this section 4 shall survive the termination or expiration of this Agreement.

5. LIMITED WARRANTY.

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

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

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

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

8. **INDEMNIFICATION.** YOU AGREE TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE, OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH YOUR USE OF SOFTWARE AS DESCRIBED IN SECTION 7.

9. INFRINGEMENT.

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

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

9.3. Mentor Graphics has no liability to you if infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the

use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you make, use or sell; (f) any Beta Code contained in Software; (g) any Software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by you that is deemed willful. In the case of (h) you shall reimburse Mentor Graphics for its attorney fees and other costs related to the action upon a final judgment.

9.4. THIS SECTION 9 STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.

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

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

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

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

14. **AUDIT RIGHTS.** With reasonable prior notice, Mentor Graphics shall have the right to audit during your normal business hours all records and accounts as may contain information regarding your compliance with the terms of this Agreement. Mentor Graphics shall keep in confidence all information gained as a result of any audit. Mentor Graphics shall only use or disclose such information as necessary to enforce its rights under this Agreement.

15. **CONTROLLING LAW AND JURISDICTION.** THIS AGREEMENT SHALL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF THE STATE OF OREGON, USA, IF YOU ARE LOCATED IN NORTH OR SOUTH AMERICA, AND THE LAWS OF IRELAND IF YOU ARE LOCATED OUTSIDE OF NORTH AND SOUTH AMERICA. All disputes

arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of Dublin, Ireland when the laws of Ireland apply, or Wilsonville, Oregon when the laws of Oregon apply. This section shall not restrict Mentor Graphics' right to bring an action against you in the jurisdiction where your place of business is located. The United Nations Convention on Contracts for the International Sale of Goods does not apply to this Agreement.

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

17. **PAYMENT TERMS AND MISCELLANEOUS.** You will pay amounts invoiced, in the currency specified on the applicable invoice, within 30 days from the date of such invoice. This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and conditions, except valid license agreements related to the subject matter of this Agreement (which are physically signed by you and an authorized agent of Mentor Graphics) either referenced in the purchase order or otherwise governing this subject matter. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse. The prevailing party in any legal action regarding the subject matter of this Agreement shall be entitled to recover, in addition to other relief, reasonable attorneys' fees and expenses.

Rev. 040401, Part Number 221417

Index

M

N

O

P

Q

R

S

T

U

V

W

X

Z